Lecture Notes in Computer Science 3766

Nicu Sebe   Michael S. Lew
Thomas S. Huang (Eds.)

Computer Vision in Human-Computer Interaction

ICCV 2005 Workshop on HCI
Beijing, China, October 21, 2005
Proceedings

Springer

Volume Editors

Nicu Sebe
University of Amsterdam, Faculty of Science
The Netherlands
E-mail: nicu@science.uva.nl

Michael S. Lew
Leiden University, LIACS Media Lab
Niels Bohrweg 1, 2333 CA Leiden
The Netherlands
E-mail: mlew@liacs.nl

Thomas S. Huang
University of Illinois at Urbana-Champaign
Beckman Institute
Urbana, IL 61801, USA
E-mail: huang@ifp.uiuc.edu

Library of Congress Control Number: 2005934480

CR Subject Classification (1998): I.4, I.5, I.3, H.5.2-3

ISSN 0302-9743
ISBN-10 3-540-29620-4 Springer Berlin Heidelberg New York
ISBN-13 978-3-540-29620-1 Springer Berlin Heidelberg New York

Springer is a part of Springer Science+Business Media

springeronline.com

Typesetting: Camera-ready by author, data conversion by Olgun Computergrafik
Printed on acid-free paper SPIN: 11573425 06/3142 5 4 3 2 1 0

Preface

Human-Computer Interaction (HCI) lies at the crossroads of many scientific areas including artificial intelligence, computer vision, face recognition, motion tracking, etc. In order for HCI systems to interact seamlessly with people, they need to understand their environment through vision and auditory input. Moreover, HCI systems should learn how to adaptively respond depending on the situation.

The goal of this workshop was to bring together researchers from the field of computer vision whose work is related to human-computer interaction. The selected articles for this workshop address a wide range of theoretical and application issues in human-computer interaction ranging from human-robot interaction, gesture recognition, and body tracking, to facial features analysis and human-computer interaction systems.

This year 74 papers from 18 countries were submitted and 22 were accepted for presentation at the workshop after being reviewed by at least 3 members of the Program Committee. We had therefore a very competitive acceptance rate of less than 30% and as a consequence we had a very-high-quality workshop.

We would like to thank all members of the Program Committee for their help in ensuring the quality of the papers accepted for publication. We are grateful to Dr. Jian Wang for giving the keynote address.

In addition, we wish to thank the organizers of the 10th IEEE International Conference on Computer Vision and our sponsors, University of Amsterdam, Leiden Institute of Advanced Computer Science, and the University of Illinois at Urbana-Champaign, for support in setting up our workshop.

August 20, 2005

Nicu Sebe
Michael S. Lew
Thomas S. Huang

IEEE International Workshop on Human-Computer Interaction 2005 (HCI 2005) Organization

Organizing Committee

Nicu Sebe	University of Amsterdam, The Netherlands
Michael S. Lew	Leiden University, The Netherlands
Thomas S. Huang	University of Illinois at Urbana-Champaign, USA

Program Committee

Kiyo Aizawa	University of Tokyo, Japan
Alberto Del Bimbo	University of Florence, Italy
Nozha Boujemaa	INRIA Rocquencourt, France
Kim Boyer	Ohio State University, USA
Edward Chang	University of California, Santa Barbara, USA
Ira Cohen	HP Research Labs, USA
Jeffrey Cohn	University of Pittsburgh, USA
James Crowley	INRIA Rhônes-Alpes, France
Jonathan Foote	FXPAL, USA
Theo Gevers	University of Amsterdam, The Netherlands
Alan Hanjalic	TU Delft, The Netherlands
Thomas S. Huang	University of Illinois at Urbana-Champaign, USA
Alejandro Jaimes	FujiXerox, Japan
Brigitte Kerherve	University of Quebec, Canada
Michael S. Lew	Leiden University, The Netherlands
Frank Nack	CWI, The Netherlands
Jan Nesvadba	Philips Research, The Netherlands
Mark Nixon	University of Southampton, UK
Maja Pantic	TU Delft, The Netherlands
Ioannis Patras	University of York, UK
Vladimir Pavlovic	Rutgers University, USA
Alex Pentland	Massachusetts Institute of Technology, USA
Stan Sclaroff	Boston University, USA
Nicu Sebe	University of Amsterdam, The Netherlands
Qi Tian	University of Texas at San Antonio, USA
Guangyou Xu	Tsinghua University, China
Ming-Hsuan Yang	Honda Research Labs, USA
HongJiang Zhang	Microsoft Research Asia, China
Xiang (Sean) Zhou	Siemens Research, USA

Sponsors

Faculty of Science, University of Amsterdam
Leiden Institute of Advanced Computer Science, Leiden University
Beckman Institute, University of Illinois at Urbana-Champaign

Table of Contents

Tracking

Interfacing

Event Detection

Augmented Reality

Hand and Gesture

Applications

Multimodal Human Computer Interaction: A Survey

Alejandro Jaimes[1] and Nicu Sebe[2]

[1] FXPAL, Fuji Xerox Co., Ltd., Japan
alex.jaimes@fujixerox.co.jp
[2] University of Amsterdam, The Netherlands
nicu@science.uva.nl

Abstract. In this paper we review the major approaches to multimodal human computer interaction from a computer vision perspective. In particular, we focus on body, gesture, gaze, and affective interaction (facial expression recognition, and emotion in audio). We discuss user and task modeling, and multimodal fusion, highlighting challenges, open issues, and emerging applications for Multimodal Human Computer Interaction (MMHCI) research.

1 Introduction

Multimodal Human computer interaction (MMHCI) lies at the crossroads of several research areas including computer vision, psychology, artificial intelligence, and many others. As computers become integrated into everyday objects (ubiquitous and pervasive computing), effective natural human-computer interaction becomes critical: in many applications, users need to be able to interact naturally with computers the way face-to-face human-human interaction takes place. We communicate through speech and use body language (posture, gaze [48], hand motions) to express emotion, mood, attitude, and attention [41].

In human-human communication, interpreting the mix of audio-visual signals is essential in understanding communication. Researchers in many fields recognize this, and thanks to advances in the development of unimodal techniques (in speech and audio processing, computer vision, etc.), and in hardware technologies (inexpensive cameras and sensors), there has been a significant growth in MMHCI research. Unlike in traditional HCI applications (a single user facing a computer and interacting with it via a mouse or a keyboard), in new applications (e.g., intelligent homes [43], remote collaboration, arts, etc.), interactions are not always explicit commands, and often involve multiple users.

Although much progress has been achieved in MMHCI, most researchers still treat each modality (e.g., vision, speech) separately, and integrate the results at the application stage. One reason for this is that the roles of multiple modalities and their interplay remain to be quantified and scientifically understood. Additionally, many open issues remain in processing each modality individually.

In this paper we highlight the main vision problems that in our view should be solved for successful MMHCI applications, and give an overview of the research areas we consider essential for MMHCI. We group vision techniques according to the human body (Figure 1). Large-scale body movement, gesture (e.g., hands), and gaze analysis are used for tasks such as emotion recognition in affective interaction, and for a variety of applications. We discuss affective computer interaction, issues in multi-modal fusion, modeling, and data collection, and a variety of emerging MMHCI

N. Sebe, M.S. Lew, and T.S. Huang (Eds.): HCI/ICCV 2005, LNCS 3766, pp. 1–15, 2005.

applications. Since MMHCI is a very dynamic and broad research area we do not intend to present a complete survey. The main contribution of this paper, therefore, is to consolidate some of the main issues and approaches, and to highlight some of the techniques and applications developed recently within the context of MMHCI.

1.1 Related Surveys

Extensive surveys have been previously published in several areas such as face detection [88][26], face recognition [91], facial expression analysis [17][54], vocal emotion [46][95], gesture recognition [38][78][57], human motion analysis [27][83][84][22] [1][44], and eye tracking [12]. A review of vision-based HCI is presented in [62] with a focus on head tracking, face and facial expression recognition, eye tracking, and gesture recognition. Adaptive and intelligent HCI is discussed in [14] with a review of computer vision for human motion analysis, and a discussion of techniques for lower arm movement detection, face processing, and gaze analysis. Multimodal interfaces are discussed in [49][50][51][52][69]. Real-time vision for HCI (gestures, object tracking, hand posture, gaze) is discussed in [33]. Here, we discuss work not included in previous surveys, expand the discussion to areas not covered previously (e.g., in [33][14][62][50]), and discuss new applications in emerging areas while highlighting the main research issues.

2 Overview of Multimodal Interaction

The term multimodal has been used in many contexts and across several disciplines. For our interests, *a multimodal HCI system is simply one that responds to inputs in more than one modality or communication channel* (e.g., speech, gesture, writing, and others). We use a human-centered approach in our definition: by modality we mean mode of communication according to human senses *or* type of computer input devices. In terms of human senses the categories are *sight, touch, hearing*, *smell,* and *taste*. In terms of computer input devices we have modalities that are equivalent to human senses: cameras (*sight*), haptic sensors (*touch*), microphones (*hearing*), olfactory (*smell*), and even taste [36]. In addition, however, there are input devices that do not map directly to human senses: keyboard, mouse, writing tablet, motion input (e.g., the device itself is moved for interaction), and many others.

In our definition, a system that uses any combination of modalities in the categories above is multimodal. For our purposes, however, interest is exclusively on systems that include vision (cameras) as a modality[1]. A system that responds only to facial expressions and hand gestures, for example, is not multimodal, even if integration of both inputs (simultaneous or not) is used (using the same argument, a system with multiple keys is not multimodal, but a system with mouse a keyboard input is). The issue of where integration of modalities takes place, if at all, is of great importance and is discussed throughout the paper.

As depicted in Figure 1, we place input modalities in two major groups: based on human senses (*vision*, *audio*, *haptic*, *olfactory* and *touch*), and others (mouse, key-

[1] Others have studied multimodal interaction using multiple devices such as mouse and keyboard, keyboard and pen, and so on

board, etc.). The visual modality includes any form of interaction that can be interpreted visually, and the audio modality any form that is audible (including multi-language input). We only discuss vision in detail, but as many new applications show (see Section 6), other modalities have gained importance for interaction (e.g., haptic [4]).

As depicted in Figure 1, multimodal techniques can be used to construct a variety of interfaces. Of particular interest for our goals are perceptual and attentive interfaces. Perceptual interfaces [80] as defined in [81], are highly interactive, multimodal interfaces that enable rich, natural, and efficient interaction with computers. Perceptual interfaces seek to leverage sensing (input) and rendering (output) technologies in order to provide interactions not feasible with standard interfaces and common I/O devices such as the keyboard, the mouse and the monitor [81]. Attentive interfaces, on the other hand, are context-aware interfaces that rely on a person's attention as the primary input [71] — the goal of these interfaces [47] is to use gathered information to estimate the best time and approach for communicating with the user.

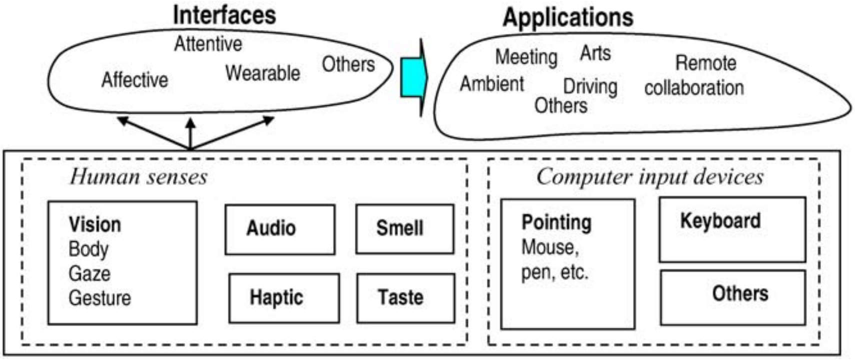


Fig. 1. Overview of multimodal interaction using a human-centered approach

Vision plays a fundamental role in several types of interfaces. As argued in [71], attention is epitomized by eye contact (even though other measures, such as cursor movement can also be indicative). Perceptual interfaces aim at natural interaction, making vision an essential component. The key point is that vision plays a major role in human-computer interfaces that aim at natural interaction. As we will see in Section 6, vision in multimodal interaction is applied in a variety of applications and interface types.

Although there have been many advances in MMHCI, as our discussions will show, the majority of research approaches focus on one mode independently and fuse the results at the highest level possible (in the application). Accordingly, in the next section we survey Computer Vision techniques for MMHCI and in the following sections we discuss fusion, interaction, and applications.

3 Core Vision Techniques

We classify vision techniques for MMHCI using a human-centered approach and divide them according to how humans may interact with the system: (1) large-scale body movements, (2) gestures, and (3) gaze. We make a distinction between *command* (actions can be used to explicitly execute commands: select menus, etc.) and *non-command* interfaces (actions or events used to indirectly tune the system to the user's needs) [45][7].

In general, vision-based human motion analysis systems used for MMHCI can be thought of as having mainly 4 stages: (1) motion segmentation, (2) object classification, (3) tracking, and (4) interpretation. While some approaches use geometric primitives to model different components (e.g., cylinders for limbs, head, and torso for body movements, or for hand and fingers in gesture recognition), others use feature representations based on appearance. In the first approach, external markers are often used to estimate body posture and relevant parameters. While markers can be accurate, they place restrictions on clothing and require calibration, so they are not desirable in many applications. Appearance based methods, on the other hand, do not require markers, but require training (e.g., with machine learning, probabilistic approaches, etc.). Methods that do not require markers place fewer constraints on the user and are more desirable, as are those that do not use geometric primitives (which are computationally expensive and often not suitable for real-time processing).

Next, we discuss some specific techniques for body, gesture, and gaze. The motion analysis steps are similar, so there is some inevitable overlap in the discussions. Some of the issues for gesture recognition, for instance, apply to body movements and gaze detection.

3.1 Large-Scale Body Movements

Tracking of large-scale body movements (head, arms, torso, and legs) is necessary to interpret pose and motion in many MMHCI applications Since extensive surveys have been published [83][84][22][1][44], we discuss the topic briefly.

The authors of [87] identify three important issues in articulated motion analysis: representation (joint angles or motion of all the sub-parts), computational paradigms (deterministic or probabilistic), and computation reduction. They propose a dynamic Markov network that uses Mean Field Monte Carlo algorithms so that a set of low dimensional particle filters interact with each other to solve a high dimensional problem collaboratively.

Body posture analysis is important in many MMHCI applications. In [77], the authors use a stereo and thermal infrared video system to estimate driver posture for deployment of smart air bags. The authors of [64] propose a method for recovering articulated body pose without initialization and tracking (using learning). The authors of [3] use pose and velocity vectors to recognize body parts and detect different activities, while the authors of [5] use temporal templates.

In some emerging MMHCI applications, group and non-command actions play an important role. The authors of [40] present an approach to segment a meeting according to actions such as monologue, presentation, white-board, discussion, and note taking. HMMs are used with a combination of audiovisual features. Visual features are extracted from head and hand/forearm blobs: the head blob is represented by the vertical position of its centroid, and hand blobs are represented by eccentricity and angle with respect to the horizontal. Audio features include energy, pitch, and speaking rate, among others. The authors of [24] use only computer vision, but make a distinction between body movements, events, and behaviors, within a rule-based system framework.

Important issues for large-scale body tracking include whether the approach uses 2D or 3D, desired accuracy, speed, occlusion and other constraints. Some of the issues pertaining to gesture recognition, discussed next, can also apply to body tracking.

3.2 Gesture Recognition

Psycholinguistic studies for human-to-human communication [41] describe gestures as the critical link between our conceptualizing capacities and our linguistic abilities. Humans use a very wide variety of gestures ranging from simple actions of using the hand to point at objects to the more complex actions that express feelings and allow communication with others. Gestures should therefore play an essential role in MMHCI [32][86][19]. A major motivation for these research efforts is the potential of using hand gestures in various applications aiming at natural interaction between the human and the computer-controlled interface. These applications range from virtual environments [31], to smart surveillance [78] and remote collaboration [19].

There are several important issues that should be considered when designing a gesture recognition system [57]. The first phase of a recognition task is choosing a mathematical model that may consider both the spatial and the temporal characteristics of the hand and hand gestures. The approach used for modeling plays a crucial role in the nature and performance of gesture interpretation. Once the model is detected, an analysis stage is required for computing the model parameters from the features that are extracted from single or multiple input streams. These parameters represent some description of the hand pose or trajectory and depend on the modeling approach used. Among the important problems involved in the analysis are that of hand localization [94], hand tracking [89], and the selection of suitable features [32]. After the parameters are computed, the gestures represented by them need to be classified and interpreted based on the accepted model and based on some grammar rules that reflect the internal syntax of gestural commands. The grammar may also encode the interaction of gestures with other communication modes such as speech, gaze, or facial expressions. As an alternative, some authors have explored using combinations of simple 2D motion based detectors for gesture recognition [29].

In any case, to fully exploit the potential of gestures for an MMHCI application, the class of possible recognized gestures should be as broad as possible and ideally any gesture preformed by the user should be unambiguously interpretable by the interface. However, most of the gesture-based HCI systems allow only symbolic commands based on hand posture or 3D pointing. This is due to the complexity associated with gesture analysis and the desire to build real-time interfaces. Also, most of the systems accommodate only single-hand gestures. Yet, human gestures, especially communicative, naturally employ actions of both hands. However, if the two-hand gestures are to be allowed, several ambiguous situations may appear (e.g., occlusion of hands, intentional vs. unintentional, etc.) and the processing time will likely increase. Another important aspect that is increasingly considered is the use of other modalities (e.g., speech) to augment the MMHCI system [51][72]. The use of such multimodal approaches can reduce the complexity and increase the naturalness of the interface for MMHCI [50].

3.3 Gaze Detection

Gaze, defined as the direction to which the eyes are pointing in space, is a strong indicator of attention, and it has been studied extensively since as early as 1879 in psychology, and more recently in neuroscience and in computing applications [12]. While early eye tracking research focused only on systems for in-lab experiments, many commercial and experimental systems are available today for a wide range of applications.

Eye tracking systems can be grouped into wearable or non-wearable, and infrared-based or appearance-based. In infrared-based systems, a light shining on the subject whose gaze is to be tracked creates a “red-eye effect:” the difference in reflection between the cornea and the pupil is used to determine the direction of sight. In appearance-based systems, computer vision techniques are used to find the eyes in the image and then determine their orientation. While wearable systems are the most accurate (approximate error rates under 1.4° vs. errors under 1.7° for non-wearable infrared), they are also the most intrusive. Infrared systems are more accurate than appearance-based, but there are concerns over the safety of prolonged exposure to infrared lights. In addition, most non-wearable systems require (often cumbersome) calibration for each individual.

Appearance-based systems use both eyes to predict gaze direction, so the resolution of the image of each eye is often small, which makes them less accurate. In [82], the authors propose using a single high-resolution image of one eye to improve accuracy. Infrared-based systems usually use only one camera. The authors of [66] have proposed using multiple cameras to improve accuracy.

One trend has been to improve non-wearable systems for use in MMHCI and other applications where the user is stationary (e.g., [74][66]). For example, the authors of [74] monitor driver visual attention using a single, non-wearable camera placed on a car’s dashboard to track face features and for gaze detection.

There have also been advances in wearable eye trackers for novel applications. In [90], eye tracking data is combined with video from the user’s perspective, head directions, and hand motions to learn words from natural interactions with users; the authors of [58] use a wearable eye tracker to understand hand-eye coordination in natural tasks, and the authors of [13] use a wearable eye tracker to detect eye contact and record video for blogging.

The main issues in developing gaze tracking systems are intrusiveness, speed, robustness, and accuracy. The type of hardware and algorithms necessary, however, depend highly on the level of analysis desired. Gaze analysis can be performed at three different levels [7]: (a) highly detailed low-level micro-events, (b) low-level intentional events, and (c) coarse-level goal-based events. Micro-events include micro-saccades, jitter, nystagmus, and brief fixations, which are studied for their physiological and psychological relevance by vision scientists and psychologists. Low-level intentional events are the smallest coherent units of movement that the user is aware of during visual activity, which include sustained fixations and revisits. Although most of the work on HCI has focused on coarse-level goal-based events (e.g., using gaze as a pointer [73]), it is easy to foresee the importance of analysis at lower levels, particularly to infer the user’s cognitive state in affective interfaces (e.g., [25]). Within this context, an important issue often overlooked is how to interpret eye-tracking data (see [67] for discussion on eye tracking data clustering).

4 Affective Human-Computer Interaction

There is a vast body of literature on affective computing and emotion recognition [2][55][61]. Affective states are intricately linked to other functions such as attention, perception, memory, decision-making, and learning [15]. This suggests that it may be beneficial for computers to recognize the user's emotions and other related cognitive states and expressions.

Researchers use mainly two different methods to analyze emotions. One approach is to classify emotions into discrete categories such as *joy, fear, love, surprise, sadness,* etc., using different modalities as inputs to emotion recognition models. The problem is that the stimuli may contain blended emotions and the choice of these categories may be too restrictive, or culturally dependent. Another way is to have multiple dimensions or scales to describe emotions. Two common scales are valence and arousal. Valence describes the pleasantness of the stimuli, with positive or pleasant (e.g, *happiness*) on one end, and negative or unpleasant (e.g., *disgust)* on the other. The other dimension is arousal or activation. For example, *sadness* has low arousal, whereas *surprise* has a high arousal level. The different emotional labels could be plotted at various positions on a two-dimensional plane spanned by these two axes to construct a 2D emotion model [35][23].

Facial expressions and vocal emotions are particularly important in this context, so we discuss them in more detail below.

4.1 Facial Expression Recognition

Most facial expression recognition research (see [54] and [17] for two comprehensive reviews) has been inspired by the work of Ekman [15] on coding facial expressions based on the basic movements of facial features called action units (AUs). In this scheme, expressions are classified into a predetermined set of categories. Some methods follow a “feature-based” approach, where one tries to detect and track specific features such as the corners of the mouth, eyebrows, etc. Other methods use a “region-based” approach in which facial motions are measured in certain regions on the face such as the eye/eyebrow and the mouth. In addition, we can distinguish two types of classification schemes: dynamic and static. Static classifiers (e.g., Bayesian Networks) classify each frame in a video to one of the facial expression categories based on the results of a particular video frame. Dynamic classifiers (e.g., HMM) use several video frames and perform classification by analyzing the temporal patterns of the regions analyzed or features extracted. They are very sensitive to appearance changes in the facial expressions of different individuals so they are more suited for person-dependent experiments [10]. Static classifiers, on the other hand, are easier to train and in general need less training data but when used on a continuous video sequence they can be unreliable especially for frames that are not at the peak of an expression.

4.2 Emotion in Audio

The vocal aspect of a communicative message carries various kinds of information. If we disregard the manner in which a message is spoken and consider only the textual content, we are likely to miss the important aspects of the utterance and we might even completely misunderstand the meaning of the message. Nevertheless, in contrast to spoken language processing, which has recently witnessed significant advances, the processing of emotional speech has not been widely explored.

Starting in the 1930s, quantitative studies of vocal emotions have had a longer history than quantitative studies of facial expressions. Traditional as well as most recent studies on emotional contents in speech (see [46], [95], and [68]) use "prosodic" information which includes the pitch, duration, and intensity of the utterance. Recent studies seem to use the "Ekman six" basic emotions, although others in the past have used many more categories. The reasons for using these basic categories are often not justified since it is not clear whether there exist "universal" emotional characteristics in the voice for these six categories [11].

The most surprising issue regarding the multimodal affect recognition problem is that although recent advances in video and audio processing could make the multimodal analysis of human affective state tractable, there are only a few research efforts [30][70][92] that have tried to implement a multimodal affective analyzer.

5 Modeling, Fusion, and Data Collection

5.1 User, Context, and Task Modeling

Multimodal interface design [63] is important because the principles and techniques used in traditional GUI-based interaction do not necessarily apply in MMHCI systems. Issues to consider, as identified in [63] include design of inputs and outputs, adaptability, consistency, and error handling, among others. In addition, one must consider dependency of a person's behavior on his/her personality, cultural, and social vicinity, current mood, and the context in which the observed behavioral cues are encountered.

Many design decisions dictate the underlying techniques used in the interface. For example, adaptability can be addressed using machine learning: rather than using a priori rules to interpret human behavior, we can potentially learn application-, user-, and context-dependent rules by watching the user's behavior in the sensed context [59]. Probabilistic graphical models have an important advantage here: well known algorithms exist to adapt the models, and it is possible to use prior knowledge when learning new models. For example, a prior model of emotional expression recognition trained based on a certain user can be used as a starting point for learning a model for another user, or for the same user in a different context. Although context sensing and the time needed to learn appropriate rules are significant problems in their own right, many benefits could come from such adaptive MMHCI systems.

5.2 Fusion

A typical issue of multimodal data processing is that multisensory data is typically processed separately and only combined at the end. Yet, people convey multimodal (e.g., audio and visual) communicative signals in a complementary and redundant manner (as shown experimentally by Chen [11]). Therefore, in order to accomplish a human-like multimodal analysis of multiple input signals acquired by different sensors, the signals cannot be considered mutually independently and cannot be combined in a context-free manner at the end of the intended analysis but, on the contrary, the input data should be processed in a joint feature space and according to a context-dependent model. In practice, however, besides the problems of context sensing and developing context-dependent models for combining multisensory information, one should cope with the size of the required joint feature space. Problems include large dimensionality, differing feature formats, and time-alignment. A potential way to achieve multisensory data fusion is to develop context-dependent versions of a suitable method such as the Bayesian inference method proposed by Pan et al. [53].

In spite of its importance, the problem of fusing multiple modalities is often largely ignored. For example, the studies in facial expression recognition and vocal affect recognition have been done largely independent of each other. Most works in facial expression recognition use still photographs or video sequences without speech. Similarly, works on vocal emotion detection often use only audio information. A legitimate question that should be considered in MMHCI, is how much information does the face, as compared to speech, and body movement, contribute to natural interaction. Most experimenters suggest that the face is more accurately judged, produces higher agreement, or correlates better with judgments based on full audiovisual input than on voice input [42].

A multimodal system should be able to deal with imperfect data and generate its conclusion so that the certainty associated with it varies in accordance to the input data. A way of achieving this is to consider the time-instance versus time-scale dimension of human nonverbal communicative signals [55]. By considering previously observed data (time scale) with respect to the current data carried by functioning observation channels (time instance), a statistical prediction and its probability might be derived about both the information that has been lost due to malfunctioning/inaccuracy of a particular sensor and the currently displayed action/reaction. Probabilistic graphical models, such as Hidden Markov Models (including their hierarchical variants), Bayesian networks, and Dynamic Bayesian networks are very well suited for fusing such different sources of information. These models can handle noisy features, temporal information, and missing values of features all by probabilistic inference. Hierarchical HMM-based systems [10] have been shown to work well for facial expression recognition. Dynamic Bayesian Networks and HMM variants [21] have been shown to fuse various sources of information in recognizing user intent, office activity recognition, and event detection in video using both audio and visual information [20]. This suggests that probabilistic graphical models are a promising approach to fusing realistic (noisy) audio and video for context-dependent detection of behavioral events such as affective states.

Despite important advances, further research is still required to investigate fusion models able to efficiently use the complementary cues provided by multiple modalities.

5.3 Data Collection and Testing

Collecting MMHCI data and obtaining the ground truth for it is a challenging task. Labeling is time-consuming, error prone, and expensive. In developing multimodal techniques for emotion recognition, for example, one approach consists of asking actors to read material aloud while simultaneously portraying particular emotions chosen by the investigators. Another approach is to use emotional speech from real conversations or to induce emotions from speakers using various methods (e.g., showing photos or videos to induce reactions). Using actor portrayals ensures control of the verbal material and the encoder's intention, but raises the question about the similarity between posed and naturally occurring expressions. Using real emotional speech, on the other hand, ensures high validity, but renders the control of verbal material and encoder intention more difficult. Induction methods are effective in inducing moods, but it is harder to induce intense emotional states in controlled laboratory settings.

In general, collection of data for an MMHCI application is challenging because there is wide variability in the set of possible inputs (consider the number of possible gestures), often only a small set of training examples is available, and the data is often noisy. Therefore, it is very beneficial to construct methods that use scarcely available labeled data and abundant unlabeled data.

Probabilistic graphical models are ideal candidates for tasks in which labeled data is scarce, but abundant unlabeled data is available. Efficient and convergent probabilistic graphical model algorithms exist for handling missing and unlabeled data. Cohen et al. [9] showed that unlabeled data can be used together with labeled data for MMHCI applications using Bayesian networks. However, they have shown that care must be taken when attempting such schemes. In the purely supervised case (only labeled data), adding more labeled data always improves the performance of the classifier. Adding unlabeled data, however, can be detrimental to the performance. Such detrimental effects occur when the assumed classifier's model does not match the data's distribution.

To conclude, further research is necessary to achieve maximum utilization of unlabeled data for MMHCI problems since it is clear that such methods could provide great benefit.

6 Applications

Throughout the paper we have discussed techniques applied in a wide variety of application scenarios, including video conferencing and remote collaboration, intelligent homes, and driver monitoring.

As many of these applications show, the model of user interface in which one person sits in front of a computer is quickly changing. In some cases, the actions or events to be recognized are not explicit commands. In smart conference room applications, multimodal analysis has been applied mostly for video indexing [40] (see [60] for a social analysis application). Although such approaches are not meant to be used in real-time, they are useful in investigating how multiple modalities can be fused in interpreting communication. It is easy to foresee applications in which "smart meeting rooms" actually react to multimodal actions in the same way that intelligent homes should [43].

Perhaps one of the most exciting application areas of MMHCI is art. Vision techniques can be used to allow audience participation [39] and influence a performance. In [85], the authors use multiple modalities (video, audio, pressure sensors) to output different "emotional states" for Ada, an intelligent space that responds to multimodal input from its visitors. In [37], a wearable camera pointing at the wearer's mouth interprets mouth gestures to generate MIDI sounds (so a musician can play other instruments while generating sounds by moving his mouth). In [56], limb movements are tracked to generate music. MMHCI can also be used in museums to augment exhibitions [76].

Robotics is yet another interesting area for MMHCI. The authors of [18] give a comprehensive review of socially active robots and discuss the role of "human-oriented perception" (speech, gesture, and gaze).

People with disabilities can benefit greatly from MMHCI technologies [34]. The authors of [75] propose a component-based smart wheel chair system and discuss other approaches that integrate various types of sensors (not only vision). In [12], computer vision is used to interpret facial gestures for wheel chair navigation. In [6], the authors present two techniques (head tilt and gesture with audio feedback) to control a mobile device. The approach could be beneficial for people with disabilities, but it points to another interesting area: use of MMHCI for mobile devices that have limited input/output resources. Finally, [65] introduces a system for presenting digital pictures non-visually (multimodal output). Other important application areas include gaming [92], and education, "safety-critical applications" (e.g., medicine, military, etc. [8]) among others.

7 Conclusion

We have highlighted major vision approaches for multimodal human-computer interaction. We discussed techniques for large-scale body movement, gesture recognition, and gaze detection. We discussed facial expression recognition, emotion analysis from audio, user and task modeling, multimodal fusion, and a variety of emerging applications.

One of the major conclusions of this survey is that most researchers process each channel (visual, audio) independently, and multimodal fusion is still in its infancy. On one hand, the whole question of how much information is conveyed by "separate" channels may inevitably be misleading. There is no evidence that individuals in actual social interaction selectively attend to another person's face, body, gesture, or speech, or that the information conveyed by these channels is simply additive. The central mechanisms directing behavior cut across channels, so that, for example, certain aspects of face, body, and speech are more spontaneous and others are more closely monitored and controlled. It might well be that observers selectively attend not to a particular channel but to a particular type of information (e.g., cues to emotion, deception, or cognitive activity), which may be available within several channels. No investigator has yet explored this possibility or the possibility that different individuals may typically attend to different types of information.

Another important issue is the affective aspect of communication that should be considered when designing an MMHCI system. Emotion modulates almost all modes of human communication—facial expression, gestures, posture, tone of voice, choice of words, respiration, skin temperature and clamminess, etc. Emotions can significantly change the message: often it is not what was said that is most important, but how it was said. As noted by Picard [61] affect recognition is most likely to be accurate when it combines multiple modalities, information about the user's context, situation, goal, and preferences. A combination of low-level features, high-level reasoning, and natural language processing is likely to provide the best emotion inference in the context of MMHCI. Considering all these aspects, Pentland [59] believes that multimodal context-sensitive human-computer interaction is likely to become the single most widespread research topic of the artificial intelligence research community. Advances in this area could change not only how professionals practice computing, but also how mass consumers interact with technology.

References

1. J.K. Aggarwal and Q. Cai, "Human motion analysis: A review," *CVIU*, 73(3):428-440, 1999.
2. Application of Affective Computing in Human-computer Interaction, *Int. J. of Human-Computer Studies*, 59(1-2), 2003.
3. J. Ben-Arie, Z. Wang, P. Pandit, and S. Rajaram, "Human activity recognition using multidimensional indexing," *IEEE Trans. On PAMI*, 24(8):1091-1104, 2002.
4. M. Benali-Khoudja, M. Hafez, J.-M. Alexandre, and A. Kheddar, "Tactile interfaces: a state-of-the-art survey*," Int. Symposium on Robotics*, 2004.
5. A.F. Bobick and J. Davis, "The recognition of human movement using temporal templates," *IEEE Trans. on PAMI*, 23(3):257–267, 2001.
6. S.A. Brewster, J. Lumsden, M. Bell, M. Hall, S. and Tasker," Multimodal 'Eyes-Free' Interaction Techniques for Wearable Devices," in proc. *ACM CHI 2003*.
7. C. S. Campbell and P.P. Maglio, "A Robust Algorithm for Reading Detection," *ACM Workshop on Perceptive User Interfaces*, 2001.
8. P.R. Cohen, D.R. McGee, "Tangible Multimodal Interfaces for Safety-critical Applications," *Communications of the ACM*, Vol. 47, Issue 1, pp. 41-46, January 2004.
9. I. Cohen, N. Sebe, F. Cozman, M. Cirelo, and T.S. Huang, "Semi-supervised learning of classifiers: Theory, algorithms, and their applications to human-computer interaction," *IEEE Trans. on PAMI*, 22(12):1553-1567, 2004.
10. I. Cohen, N. Sebe, A. Garg, L. Chen, and T.S. Huang, "Facial expression recognition from video sequences: Temporal and static modeling," *CVIU*, 91(1-2):160–187, 2003.
11. L.S. Chen, *Joint processing of audio-visual information for the recognition of emotional expressions in human-computer interaction*, PhD thesis, UIUC, 2000.
12. A.T. Duchowski, "A Breadth-First Survey of Eye Tracking Applications," *Behavior Research Methods, Instruments, and Computing*, 34(4):455-70, 2002.
13. C. Dickie, R. Vertegaal, D. Fono, C. Sohn, D. Chen, D. Cheng, J.S. Shell and O. Aoudeh, "Augmenting and Sharing Memory with eyeBlog," in *CARPE* 2004.
14. Z. Duric, W. Gray, R. Heishman, F. Li, A. Rosenfeld, M. Schoelles, C. Schunn, and H. Wechsler, "Integrating perceptual and cognitive modeling for adaptive and intelligent human-computer interaction," *Proc. of the IEEE*, 90(7):1272-1289, 2002.
15. P. Ekman, ed., *Emotion in the Human Face*, Cambridge University Press, 1982.
16. C. Fagiani, M. Betke, and J. Gips, "Evaluation of tracking methods for human-computer interaction," *IEEE Workshop on Applications in Computer Vision*, 2002.
17. B. Fasel and J. Luettin, "Automatic facial expression analysis: A survey" *Patt. Recogn.*, 36:259–275, 2003.
18. T. Fong, I. Nourbakhsh, K. Dautenhahn, "A survey of socially interactive robots," *Robotics and Autonomous Systems,* 42(3-4):143-166, 2003.
19. S. Fussell, L. Setlock, J. Yang, J. Ou, E. Mauer, A. Kramer, "Gestures over video streams to support remote collaboration on physical tasks," *Human-Computer Int.*, 19(3):273-309, 2004.
20. A. Garg, M. Naphade, and T.S. Huang, "Modeling video using input/output Markov models with application to multi-modal event detection," *Handbook of Video Databases: Design and Applications*, 2003.

21. I. Garg, V. Pavlovic, and J. Rehg. “Boosted learning in dynamic Bayesian networks for multimodal speaker detection,” *Proceedings of the IEEE*, 91(9):1355–1369, 2003.
22. D.M. Gavrila, “The Visual Analysis of Human Movement: A Survey,” *CVIU*, 73(1):82-98, 1999.
23. A. Hanjalic and L-Q. Xu, “Affective video content representation and modeling,” *IEEE Trans. on Multimedia,* 7(1):143– 154, 2005.
24. A. Hakeem and M. Shah, “Ontology and taxonomy collaborated framework for meeting classification,” *ICPR*, 2004.
25. R. Heishman, Z. Duric, and H. Wechsler, “Using eye region biometrics to reveal affective and cognitive states,” *CVPR Workshop on Face Processing in Video*, 2004.
26. E. Hjelmas and B. K. Low, "Face detection: A survey," *CVIU*, 83:236–274, 2001.
27. W. Hu, T. Tan, L. Wang, and S. Maybank, “A Survey on Visual Surveillance of Object Motion and Behaviors”, *in IEEE Trans. On Systems, Man, and Cybernetics*, Vol. 34, No. 3, Aug. 2004.
28. S. Intille, K. Larson, J. Beaudin, J. Nawyn, E. Tapia, P. Kaushik, "A living laboratory for the design and evaluation of ubiquitous computing technologies," *Conf. on Human Factors in Computing Systems*, 2004.
29. A. Jaimes and J. Liu, “Hotspot Components for Gesture-Based Interaction,” in proc. *IFIP Interact 2005*, Rome, Italy, Sept. 2005.
30. R. El Kaliouby and P. Robinson: Real time inference of complex mental states from facial expressions and head gestures, *CVPR Workshop on Real-time Vision for HCI*, 2004.
31. S. Kettebekov and R. Sharma, “Understanding gestures in multimodal human computer interaction,” *Int. J. on Artificial Intelligence Tools*, 9(2):205-223, 2000.
32. T. Kirishima, K. Sato, and K. Chihara, “Real-time gesture recognition by learning and selective control of visual interest points,” *IEEE Trans. on PAMI*, 27(3):351–364, 2005.
33. T. Kisacanin, V. Pavlovic, and T.S.Huang (eds.). *Real-Time Vision for Human-Computer Interaction*, Springer-Verlag, New York, 2005.
34. Y. Kuno, N. Shimada, and Y. Shirai, “Look where you're going: A robotic wheelchair based on the integration of human and environmental observations,” *IEEE Robotics and Automation*, 10(1):26-34, 2003.
35. P. Lang, “The emotion probe: Studies of motivation and attention,” *American Psychologist*, 50(5):372–385, 1995.
36. A. Legin, A. Rudnitskaya, B. Seleznev, Yu. Vlasov, “Electronic tongue for quality assessment of ethanol, vodka and eau-de-vie,” *Analytica Chimica Acta*, Vol. 534, pp. 129-135, April 2005.
37. M.J. Lyons, M. Haehnel, and N. Tetsutani, “Designing, playing, and performing, with a vision-based mouth Interface,” *Conf. on New Interfaces for Musical Expression*, 2003.
38. S. Marcel, “Gestures for multi-modal interfaces: A Review,” *Technical Report IDIAP-RR 02-34*, 2002.
39. D. Maynes-Aminzade, R. Pausch, and S. Seitz, “Techniques for interactive audience participation,” *ICMI* 2002.
40. I. McCowan, D. Gatica-Perez, S. Bengio, G. Lathoud, M. Barnard, and D. Zhang, “Automatic analysis of multimodal group actions in meetings,” *IEEE Trans. on PAMI*, 27(3):305-317, 2005.
41. D. McNeill, *Hand and Mind: What Gestures Reveal About Thought,* Univ. of Chicago Press, Chicago, IL, 1992.
42. A. Mehrabian, “Communication without words,” *Psychology Today*, 2(4):53–56, 1968.
43. S. Meyer and A. Rakotonirainy, “A Survey of research on context-aware homes,” *Australasian Information Security Workshop Conference on ACSW Frontiers*, 2003
44. T. B. Moeslund and E. Granum, “A survey of computer vision-based human motion capture,” *CVIU*, 81(3):231-258, 2001.
45. J. Nielsen, “Non-command user interfaces,” *Comm. of the ACM,* 36(4):83-99, 1993.

46. P.Y. Oudeyer, “The production and recognition of emotions in speech: Features and algorithms,” *Int. J. of Human-Computer Studies*, 59(1-2):157–183, 2003.
47. A. Oulasvirta and A. Salovaara, “A cognitive meta-analysis of design approaches to interruptions in intelligent environments,” in *Proceedings of ACM Conference on Human Factors in Computing Systems (CHI'04): Extended Abstracts (2004).*
48. P. Qvarfordt, and S. Zhai, “Conversing with the user based on eye-gaze patterns,” *Conf. Human-Factors in Computing Syst.*, 2005.
49. S. Oviatt, T. Darrell, and M. Flickner, “Multimodal Interfaces that Flex, Adapt, and Persist,” (eds.) special issue, *Communications of the ACM*, Vol. 47, Issue 1, January 2004.
50. S.L. Oviatt and P. Cohen, “Multimodal interfaces that process what comes naturally,” *Comm. of the ACM,* 43(3):45-48, 2000.
51. S.L. Oviatt, “Multimodal interfaces,” *Human-Computer Interaction Handbook: Fundamentals, Evolving Technologies and Emerging Applications*, chap.14, 286-304, 2003.
52. S.L. Oviatt, P. Cohen, L. Wu, J. Vergo, L. Duncan, B. Suhm, J. Bers, T. Holzman, T. Winograd, J. Landay, J. Larson, and D. Ferro, “Designing the user interface for multimodal speech and pen-based gesture applications: State-of-the-art systems and future research directions,” *Human-Computer Int.*, 15:263-322, 2000.
53. H. Pan, Z.P. Liang, T.J. Anastasio, and T.S. Huang. “Exploiting the dependencies in information fusion,” *CVPR*, vol. 2:407–412, 1999.
54. M. Pantic and L.J.M. Rothkrantz, “Automatic analysis of facial expressions: The state of the art,” *IEEE Trans. on PAMI*, 22(12):1424–1445, 2000.
55. M. Pantic and L.J.M. Rothkrantz, “Toward an affect-sensitive multimodal human-computer interaction,” *Proceedings of the IEEE*, 91(9):1370–1390, 2003.
56. J. Paradiso and F. Sparacino, “Optical Tracking for Music and Dance Performance,” *Optical 3-D Measurement Techniques IV*, A. Gruen, H. Kahmen, eds., pp. 11-18, 1997.
57. V.I. Pavlovic, R. Sharma and T.S. Huang, “Visual interpretation of hand gestures for human-computer interaction: a review”, *IEEE Trans. on PAMI*, 19(7):677-695, 1997.
58. J.B. Pelz, “Portable eye-tracking in natural behavior,” *J. of Vision*, 4(11), 2004.
59. A. Pentland, “Looking at People,” *Comm. of the ACM,”* 43(3):35-44, 2000.
60. A. Pentland, “Socially Aware Computation and Communication,” IEEE Computer Vol 38,No. 3, March 2005.
61. R. W. Picard, *Affective Computing*, MIT Press, 1997.
62. M. Porta, “Vision-based user interfaces: methods and applications,” *Int. J. Human-Computer Studies*, 57(1):27-73, 2002.
63. L. M. Reeves et. al., “Guidelines for multimodal user interface design,” *Communications of the ACM*, Vol. 47, Issue 1, pp. 57-69, January 2004.
64. R. Rosales and S. Sclaroff, “Learning body pose via specialized maps,” *NIPS*, Vol. 14, pp 1263-1270, 2001.
65. P. Roth, and T. Pun, “Design and evaluation of a multimodal system for the non-visual exploration of digital pictures,” *INTERACT* 2003.
66. R. Ruddaraju, A. Haro, K. Nagel, Q. Tran, I. Essa, G. Abowd, E. Mynatt, “Perceptual user interfaces using vision-based eye tracking,” *ICMI,* 2003.
67. A. Santella and D. DeCarlo, “Robust clustering of eye movement recordings for quantification of visual interest,” *Eye Tracking Research and Applications (ETRA)*, pp. 27-34, 2004.
68. N. Sebe, I. Cohen, and T.S. Huang, Multimodal emotion recognition, *Handbook of Pattern Recognition and Computer Vision*, World Scientific, 2005.
69. E. Schapira and R. Sharma, “Experimental evaluation of vision and speech based multimodal interfaces,” W*orkshop on Perceptive User Interfaces*, pp. 1-9, 2001.
70. B. Schuller, M. Lang, G. Rigoll: “Multimodal emotion recognition in audiovisual communication,” *ICME,* 2002.
71. T. Selker, “Visual Attentive Interfaces,” *BT Technology Journal*, Vol. 22, No. 4, pp. 146-150, Oct. 2004.

72. R. Sharma, M. Yeasin, N. Krahnstoever, I. Rauschert, G. Cai, I. Brewer, A. MacEachren, and K. Sengupta, “Speech–gesture driven multimodal interfaces for crisis management,” *Proceedings of the IEEE*, 91(9):1327–1354, 2003.
73. L.E. Sibert and R.J.K. Jacob, “Evaluation of eye gaze interaction,” *Conf. Human-Factors in Computing Syst.*, pp. 281-288, 2000.
74. P. Smith, M. Shah, and N.d.V. Lobo, “Determining driver visual zttention with one camera,” *IEEE Trans. on Intelligent Transportation Systems*, 4(4), 2003.
75. R. Simpson, E. LoPresti, S. Hayashi, I. Nourbakhsh, D. Miller, “The smart wheelchair component system,” *J. of Rehabilitation Research and Development*, May/June 2004.
76. F. Sparacino, “The museum wearable: Real-time sensor-driven understanding of visitors' interests for personalized visually-augmented museum experiences,” *Museums and the Web*, 2002.
77. M. M. Trivedi, S. Y. Cheng, E. M. C. Childers, S. J. Krotosky, “Occupant posture analysis with stereo and thermal infrared video: Algorithms and experimental evaluation,” *IEEE Trans. on Vehicular Technology*, 53(6):1698-1712, 2004.
78. M. Turk, "Gesture recognition," *Handbook of Virtual Environment Technology*, K. Stanney (ed.), 2001.
79. M. Turk, "Computer vision in the interface," *Communications of the ACM*, Vol. 47, Issue 1, pp. 60-67, January 2004.
80. M. Turk and G. Robertson, “Perceptual Interfaces,” *Communications of the ACM*, Vol. 43, Issue 3, pp. 32-34, 2000.
81. M. Turk and M. Kölsch, "Perceptual Interfaces," G. Medioni and S.B. Kang (eds.), Emerging Topics in Computer Vision, Prentice Hall, 2004.
82. J.-G. Wang, E. Sung, and R. Venkateswarlu, “Eye gaze estimation from a single image of one eye,” *ICCV,* pp. 136-143, 2003.
83. L. Wang, W. Hu and T. Tan “Recent developments in human motion analysis,” *Patt. Recogn.*, 36 (2003) 585-601
84. J.J.L. Wang and S. Singh, “Video analysis of human dynamics – A survey,” *Real-Time Imaging,* 9(5):321-346, 2003.
85. K.C. Wassermann, K. Eng, P.F.M.J. Verschure, and J. Manzolli, “Live soundscape composition based on synthetic emotions,” *IEEE Multimedia Magazine*, 10(4), 2003.
86. Y. Wu and T. Huang. Vision-based gesture recognition: A review, *3rd Gesture Workshop*, 1999.
87. Y. Wu, G. Hua and T. Yu, “Tracking articulated body by dynamic Markov network,” *ICCV*, pp.1094-1101, 2003.
88. M.-H. Yang, D. Kriegman, and N. Ahuja, “Detecting faces in images: A survey,” *IEEE Trans. on PAMI*, 24(1):34–58, 2002.
89. Q. Yuan, S. Sclaroff and V. Athitsos, “Automatic 2D hand tracking in video sequences,” *IEEE Workshop on Applications of Computer Vision*, 2005.
90. C. Yu and D. H. Ballard, "A multimodal learning interface for grounding spoken language in sensorimotor experience", *ACM Trans. on Applied Perception*, 2004.
91. W. Zhao, R. Chellappa, A. Rosenfeld, and J. Phillips, “Face recognition: A literature survey,” *ACM Computing Surveys*, 12:399–458, 2003.
92. K. Salen and E. Zimmerman, *Rules of Play: Game Design Fundamentals*, MIT Press, 2003.
93. Z. Zeng, J. Tu, M. Liu, T. Zhang, N. Rizzolo, Z. Zhang, T.S. Huang, D. Roth, and S. Levinson, “Bimodal HCI-related affect recognition,” *ICMI,* 2004.
94. Y. Wu and T.S. Huang, “Human hand modeling, analysis and animation in the context of human computer interaction,” *IEEE Signal Processing,* 18(3):51-60, 2001.
95. I.R. Murray and J.L. Arnott, “Toward the simulation of emotion in synthetic speech: A review of the literature of human vocal emotion,” *J. of the Acoustic Society of America*, 93(2):1097–1108, 1993.

Tracking Body Parts of Multiple People for Multi-person Multimodal Interface

Sébastien Carbini, Jean-Emmanuel Viallet,
Olivier Bernier, and Bénédicte Bascle

France Télécom R&D,
2 av Pierre Marzin,
22307 Lannion Cedex, France
{sebastien.carbini,jeanemmanuel.viallet}@francetelecom.com
{olivier.bernier,benedicte.bascle}@francetelecom.com

Abstract. Although large displays could allow several users to work together and to move freely in a room, their associated interfaces are limited to contact devices that must generally be shared. This paper describes a novel interface called SHIVA (Several-Humans Interface with Vision and Audio) allowing several users to interact remotely with a very large display using both speech and gesture. The head and both hands of two users are tracked in real time by a stereo vision based system. From the body parts position, the direction pointed by each user is computed and selection gestures done with the second hand are recognized. Pointing gesture is fused with n-best results from speech recognition taking into account the application context. The system is tested on a chess game with two users playing on a very large display.

1 Introduction

Although large displays could theoretically allow several users to work together and to move freely in a room, their associated interfaces are limited to contact devices that must generally be shared. Moreover when interacting together, users use natural powerful ways of communication like voice and gestures, but when interacting with a computer, they are still limited to poor means of interaction like pushing buttons. This paper describes a novel interface called SHIVA (Several-Humans Interface with Vision and Audio) designed to allow several users to interact remotely with a very large display by using both speech and gesture. From a stereo camera, the head and both hands of two users are detected and tracked in real time. The first detected hand position of each user is used to estimate the pointed direction and the second one is used for selection or to control a third axis in 3D applications. Then, the pointed direction is fused with n-best results from speech recognition to interpret the user multimodal command, also taking into account the application context (here a chess game).

This paper is organized in the following manner. In the next section, some previous work on multi-person tracking and on pointing gesture recognition is discussed. Then our work novel contribution is presented. The following section

N. Sebe, M.S. Lew, and T.S. Huang (Eds.): HCI/ICCV 2005, LNCS 3766, pp. 16–25, 2005.

describes the body parts detection and tracking. The next section deals with the modalities fusion and synchronisation. Then, some results are given.

2 Previous Work

Concerning multiple person tracking, in [5], the authors present a particle filter based multi-person tracker with audio and video state components (position, height and whether each user is speaking). In [16], the authors track two persons in an outdoor context to identify their interactions. Tracking is based on blobs extraction and background substraction from monocular grayscale images. In [4], the authors use a multi-camera system based on Bayesian modality fusion to track multiple people in an indoor environment. The main common drawbacks of these methods is that each person is considered as one object and no information on their body parts positions are provided. In [15], the authors present a method to track body parts of multiple people with limited occlusions using a multiple hypothesis tracking with path coherence constraints. This interesting method requires a high frame rate to verify the assumption of smooth movements. Moreover, their method tracks skin-coloured blobs without knowing which one is the head and which one is the hand and without assigning each head and hand to a person. These methods seem to have an expensive computational cost which forbids their use in real time fusion with speech recognition and interaction with an application.

Since Bolt's "Put That There" [2], where a magnetic sensor was used to retrieve the pointed direction, some recent studies in computer vision deals with pointing gesture recognition in front of a large display. In [6], the body pose of a user is tracked thanks to a 3D articulated model and a stereo camera. The system works with cluttered background but need an initial calibration step (user with extended arms). Moreover, as in [9], to select an object, a user must keep pointing for a certain amount of time (2 sec) which slows the interaction, tires the user and can lead to unwanted selections. In [10], the head and a hand of a user are tracked and used to compute the pointed direction. The algorithm compute extremal points of the body silhouettes extracted from multiple cameras. A computer per camera is used and an off-line calibration step between cameras is needed. Moreover, the hand and the head are not always the extremal points of the silhouettes and other body parts like the elbow can be mistakenly labelled as one of the tracked part. In [13], based on a stereo camera, the hands and head of a user are detected and tracked as skin-colour blobs. As a result, the system would probably fail with skin-coloured clothes. Moreover, the precision on the head-hand line estimation is too low to be used continuously in a large display interface. In [12], head and both hands of two users are tracked simultaneously with a monocular camera for each user. The use of a monocular camera makes it difficult to compute the 3D head-hand axis or the 3D arm axis and thus a cursor is moved according to the 2D position of the hand only. It could be a problem specially for small people to point the top of a very large screen. Moreover, in 2D it is harder to detect the purposiveness of the user. In [17], the intentional arm-pointing gestures of several users are recognized using surrounding stereo cameras. The pointing precision is good and is robust to lighting change and users' clothing but interaction is limited by using only a one arm horizontal and vertical swinging gesture interface in addition to pointing gestures.

Compared to other studies, SHIVA is able to detect and track the head and both hands of two users as in [15] but each body part is assigned to a person. The interface is based on our previous work for tracking the body parts of one person described in [3]. As in [6, 9, 13], a stereo camera is used with a 320x240 image resolution. The tracker is robust to reasonable variations of luminosity, to skin-coloured clothes and to cluttered backgrounds. Moreover, body parts detection, tracking and losses detection are fully automatic and perform in real time.

No calibration, specific user prior or learning is needed. As long as no occlusions between body parts occur, the system keeps the same behaviour when users move freely in the room as in [17]. The head-hand axis is used as pointing convention as in [9, 10, 13]. Selection is done with the second hand or thanks to speech recognition. In addition, speech recognition has a small vocabulary adapted to the application to allow more powerful commands rather than just a gesture mouse [3, 6, 9]. The speech recognition results are fused with gesture and the application context to allow shorter multimodal commands and to create a flexible interface.

3 Body Parts Detection and Tracking

3.1 Preprocessing

Detection and tracking of body parts rely on skin-colour (figure 1-2), disparity (figure 1-3) and movement (figure 1-4) information. Skin colour is obtained by a broad filter look-up table obtained on different users and under different lighting conditions. Obtained from the two images of a stereo camera, the disparity gives the 3D position of a pixel with respect to the camera coordinates. Disparity cannot be precisely determined in homogeneous colour zones or in areas with large depth difference. A filtered disparity image is computed for each user for which the observations too far away ($> 1.3m$) from the head centre are discarded (once the head has been detected). The movement is obtained by background subtraction which is updated every image so that a still person quickly fades in the background.

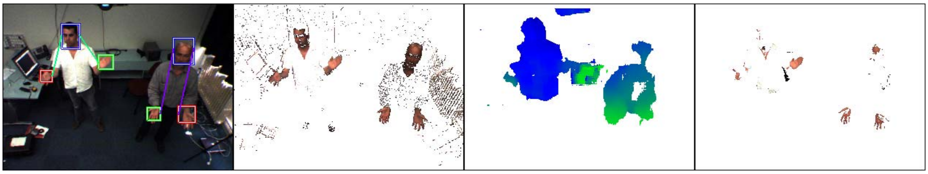


Fig. 1. From left to right: (1) RGB image. (2) Skin-colour image. (3) Mixed disparity image of the two users. (4) Movement image

3.2 Head Detection

When no user is present, the system searchs for a face. As no face recognition techniques are used, the first detected head is labelled as user A head and second detected head as user B head. Then when a head lost is automatically detected (c.f. §3.5), its detection is retriggered and the remaining head is still tracked. An accurate face detector is used to initialize the tracking: it is a modular neural network with high detection rate and a very low false alarm rate. It has been described in details in [8]. For human computer interfaces (HCI), failure to detect face in an image is not crucial since the face can be detected in the following images. On the other hand, continuously tracking a false alarm is a major problem. The face detector is accurate but slow. To speed up the process, the following procedure is used. The image is equally divided into sixteen regions. At each image, candidate regions for face detection are selected. These are regions with a sufficient number of moving pixels, skin colour pixels and depth pixels within the face detector detection range. The regions containing the other user body parts, if known, are removed from candidate regions list. Only one of these candidate regions is then randomly selected at each image. The face detector is only applied on this region giving real time performance. Each pixel of this region is tested as possible centre for the face by the neural network. Pixels of other regions are taken into account when testing near boundaries pixels so that the detection works even when face is located at a boundary. A selection algorithm ensures that a region is not selected again before all other candidate regions are first tested. Once a face is found, the 3D tracking is initialized on the image corresponding to the detected face.

3.3 Body Space

Once a head is detected, its 3D position is used to define several areas. These areas help to detect hands and to detect the pointing purposiveness of the user. Biometric constraints limit the search space during hand detection to a spheroid centred on the face. The area outside this spheroid is discarded (figure 2-area 3). Furthermore, it is reasonable to admit that, when interacting with the display, the user moves its dominant hand towards the display and sufficiently away from its face (>30 cm). Thus the hand search space is restricted to a volume delimited by a sphere and a plane, a volume called the 'action area' (figure 2-area 2). Each user has a private area (figure 2-area a and area c). If the distance between the

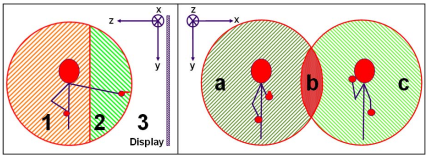


Fig. 2. Left: side view 1: rest area, 2: action area, 3: discarded area. Right: front view a: user A private area, b: overlapped area, c: user B private area

two users is too small the private areas overlap. The 'overlapped area' (figure 2-area b) is excluded from the hand search space to avoid assigning a hand to a wrong user. For both users, the first detected hand will be labelled as pointing hand and the second one as controlling hand (used for selection gesture or to control a third axis in 3D environments). The system works both for right-handed or left-handed users without the need for left or right hand labels. We assume that users will first point at an object with their predominant hand before using the other hand to interact with it. Once detected a hand is continuously tracked in the 'action area' and 'rest area' (figure 2-area 1) and in the other user private area. But the hand function (pointing or controlling) will be taken into account only when the hand is in the 'action area' to detect the user purposiveness. These areas are face referenced, so that their absolute positions change as users move but the system behaviour remains the same.

3.4 Hands Detection

Contrary to faces, hands exhibit extremely variable shapes as seen from a camera and are thus difficult to detect, specifically at low image resolution. Once the face is detected and using disparity information, the 3D position of the face is obtained and the body search areas are defined (c.f. §3.3). A hand is then detected as a skin colour moving zone, in the action area, the closest one to the display. When present, the pixels of the other hand and the corresponding arm are subtracted from the search space as described in [3]. Indeed, a naked arm or an arm covered by a skin colour cloth could be mistakenly selected as the second skin colour moving zone closest to the display. In a similar way, tracked body parts of the other user are discarded.

3.5 Tracking

Once detected body parts are first tracked for user A and then for user B. The core of the tracking process is the statistical model, which aims at explaining the result of the pre-processing stages. Each model represent a body part and is a combination of a colour histogram and a 3D spatial Gaussian function. Moreover an exponential function gives higher probability to forward observations for the hands (to be robust to skin-coloured arms). A model with a colour histogram only is used for the discarded class. Only skin-colour observations belonging either to the action area or to the rest area are taken into account and the observations belonging to the other user body parts are ignored.

The parameters of all models are adapted for each image with an EM algorithm described more in details for one person in [3]. This algorithm alternates an expectation E step with a maximization M step and converges towards the parameters to the a posteriori (since a prior is used) maximum probability. At each E step, the parameters are initialized with the values computed at the previous M step or, for the first iteration, with the results obtained for the previous image. During the expectation step, the probability of belonging to a model (the hidden variable of the EM algorithm) is computed for every observation and for each of the models. During the Maximisation step, new estimations of the parameters are computed from the expectations.

It is a necessity to automatically detect tracking losses, so that the system is able to recover from tracking drifts or if a user steps out of the field of view of the camera. A measure of the number of skin colour pixels and of valid disparity pixels is performed. If this measure is below some given threshold, the model is considered as lost and the corresponding body part detection process is triggered. In addition, hand models are considered to be lost if the distance between the head and the hand exceeds some value ($\approx 1,3m$).

4 SHIVA Multimodal System

The aim of SHIVA interface is to enable both users to talk and point simultaneously. Currently, SHIVA has been implemented on a chess game (figure 5-E and 5-F) where users interact with speech and gesture one at a time.

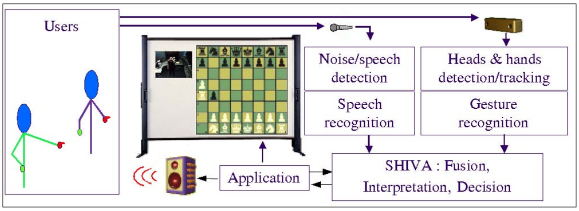


Fig. 3. The different speech and gesture components of SHIVA

A HCI is a loop where the user should not be only consider as input but also as an output of the system (figure 3). Visual and audio feedbacks are provided to users. The tracking results and speech recognition are displayed and animations allow to visualise a chess move even if it is forbidden by chess rules.

4.1 Synchronisation and Fusion Techniques

One of the first steps towards interpreting a multimodal command is synchronising speech and gesture [7]. With a pen-voice interface, S. Oviatt et al. [14] showed that for 32% of the patterns, the pen gestures were completed first and were followed by the voice utterance with a lag of 1.4 sec. For a gesture interface similar to SHIVA, S. Kettebekov and R. Sharma [11] observe that 93.7% of the gestures were temporally aligned with the semantically associated speech utterance during a map manipulation application. We assume that gesture and speech are synchronous.

In [7], the authors address the problem of speech-gesture alignment by choosing the appropriate gesture to link each verbal utterance. But in our system, the pointing gesture modality is a permanent process, while the hand stays sufficiently in front of the head, and all pointed locations during interaction are known. Selection gesture is a discrete event whereas oral selection utterance is an event, whose beginning and end are known (figure 4): to detect end of speech, some consecutive silent frames must be observed. So the n-best results of speech recognition are delivered 240 ms after the end of speech signal. To synchronize pointing and selection, we consider the pointing location at the instant corresponding to the selection gesture event [1] and the mean pointing location corresponding to the speech signal time interval (figure 5-Left frame). Direct fusion between modalities occurs by associating the best speech recognition result coherent with the object found at the pointed location.

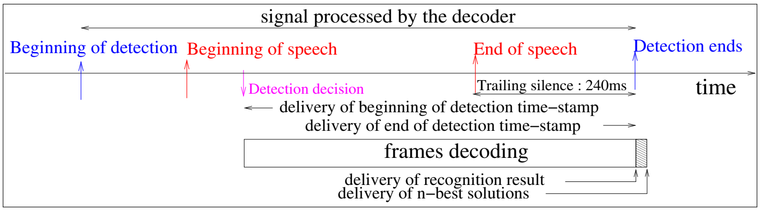

signal processed by the decoder
Beginning of detection
Beginning of speech
End of speech
Detection ends
Detection decision
Trailing silence : 240ms
time
delivery of beginning of detection time−stamp
delivery of end of detection time−stamp
frames decoding
delivery of recognition result
delivery of n−best solutions

Fig. 4. Different times related to speech recognition. Speech recognition result is only available after a trailling silence delay following speech signal

4.2 Application Context Awareness

A chess game [18] has been modified to accept input commands generated by the multimodal interpreter and to provide contextual constraint information outputs (figure 3). The chess application context includes displacement rules, current position of the pieces, displacement history and player turn. From this context, a list of legal or illegal, of ambiguous or explicit moves is obtained. The aim is to match one of these moves with a multimodal command.

Two specified squares suffice to move a piece: the square where the piece to move is and the destination square. Pointing at a square, a user specifies one of these two required squares. The information about the second one can be specified by speech. To make shorter utterance, the user will lean on information already available in the context and specify only the missing information. So to find the second square, SHIVA must retrieve information from speech and from the application context. The current context is linked to the pointed square:

- When a user points at one of his or her piece, the current context is the opposing pieces that can be taken by the pointed piece (figure 5-D).
- When a user points at one of the opposing piece or at an empty square, the current context is the user pieces that can take the pointed opposing piece or move to the empty square.

When a user utters the name of a piece which is unique in the current context the second square is known and the move activated. If several pieces of the current context have the same uttered name (pawns in figure 5-D), the move is ambiguous and a feedback is sent. If the uttered piece does not belong to the

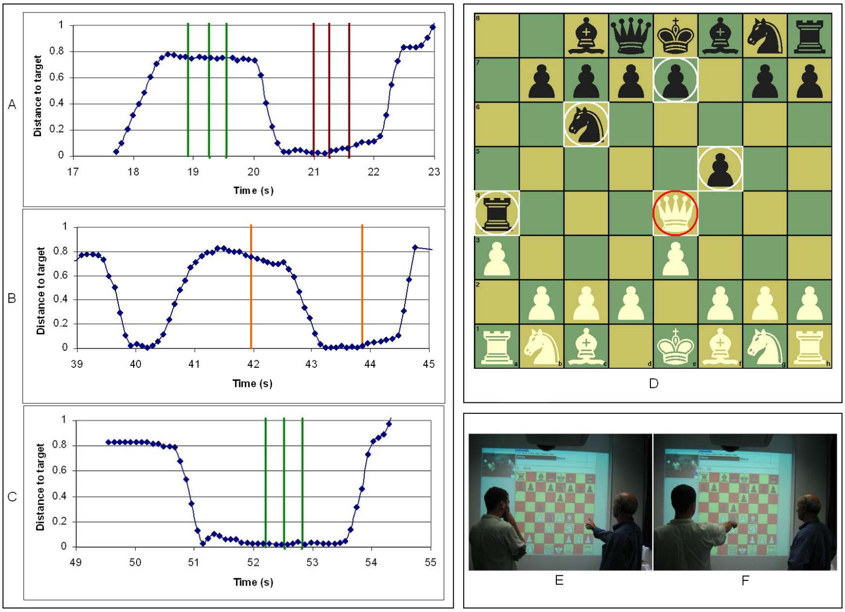


Fig. 5. *Left frame:* Three alternative ways to move a chess piece. (A) Pointing gestures (curve) to destination and drag/drop Oral commands (group of 3 vertical lines indicates begin, end of speech signal and speech recognition time stamps). (B) Pointing gestures (curve) and select/unselect gestures (vertical lines indicate gesture time event). (C) Multimodal pointing gestures (curve) and one oral selection command (group of 3 vertical lines indicates begin, end of speech signal and speech recognition time stamps). For commodity, pointing gestures are described by a distance from the current pointing position to an arbitrary target reference in the chessboard. *Top-Right frame:* (D) When white player point the white queen (red circle), the corresponding context (white circles) includes the pieces that can be taken by the queen: a rook, a knight and two pawns. *Bottom-Left frame:* (E) Right user moving a piece while left user waits. (F) Left user moving a piece while right user waits

current application context (a bishop in figure 5-D), the move is forbidden, and a feedback shows the illegal nature. When a user is not pointing, all the information to find the two required squares must be found in speech and context. In this case, the current context is all the possible takes. To play without pointing, a user utters the name of two non ambiguous pieces with a 'take' verb. For example, in figure 5-D, the utterance "queen takes rook" works even without pointing, since only one rook can be taken by the queen. But "queen takes pawn" is ambiguous as two pawns can be taken by the queen.

5 Results

The tracker performs at about 11 Hz when one user is tracked and 9 Hz with two users on a Pentium IV (3 Ghz). This is acceptable for such applications as chess game. Thanks to the tracking method used, there is few computational cost differences between tracking one and two persons. This is a promising result for tracking more than two persons although it has not been implemented yet.

Concerning the precision, some previous results obtained with a one person interface and from experiments on 14 users showed that the pointed direction is estimated with a precision of about 0.75 deg [3] (1.5 % of the diagonal of the very large display (120") for a user at 1.5 m of the large display). The precision is the same when two persons interact. The system precision has been improved since these experiments but experimental values have not been updated yet. The tracker is robust to cluttered backgrounds (figure 6), and to skin-coloured arms (figure 6-right). Users can sit or stand (figure 6-left) and move in the field of the camera. Once detected, hands are tracked even when entering the other user private area (figure 6-middle and 6-right).

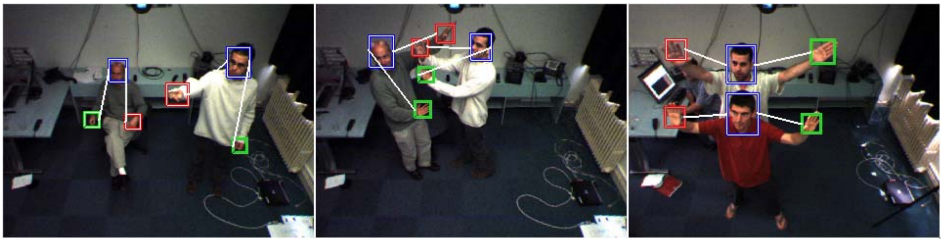


Fig. 6. Some tracking results exemples. Rectangles: tracked body parts (blue: head, red: 1st hand, green: 2nd hand). White lines: assignment of hands to a head. From left to right: (1) One left-handed user sitting in the back, the other standing and pointing. (2) Even when private areas overlap, body parts are still tracked and assigned correctly. (3) Shiva posture, the tracker is robust to naked forearms and to freely moving users

6 Conclusion

A novel method has been presented to track body parts of two persons in quasi real time. Some previous results showed that the pointed direction is estimated with a good precision. This precision and the low computational cost of our method allow its use in real time fusion with speech recognition. The use of the application context during modalities fusion allow shorter multimodal commands. As a result, SHIVA is a flexible interface where one or two users can interact with gesture alone, speech alone or using speech-gesture multimodal commands. The next step will be to test SHIVA with an application requiring simultaneous cooperative interaction of both users.

References

1. Y. Bellik: Technical Requirements for a Successful Multimodal Interaction, International Workshop on Information Presentation and Natural Multimodal Dialogue, Verona, Italy, 2001.

2. R. A. Bolt: “Put-that-there”: Voice and gesture at the graphics interface, Proceedings of the 7th Annual Conference on Computer Graphics and Interactive Techniques, p. 262-270, Seattle, Washington, 1980.
3. S. Carbini, J.E. Viallet and O. Bernier: Pointing Gesture Visual Recognition for Large Display, Pointing’04 ICPR Workshop, Cambridge, U. K., 2004.
4. T.H. Chang and S. Gong: Tracking multiple people with a multi-camera system, IEEE Workshop on Multi-Object Tracking, p. 19-26, Vancouver, Canada, 2001.
5. N. Checka, K. Wilson, M. Siracusa, T. Darrell: Multiple Person and Speaker Activity Tracking with a Particle Filter, ICASSP, Montreal, Canada, 2004.
6. D. Demirdjian and T. Darrell: 3-D Articulated Pose Tracking for Untethered Diectic Reference, Proceedings of International Conference on Multimodal Interfaces, p. 267, Pittsburgh, Pennsylvania, 2002.
7. J. Eisenstein and C. M. Christoudias: A Salience-Based Approach to Gesture-Speech Alignement, HLT-NAACL, p. 25-32, Boston, Massachusetts, 2004.
8. R. Feraud, O. Bernier, J.E. Viallet and M. Collobert: A fast and accurate face detector based on neural networks, PAMI, Vol. 23, n. 1, p. 42-53, 2001.
9. N. Jojic and B. Brumitt and B. Meyers and S. Harris: Detecting and Estimating Pointing Gestures in Dense Disparity Maps, IEEE International Conference on Face and Gesture recognition, p. 468, Grenoble, France, 2000.
10. R. Kehl and L. Van Gool: Real-time Pointing Gesture Recognition for an Immersive Environment, IEEE International Conference on Automatic Face and Gesture Recognition, p. 577-582, Seoul, Korea, 2004.
11. S. Kettebekov and R. Sharma, Understanding Gestures in a Multimodal Human Computer Interaction, International Journal of Artificial Intelligence Tools, vol. 9, n. 2, p. 205-223, 2000.
12. N. Krahnstoever, S. Kettebekov, M. Yeasin and R. Sharma: A Real-Time Framework for Natural Multimodal Interaction with Large Screen Displays, International Conference on Multimodal Interfaces, p. 349, Pittsburgh, Pennsylvania, 2002.
13. K. Nickel, E. Seemann and R. Stiefelhagen: 3D-Tracking of Head and Hands for Pointing Gesture Recognition in a Human-Robot Interaction Scenario, IEEE International Conference on Automatic Face and Gesture Recognition, p. 565, Seoul, Korea, 2004.
14. S. Oviatt, A. De Angeli and K. Kuhn: Integration and synchronization of input modes during multimodal human-computer interaction, CHI ’97: Proceedings of the SIGCHI Conference on Human factors in computing systems, p. 415-422, Atlanta, Georgia, 1997.
15. E. Polat, M. Yeasin and R. Sharma: A Tracking Framework for Collaborative Human Computer Interaction, International Conference on Multimodal Interfaces, p. 27-32, Pittsburgh, Pennsylvania, 2002.
16. K. Sato and J.K. Aggarwal: Tracking and recognizing two-person interactions in outdoor image sequences, Workshop on Multi-Object Tracking, p. 87-94, Vancouver, Canada, 2001.
17. Y. Yamamoto, I. Yoda and K. Sakaue, Arm-Pointing Gesture Interface Using Surrounded Stereo Cameras System, ICPR (International Conference on Pattern Recognition), p. 965-970, Cambridge, UK , 2004.
18. Xboard: http://www.tim-mann.org/xboard.html.

Articulated Body Tracking Using Dynamic Belief Propagation

Tony X. Han and Thomas S. Huang

Beckman Institute and ECE Department
University of Illinois at Urbana-Champaign
405 N. Mathews Ave., Urbana, IL 61801, USA
{xuhan,huang}@ifp.uiuc.edu
http://www.ifp.uiuc.edu/~xuhan

Abstract. An efficient articulated body tracking algorithm is proposed in this paper. Due to the high dimensionality of human-body motion, current articulated tracking algorithms based on sampling [1], belief propagation (BP) [2], or non-parametric belief propagation (NBP) [3], are very slow. To accelerate the articulated tracking algorithm, we adapted belief propagation according to the dynamics of articulated human motion. The searching space is selected according to the prediction based on human motion dynamics and current body-configuration estimation. The searching space of the dynamic BP tracker is much smaller than the one of traditional BP tracker [2] and the dynamic BP need not the slow Gibbs sampler used in NBP [3–5]. Based on a graphical model similar to the pictorial structure [6] or loose-limbed model [3], the proposed efficient, dynamic BP is carried out to find the MAP of the body configuration. The experiments on tracking the body movement in meeting scenario show robustness and efficiency of the proposed algorithm.

1 Introduction

Accurate, robust, and efficient visual tracking of articulated objects such as human body/hand, has wide applications in human-computer interaction, motion capture, video surveillance,augmented reality, annotation, and activity recognition [4, 7]. However, the articulated human body tracking is inherently a very difficult problem due to: 1) high degree (usually 20–68) of freedom of the articulated body movement [2, 6, 8, 9]; 2) large appearance change of body parts during the movement; 3) occlusion between body parts; 4) no typical appearance due to clothing; 5) fast movement of human arms and legs; 6) the posterior distribution of body configuration is multimodal and spiky.

Many enlightening articulated body tracking algorithms appeared in recent years. Bregler and Malik used exponential map to model the articulated twist. After model the kinematic chain as the product of exponentials, a Newton-Raphson style minimization is carried out to find the minimizer (the body configuration) of the cost function [9]. The introduction of the exponential map is a neat idea, but the Newton-Raphson is inherently a variant of the gradient descent method. Therefore, the optimization procedure is likely to be trapped by

N. Sebe, M.S. Lew, and T.S. Huang (Eds.): HCI/ICCV 2005, LNCS 3766, pp. 26–35, 2005.

local minima. We have mentioned above that the posterior distribution of body configuration is multimodal and spiky. In other words, there are many local minima in the objective function which usually make the local minimizer found by the tracker different from the global minimizer, i.e. the optimal body configuration given the current observation. The fact that body parts usually move very fast compared with the common frame rate, further validates the claim. Annealing the particle filter may be one way to tackle this difficulty [1]. However, the sampling based algorithms are usually very slow when large amount of particles are required. The high dimensionality of articulated body motion requires large number of particles even if the annealed particle filter is applied. According to [1], the tracker using 10 annealing layers with 200 particles need around 1 hour to process 5 seconds of footage.

Ramanan and Forsyth proposed a 2d tracker with automatic initialization, which can track long video sequence [2, 10]. The continuous body configuration space is discretized first and a variant of Belief Propagation (BP)[11–13], max product, is then carried out on the loopy graphical model to find the approximate estimate of the MAP, i.e. the suboptimal body configuration given current video input. BP is a very powerful algorithm, which reduces the computational complexity of brutal force search from $O(N^m)$ to $O(N^2m)$ for non-loopy graph. Here N is the size of the domain of each node (the possible discrete values the node can take) and m is the number of nodes in the graphical model. But usually N, i.e. the domain size of each node, is very large in the tracking context. For articulated 2d tracker, the configuration of each body part is a triplet $(x, y, \theta)^T$, representing the horizontal translation, vertical translation, and in-plane rotation respectively. Suppose each axis is discretized into 20 bins, the searching space for each node is 20^3. As we have pointed out,the computation complexity of the 2d tracker is $O(20^6 \cdot 9)$. Therefore, the articulated tracker based on the BP in the discretized space requires huge computational power or lots of cpu time. The complexity would be even formidable if we want to realize a 3d articulated tracker using the same approach.

Sigal et al [3] devised a 3D articulated body tracker and Sudderth et al [4] came up with a articulated hand tracker both based on nonparametric BP (NBP) [5] algorithm. In the framework of NBP, the large searching space is represented as mixture of Gaussians. For each iteration of NBP, the parameters of the mixture of Gaussians are recomputed using Gibbs sampling. The computational complexity is reduced from $O(N^2m)$ to $O(M \cdot d \cdot m)$, where $d \ll N$ is the components number of the mixture of Gaussians and M is the number of samples. However, to ensure good approximation, the Gibbs sampler require large number of particles, which make their articulated hand tracker inevitably slow. According to [4], with 200 particles, their matlab implementation requires about one minute for each NBP iterations.

To reduce the huge discretized searching space of BP while avoid the slow Gibbs sampler used in NBP, we propose the dynamic BP framework. The basic idea is, that the temporal constraints or human motion dynamics should **NOT** be used to build the compatibility functions between the nodes of different frames as what is done in [2, 4]. Instead, the temporal constraints should be used to limit the searching space of the nodes in different frames. Since the body motion is continuous, the values that certain node can take at frame t should lie in the neighborhood of the configuration of the corresponding node at frame $t-1$. The size of neighborhood between corresponding nodes at different frames and the neighborhood prior are determined by body motion dynamics. Please refer to section 3 for detail. Our implementation of a 2d articulated body tracker based on the dynamic BP framework run at around 2 frame/second for video with resolution of 1024×768, which is around 100 times faster then our implementation of the tracker based on BP. Also our experiments show the accuracy and the robustness of our tracker.

This paper is organized as follows. The articulated body tracking model with elastic constraints is introduced in section 2. BP on the whole discretized searching space is also reviewed in this section. Section 3 discusses how to use the human motion dynamics to limit the searching space of BP to achieve dynamic BP. Experiment results and the discussion are given in section 4. And section 5 concludes the paper.

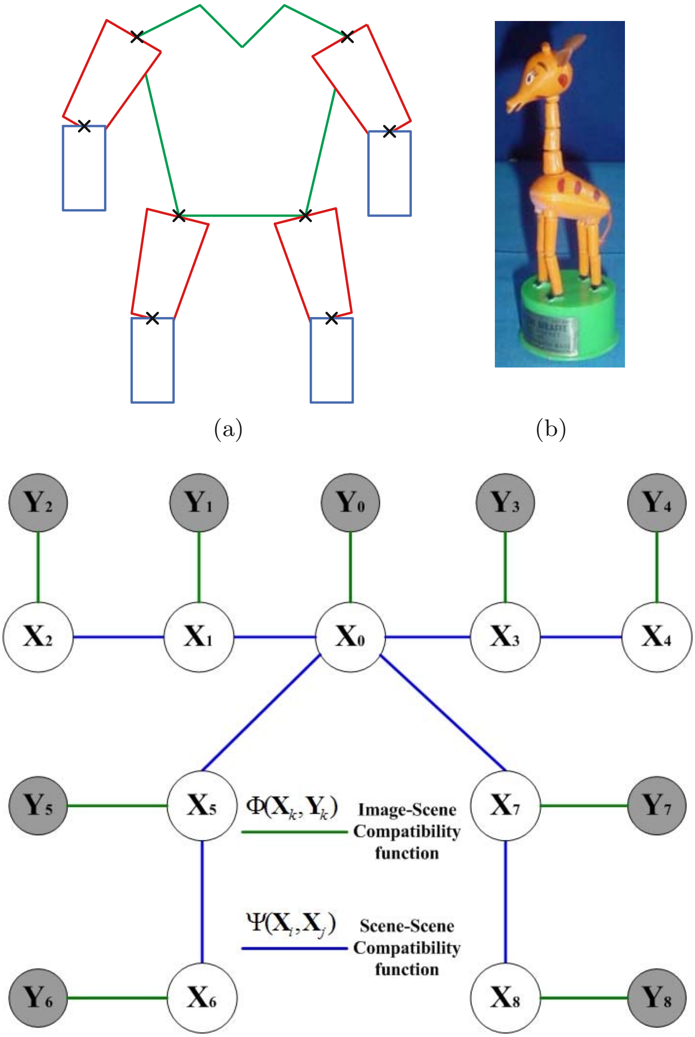
2 Articulated Body Model and Tracking Scheme

2.1 Graphical Body Model with Elastic Constraints

We model the 2D view of the human body as a connected card board model shown in figure 1(a). In our body tracking framework, we use a “loose-limbed” body model [6, 8] in which the limbs are not rigidly connected but are rather “attracted” to each other. Instead of representing the body as a single 33-dimensional kinematic tree, each limb is treated quasi-independently with soft constraints between the position and orientation of adjacent parts. The model resembles a Push Puppet toy which has elastic connections between the limbs as shown in figure 1(b). For each body part, i.e. the node in the graph shown in figure 1(c), the configuration space for each node is a 3 dimensional vector space (x, y, θ), representing the horizontal translation, vertical translation, and in-plane rotation respectively. The elastic constraints between body parts are modeled as the Scene-Scene compatibility function $\Psi(\mathbf{X}_i, \mathbf{X}_j)$. The closer the two nodes in spacial distance, the bigger the scene-scene compatibility function. The Image-Scene compatibility function $\Phi(\mathbf{X}_k, \mathbf{Y}_k)$ model the similarity between the tracked parts and the parts' template.

In the probabilistic framework, the compatibility functions should be proportional to the corresponding conditional distributions or join distributions, i.e. $\Phi(\mathbf{X}_k, \mathbf{Y}_k) \propto P(\mathbf{Y}_k|\mathbf{X}_k)$; $\Psi(\mathbf{X}_i, \mathbf{X}_j) \propto P(\mathbf{X}_i, \mathbf{X}_j)$. Since the graphical model in figure 1(c) is a Markov Random Field (MRF), the joint probability over the scenes $\mathbf{X}_i$'s and observations $\mathbf{Y}_k$'s can be written as [13]:

$$P(\mathbf{X}_0, \mathbf{X}_1, \ldots, \mathbf{X}_{m-1}, \mathbf{Y}_0, \mathbf{Y}_1, \ldots, \mathbf{Y}_{m-1}) = \prod_{(i,j)} \Psi(\mathbf{X}_i, \mathbf{X}_j) \prod_k \Phi(\mathbf{X}_k, \mathbf{Y}_k) \ . \quad (1)$$


(a)

(b)

(c)

Y₂
Y₁
Y₀
Y₃
Y₄
X₂
X₁
X₀
X₃
X₄
Y₅
X₅
Φ(X_k, Y_k)
Image-Scene Compatibility function
X₇
Y₇
Ψ(X_i, X_j)
Scene-Scene Compatibility function
Y₆
X₆
X₈
Y₈

Fig. 1. The model setup for the 2D articulated body tracker. (a) The card board model of human body with elastic joints. (b) Toy Push Puppet with elastic joints. (c) The loose attractive graphical body model, on which the belief propagation is carried on. $\mathbf{X}_0$: Torso configuration; $\mathbf{X}_1$: Right upper arm configuration; $\mathbf{X}_2$: Right forearm configuration; $\mathbf{X}_3$: Left upper arm configuration; $\mathbf{X}_4$: Left forearm configuration; $\mathbf{X}_5$: Right thigh configuration; $\mathbf{X}_6$: Right crus configuration; $\mathbf{X}_7$: Left thigh configuration; $\mathbf{X}_8$: Left crus configuration; $\mathbf{Y}_i$ is the observation of $\mathbf{X}_i$. The random variable $\mathbf{X}_i$'s only directly depend on their neighbors

Thus given current image observation, the MAP estimation of the configuration of the jth body part $\hat{\mathbf{X}}_{j\mathrm{MAP}}$ is:

$$\begin{aligned}\hat{\mathbf{X}}_{j\mathrm{MAP}} &= \arg\max_{\mathbf{X}_j} \max_{[\mathrm{all}\mathbf{X}_i, i \neq j]} P(\mathbf{X}_0, \mathbf{X}_1, \ldots, \mathbf{X}_{m-1}, \mathbf{Y}_0, \mathbf{Y}_1, \ldots, \mathbf{Y}_{m-1}) \\ &= \arg\max_{\mathbf{X}_j} \max_{[\mathrm{all}\mathbf{X}_i, i \neq j]} \prod_{(i,j)} \Psi(\mathbf{X}_i, \mathbf{X}_j) \prod_k \Phi(\mathbf{X}_k, \mathbf{Y}_k) \ . \end{aligned} \tag{2}$$

For a non-loopy graph (like the graph in figure 1(c)), Eqs. 2 can be computed by iterating the following steps [11] until the solution converges.

$$\hat{\mathbf{X}}_{j\mathrm{MAP}} = \arg\max_{\mathbf{X}_j} \Phi(\mathbf{X}_j, \mathbf{Y}_j) \prod_k M_j^{[k]} \ , \tag{3}$$

where k runs over all scene neighbors of node $\mathbf{X}_j$. $M_j^{[k]}$ is the message from node $\mathbf{X}_k$ to node $\mathbf{X}_j$. $M_j^{[k]}$ is updated according to:

$$M_j^{[k]} = \max_{[\mathbf{X}_k]} \Psi(\mathbf{X}_j, \mathbf{X}_k) \Phi(\mathbf{X}_k, \mathbf{Y}_k) \prod_{l \neq j} \tilde{M}_k^{[l]} \ , \tag{4}$$

where $\tilde{M}_k^{[l]}$ is $M_k^{[l]}$ from the previous iteration. The initial $M_k^{[l]}$'s are set to uniformly 1 in the discretized searching space of the variable $\mathbf{X}_i$'s.

By using BP described above, we can efficiently find the MAP estimation of the high-dimensional body configuration given the current image observation. And the computational complexity is reduced from $O(N^m)$ (when brutal force search is applied) to $O(N^2 m)$ (using BP).

2.2 Substantiating the Compatibility Functions

In our implementation of the tracker, the template of each body part is manually labeled at the first frame of the video, or, at a typical frame where no self-occlusion exists. The template for each body part is actually an image patch encompassed by a polygon as shown in figure 2. Denote the template for jth body part as T_j and the warped version of the template $T_j \langle \mathbf{X}_j \rangle$ according to the configuration parameter $X_j = (x, y, \theta)^T$. Denote the tth frame of the tracking video as I_t and the observed image patch for jth body part with configuration $\mathbf{X}_j$ is therefore $I_j \langle \mathbf{X}_j \rangle$. The image-scene compatibility function $\Phi(\mathbf{X}_k, \mathbf{Y}_k)$, which is proportional to the conditional probability $P(\mathbf{Y}_k | \mathbf{X}_k)$ is therefore defined as

$$\begin{aligned}\Phi(\mathbf{X}_k, \mathbf{Y}_k) &\propto P(\mathbf{Y}_k | \mathbf{X}_k) \\ &= C_k \exp\left(SSD_{rgb}\left(I_j \langle \mathbf{X}_j \rangle, T_k \langle \mathbf{X}_k \rangle\right)\right) \ , \end{aligned} \tag{5}$$

where $SSD_{rgb}(\cdot, \cdot)$ is the Sum of Squared Difference (SSD) between two image patch in RGB color space and C_k the normalization constant.

The elastic constraints between body parts, i.e. the scene-scene compatibility function $\Psi(\mathbf{X}_i, \mathbf{X}_j)$ is characterized by the Euclidean distance between the joint points of the adjacent body parts. For two adjacent body parts i and j, denote the soft joint on ith part, which is elastically connected to the jth part, as $\mathbf{J}_{i \to j}$. The corresponding soft joint on ith part is $\mathbf{J}_{j \to i}$. The 2D location of $\mathbf{J}_{i \to j}$

with ith body part's configuration $\mathbf{X}_i$, is $\mathbf{J}_{i \to j}\langle \mathbf{X}_i \rangle$. Therefore the scene-scene compatibility function is defined as:

$$\Psi(\mathbf{X}_i, \mathbf{X}_j) = \exp\left(K \left\| \mathbf{J}_{i \to j}\langle \mathbf{X}_i \rangle - \mathbf{J}_{j \to i}\langle \mathbf{X}_j \rangle \right\|_2\right) \ , \tag{6}$$

where K is the spring constant depending on how "elastic" we want the body card model to be.

With the compatibility functions define above, BP can be carried out on the body graphical model as discussed in section 2.1.

3 Using the Motion Dynamics to Limit the BP Searching Space

The belief propagation has reduced the computation complexity from $O(N^m)$ to $O(N^2 m)$, however, for articulated body tracking, N, i.e. the possible discretized state for each node, is still very large. In the graphical model for 2d body tracking shown in figure 1, each node represents all possible 2d configurations of the body part in the loosely connected model. For each body part, the possible configuration is 3-dimension vector $(x, y, \theta)^T$. Suppose we only have 20 discrete state for each dimension, thus $N = 20^3$ and $m = 9$ (9 body parts). Even if BP is applied, the total searching complexity is still $O\left((20^3)^2 \cdot 9\right)$. This makes the BP based tracker either impractical or extremely slow .

Previous work [2, 4] use the temporal constraints or human motion dynamics to build the compatibility functions between the nodes at different frames. This kind of approach can improve the robustness of the tracker at the cost of more computation because more compatibility functions are added into the graphical model.

To reduce the computational complexity, the temporal constraints is used to limit the searching space of the nodes in different frames. Since the body motion is continuous, the value that certain node can take at frame t should lies in the neighborhood of the configuration of the corresponding node at frame $t-1$. The size of neighborhood between corresponding nodes at different frames and the neighborhood prior are determined by body motion dynamics. Therefore Eqs. 2 should by adapted as:

$$\begin{aligned} &\check{\mathbf{X}}_{j\mathrm{MAP}}^{(t)} \\ &= \underset{\mathbf{x}_j^{(t)} \in \mathcal{N}\left(\mathbf{X}_j^{(t-1)}, \sigma\right)}{\arg\max} \quad \max_{\substack{\mathbf{x}_i^{(t)} \in \mathcal{N}\left(\mathbf{x}_i^{(t-1)}, \sigma\right) \\ i \not\dashv j}} \prod_{(i,j)} \Psi(\mathbf{X}_i^{(t)}, \mathbf{X}_j^{(t)}) \prod_k \Phi(\mathbf{X}_k^{(t)}, \mathbf{Y}_k^{(t)}) \ , \end{aligned} \tag{7}$$

where $\mathcal{N}\left(\mathbf{X}_i^{(t-1)}, \sigma\right)$ is the neighborhood the ith body part's configuration at frame $t-1$, denoted as $\mathbf{X}_i^{(t-1)}$. Here σ is the scale parameter we choose to adjust the neighborhood size.

Then BP is carried on the discretized searching space "pruned" by considering body motion dynamics and temporal consistency. The dynamically pruned searching space is much smaller the original searching space. For the body tracking task in our experiment, the dynamic BP reduce the complexity of BP from $O\left(20^6 \cdot 9\right)$ to $O\left(5^6 \cdot 9\right)$ if we choose the hypercube of size $5 \times 5 \times 5$ as the neighborhood of the previous configuration.

To summary, the propose dynamic BP algorithm can reduce the computational complexity of BP from $O(N^2 d)$ to $O(H^2 d)$, where $H \ll N$. H is the volume of a small neighborhood of the configuration of $t-1$ frame, in which the dynamic BP is carried out. Usually the larger the N is, the more the dynamic BP can reduce the computation. The efficiency of the dynamic BP depends on how well we can estimate the dynamics of the body motion. Ideally the dynamics should be learn from training data.

4 Experiment Results

We realize a 2d articulated body tracker based on dynamic BP and test it in real-meeting scenario. The 2d tracker achieve robust tracking for both frontal view (figure 2) and oblique view (figure 3). For the oblique view, at some frames the tracker lost the track of the lower arms. This is due to the self-occlusion and large 3D motion. The 2D articulated tracker cannot handle above 2 situations, therefore an articulated 3d tracker is our next step. However, the results in figure 3 do show the power of the graphical model. As soon as the person of interest takes a pose not very different from the initial pose, the tracker can recover true lower arm positions by inference from the torso and upper arm positions. We also test our tracker on a walking sequence with scale change. The results are shown in figure 4. The tracker loses the lower arms for some frames due to shading and similarity between the background and the skin color. So we will combine edge and color as features in our future work.

The proposed dynamic BP accelerate the tracker a lot. For each node, the size of over all searching space is $20^3 = 8000$. Therefore the complexity will be $O(20^6 \cdot 9)$ for one frame of video if we directly apply BP, while after dynamically prune the searching space, the dynamic BP has computational complexity of $O(5^6 \cdot 9)$, which is quite acceptable. Our articulated 2d tracker can achieve 2 frames/second in the shown 1024×768 video.

5 Conclusion

A 2d articulated body tracker based on dynamic BP on the graphical model is proposed in this paper. The discretized searching space is "pruned" by considering body motion dynamics and temporal consistency. The dynamically pruned searching space is much smaller the original searching space and therefore drastically reduce the computational complexity of the articulated tracker based on BP. The experiments on articulated tracking of the body movement in real meeting scenario show robustness and efficiency of the proposed algorithm. This articulated tracker is promising with the application to HCI and video annotation. The future step is going to 3D and further integrate temporal constraints

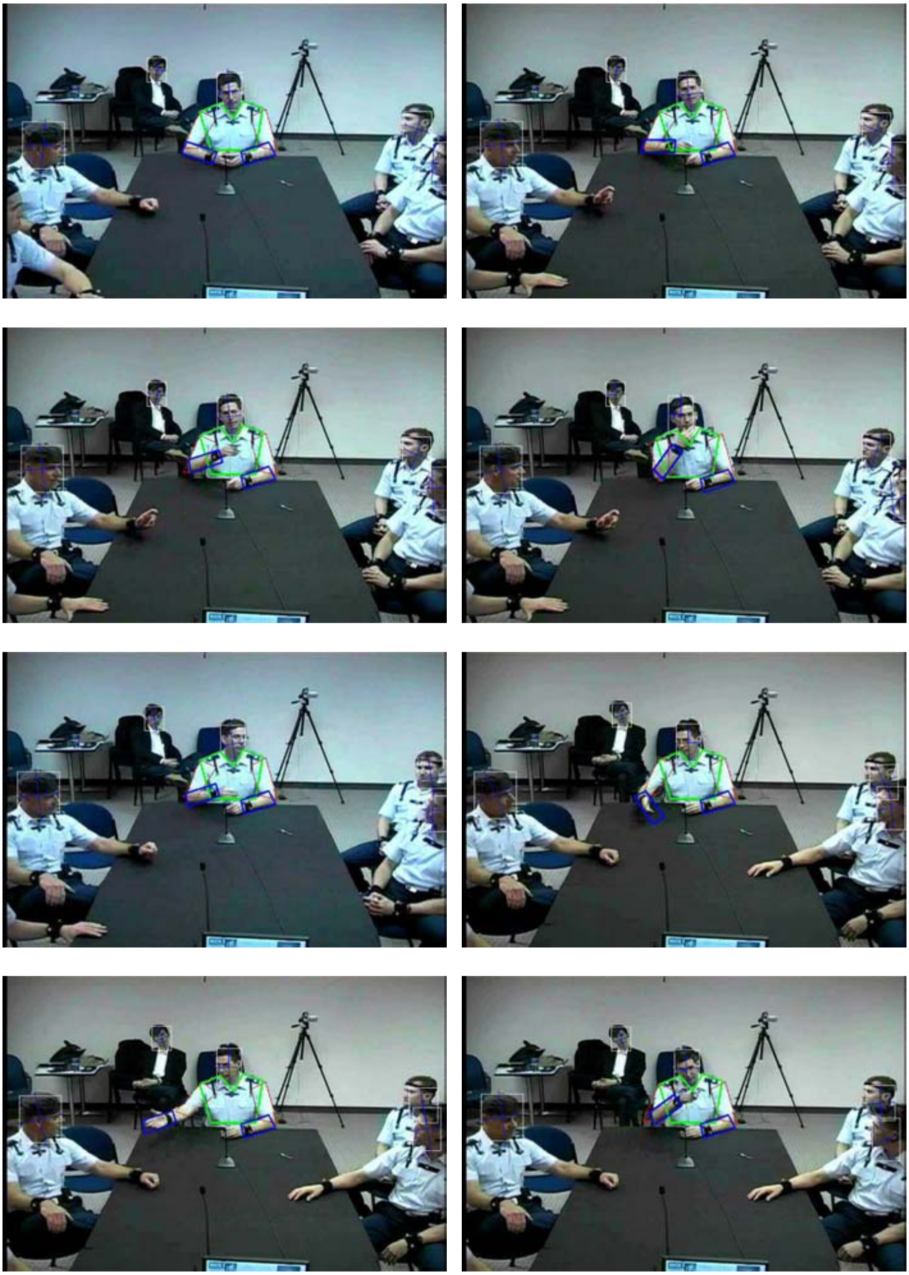


Fig. 2. 2D articulated body tracking results for meeting scenario, frontal view

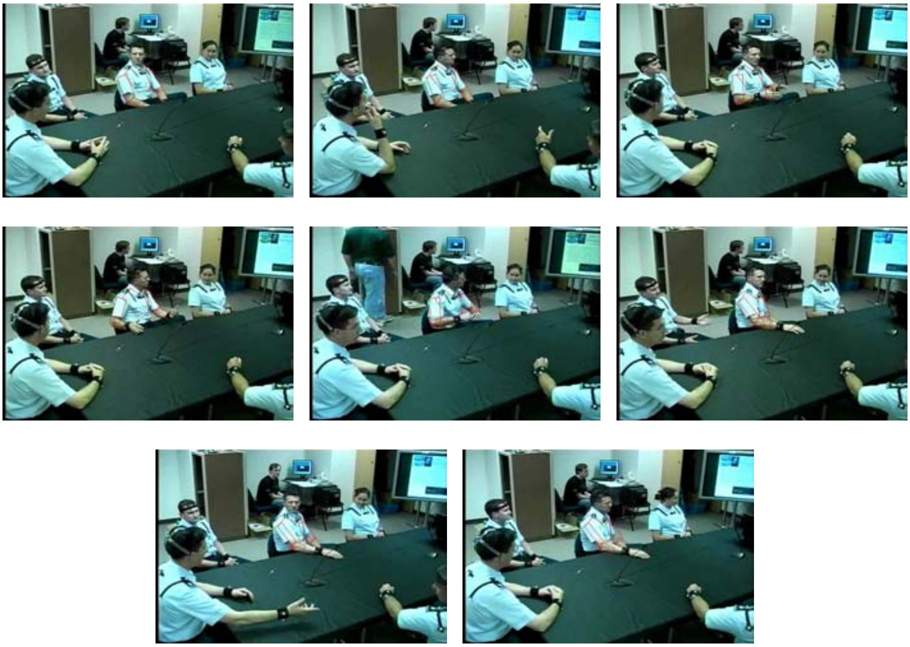


Fig. 3. 2D articulated body tracking results for meeting scenario, oblique view

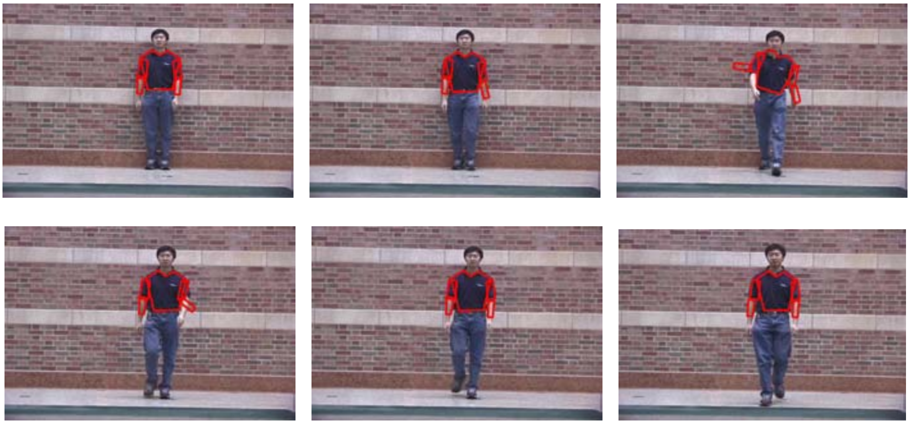


Fig. 4. 2D articulated body tracking for a walking sequence

across the whole video instead of between adjacent frames, like what we have done in [14].

Acknowledgments

This work was supported in part by National Science Foundation Grant IIS 00-859890 and ARDA Contract MDA904-03-C-1787.

References

1. J. Deutscher, A. Blake, and I. Reid, "Articulated body motion capture by annealed particle filtering," in *CVPR00*, pp. II: 126–133, 2000.
2. D. Ramanan and D. A. Forsyth, "Finding and tracking people from the bottom up.," in *CVPR*, pp. 467–474, 2003.
3. L. Sigal, S. Bhatia, S. Roth, M. Black, and M. Isard, "Tracking loose-limbed people," in *CVPR04*, pp. I: 421–428, 2004.
4. E. B. Sudderth, M. I. Mandel, W. T. Freeman, and A. S. Willsky, "Distributed occlusion reasoning for tracking with nonparametric belief propagation," in *Advances in Neural Information Processing Systems 17*, pp. 1369–1376, Cambridge, MA: MIT Press, 2005.
5. E. Sudderth, T. Ihler, W. Freeman, and A. Willsky, "Nonparametric belief propagation," *Proc. CVPR*, pp. 605–612, 2003.
6. P. Felzenszwalb and D. Huttenlocher, "Efficient matching of pictorial structures," in *CVPR*, pp. 66–73, 2000.
7. Y. Wu and T. Huang, "Hand modeling, analysis and recognition," *IEEE Signal Processing Magazine*, vol. 18, pp. 51–60, May 2001.
8. L. Sigal, M. Isard, B. H. Sigelman, and M. J. Black, "Attractive people: Assembling loose-limbed models using non-parametric belief propagation," in *NIPS*, 2003.
9. C. Bregler and J. Malik, "Tracking people with twists and exponential maps," in *CVPR*, p. 8, 1998.
10. D. Ramanan, D. A. Forsyth, and A. Zisserman, "Strike a pose: Tracking people by finding stylized poses.," in *CVPR*, 2005.
11. J. Pearl, *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*. Morgan Kaufmann Publishers Inc., 1988.
12. F. Kschischang and B. Frey, "Factor graphs and the sum-product algorithm," *IEEE Trans. on Information Theory*, vol. 47, pp. 498–519, February 2001.
13. W. Freeman, E. Pasztor, and O. Carmichael, "Learning low-level vision," *IJCV*, vol. 40, pp. 25–47, 2000.
14. M. Liu, T. X. Han, and T. S. Huang, "Online appearance learning by template prediction," in *AVSS*, 2005.
15. J. Yedidia, W. Freeman, and Y. Weiss, "Understanding belief propagation and its generalization," 2001.
16. K. P. Murphy, Y. Weiss, and M. I. Jordan, "Loopy belief propagation for approximate inference: An empirical study," in *Proc. Uncertainty in AI*, pp. 467–475, 1999.
17. Y. Wu, G. Hua, and T. Yu, "Tracking articulated body by dynamic markov network," in *ICCV03*, pp. 1094–1101, 2003.

Recover Human Pose from Monocular Image Under Weak Perspective Projection

Minglei Tong[1], Yuncai Liu[1], and Thomas S. Huang[2]

[1] Institute of Image Processing and Pattern Recognition Shanghai Jiao Tong University,
P.R. China, 200030
{tongminglei,whomliu}@sjtu.edu.cn

[2] Beckman institue, University of Illinois at Urbana-Champaign, Urbana, IL 61801
huang@ifp.uiuc.edu

Abstract. In this paper we construct a novel human body model using convolution surface with articulated kinematic skeleton. The human body's pose and shape in a monocular image can be estimated from convolution curve through nonlinear optimization. The contribution of the paper is in three folds: Firstly, human model based convolution surface with articulated skeletons is presented and its shape is deformable when changing polynomial parameters and radius parameters. Secondly, we give convolution surface and curve correspondence theorem under weak perspective projection, which provide a bridge between the 3D pose and 2D contour. Thirdly, we model the human body's silhouette with convolution curve in order to estimate joint's parameters from monocular images. Evalution of the method is performed on a sequence of video frames about a walking man.

1 Introduction

Over the course of the past decade, human motion analysis has received more and more attentions in computer vision research community with applications in areas like marker-less motion capture for animation, virtual reality, human-computer interaction and intelligent surveillance, etc. Various model-based methods have been proposed for the estimation and analysis of full body's structure, therefore many models, deformable or rigid, [2,8,9,10,13,14,15,20] are given for human motion capture or tracking.

Our goal in this paper is to build a new human body model using convolution surface whose shape can be modulated by polynomial function. Convolution surface was firstly introduced into computer graphics by Bloomenthal and Shoemake[1]. In fact, convolution surface is a kind of special implicit surface using 3D convolution between skeleton (line or curve) and some kernels. As mentioned in Xiaogang Jin [3], if the kernels are modulated by a polynomial function along a skeleton, different shape will be produced along the skeleton. Chiew-Lan Tai [4] presents prototype model from sketched silhouettes based on convolution surfaces.

Another goal of the paper is to determine the model's shape and estimate the body's pose from image. We extract the silhouette of a human body with different gestures and trace the boundary of the human body's region in image, then we approximate the contour with convolution curve. According to the correspondence theorem between 3D surface and 2D contour, we can estimate the 3D joint parameters by fitting the convolution curve with points of human contour. The relevant polynomial

N. Sebe, M.S. Lew, and T.S. Huang (Eds.): HCI/ICCV 2005, LNCS 3766, pp. 36–46, 2005.

coefficients, which are used to adjust the shape of the model, can be estimated in model initialization process.

In the past decades, many methods about deformable model reconstruction and human motions capture based on image or video are presented in computer vision. Some of methods are proposed depending on extracting silhouettes and fitting body models to them. The researches on model based human motion analysis have been reviewed in [5][6][7]. Traditionally human bodies are represented as stick figures [8], 2D contour [9] or volumetric model [10].

Recent works are inclined to make more accurate deformable human model with ingenious computer graphics techniques. C.Bregler [11][12] made use of factorization techniques, which built the model with a linear combination of prototype shapes. Under this model, the tracking matrix is of higher rank, and can be factored in a three steps process to yield pose, configuration and shape. But the technique has not been applied in full body tracking. R.Plankers and P.Fua [14] proposed articulated soft objects, in which the skin surface is presented as a series of implicit volumetric primitives attached to skeletons. Each primitive defines a field function (soft objects) and the skin is taken to be a level set of the sum of these fields. The model is optimized by observations from silhouettes and stereo depth. Though Plankers made use of very elaborate method and got a very accurate model, the model requires too many parameters to represent. P.Sand and L.McMillan [13] described a very novel method for human body reconstruction from silhouettes. They represent the human skin surface using points along needles that are rigidly attached to a skeleton. C.Sminchisescu [15] proposed a human body model consists of kinematics skeletons of articulated joints covered by flesh built from super-quadric ellipsoids with additional tapering and bending parameters.

The human model is described in section 2. Image process is mentioned in section 3; Parameters estimation method is described in Section 4. The experiments list in Section 5.

2 Human Model

The convolution surface model in this paper is constructed through convolution between articulated skeletons and Cauchy kernel function, and the whole body needs 32 articulated line skeletons.

2.1 Convolution Surface and Curve

Bloomenthal and Shoemake [1] introduced convolution surface along line skeleton as a logical extension of point based implicit surface models. Xiaogang Jin [3] presents an analytical solution for convolving line segment skeletons with a variable kernel modulated by a polynomial function.

A convolution surface is a level set of points (x, y, z) that satisfy

$$F(x, y, z) \ -T = 0 \tag{1}$$

T is constant and $F(x, y, z)$ is field function, achieved through a 3D convolution of a kernel function with a skeleton function $g(P)$:

$$\int_V g(Q)f(P-Q)dv - T = 0 \tag{2}$$

integrating all the points $Q \in V$

$$g(P) = \begin{cases} 1 & P \in V \\ 0 & P \notin V \end{cases} \tag{3}$$

Equ.2 can be expressed as the following form:

$$(f \otimes g)(P) - T = 0 \tag{4}$$

A line segment of length l with a start point $\vec{b} = (b_1, b_2, b_3)$ and unit direction $\vec{a} = (a_1, a_2, a_3)$, can be noted parametrically as:

$$L(t) = \vec{b} + t.\vec{a} \qquad 0 \le t \le l \tag{5}$$

$\vec{b}$ and $\vec{a}$ are 3 dimensional vector. Letting $\vec{d} = \vec{P} - \vec{b} = (x - b_1, y - b_2, z - b_3)$, the squared distance from a point $\vec{P}$ to a point on the line $L(t)$ descriped in Fig.1, is

$$\|\vec{r}(t)\|^2 = \|\vec{d}\|^2 + t^2 - 2t\vec{d} \bullet \vec{a} \quad \vec{P} \notin L(t) \tag{6}$$

Cauchy function [16] was proved to be a very useful kernel in surface modeling:

$$f(\vec{r}) = \frac{1}{(1 + s^2\|\vec{r}\|^2)^2} \tag{7}$$

s is radius parameter used to control the width of the kernel. Equ.2 can be regarded as a line integral with unit line density. In this section we will introduce a polynomial density distribution to the line:

$$\int_0^l \left(\sum_{i=0}^{n} q_i t^i\right) \frac{1}{(1 + s^2\|\vec{r}\|^2)^2} dt - T = 0 \tag{8}$$

We choose the first two coefficients q_1 and q_2 to construct a human model with 32 skeletons. Convolution curve can be achieved though a 2D line segment convolved with 2D kernel function modulated by polynomial function.

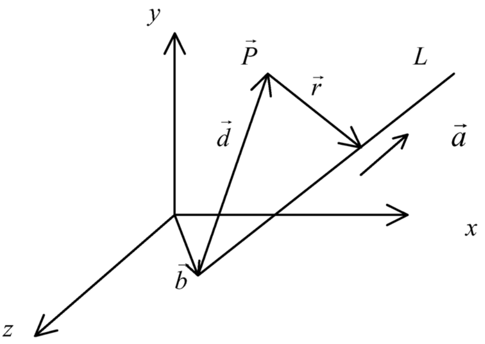


Fig. 1. Line segment L and vector $\vec{r}$

$$\int_0^{l'} \left(\sum_{i=0}^{1} q_i t^i\right) \frac{1}{(1+s^2\|\vec{r}'\|^2)^2}\, dt - T = 0 \tag{9}$$

l' is the length of a 2D line segment L', and $\vec{r}'$ is 2D vector

$$\|\vec{r}'(t)\|^2 = \|\vec{d}'\|^2 + t^2 - 2t\vec{d}' \bullet \vec{a}' \quad \vec{P}' \notin L'(t) \tag{10}$$

where $\vec{b}'$ is the beginning point of L', $\vec{a}'$ is unit direction of L', and $\vec{d}' = \vec{P}' - \vec{b}' = (x - b_1', y - b_2')$.

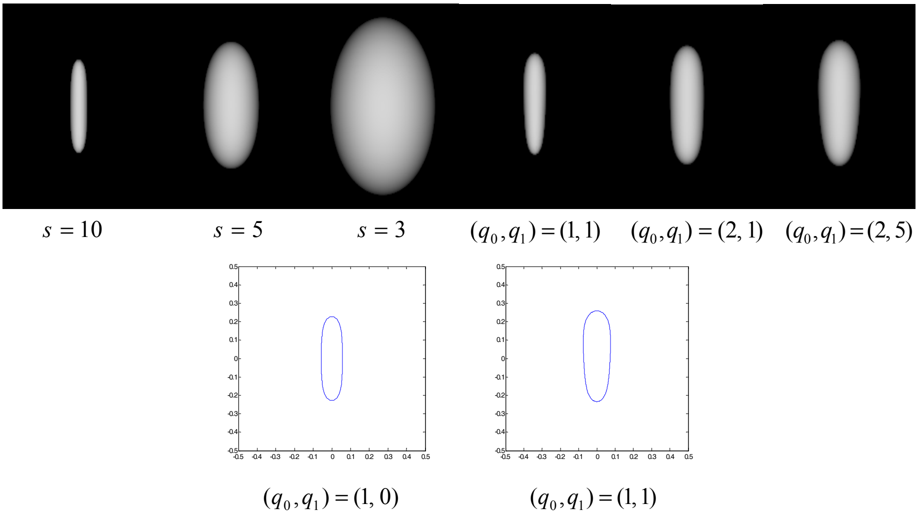


Fig. 2. Adjusting s and (q_0, q_1) can change the shape of the convolution surface and curve

2.2 Convolutions Surface and CurveCorresponedence

In this section, we present convolution surface and curve correspondence theorem under weak perspective projection without proof, which provide a bridge between the 3D pose and 2D contour.

Theorem: [Convolution Surface And Curve Correspondence Theorem Under Weak Perspective Projection]
If a surface $P(x, y, z) = 0$ satisfying Equ.8 is projected onto a plane along z direction under weak perspective projection, the contour on the projection plane is almostly a convolution curve $C(x, y) = 0$ and its line skeleton L' is the weak perspective projection of L.

Definition: $P(x, y, z)$ and $C(x, y)$ are called convolution surface and curve correspondence under weak perspective projection, if $P(x, y, z)$ and $C(x, y)$ satisfying convolution surface and curve correspondence theorem.

2.3 Human Body Model and Initialization

In this section, we constructed the human body model with 32 articulated skeletons. The pose of the model can be easily controlled when changing the joint angles of the skeletons. The model surface can be represented by the following formula:

$$\sum_{j=0}^{31}\left(\int_{0}^{l_j}\left(\sum_{0}^{1} q_{ij}t^i\right)\frac{1}{(1+s_j^{\,2}\left\|\vec{r}_j\right\|^2)^2}\right)dt - T = 0 \tag{11}$$

where q_{ij} is the $i\ th$ parameter of the $j\ th$ skeleton. The model shape can be deformed when changing s_j and q_{ij}.

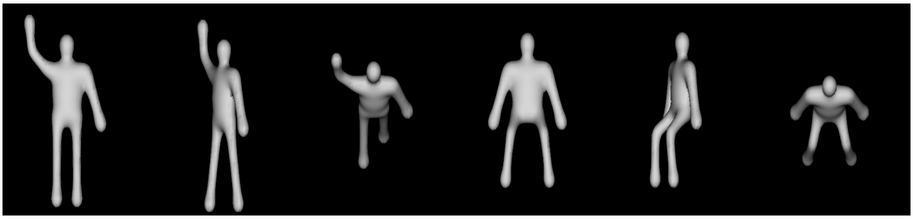
A typical model in our paper has about 23 joint parameters X_J, plus 14 shape parameters X_D and 5 global location parameters X_L. A completed model can be encoded as a single parameter vector $X=(X_J, X_D, X_L)$. In this paper, the articulated skeletons are modeled with twists relative to the center of the body [11][19]. Some joint angles will be limited in this paper, for example the elbows and the knees cannot bend backward and the neck can rotate in a limited range etc. Two gestures in different viewpoint are shown in Fig. 3.

The correspondence convolution curve of the model is

$$\sum_{j=0}^{31}\left(\int_{0}^{l_j'}\left(\sum_{0}^{1} q_{ij}'t^i\right)\frac{1}{(1+s_j^{\,2}\left\|\vec{r}_j^{\,\prime}\right\|^2)^2}\right)dt - T = 0 \tag{12}$$

The numbers of DOF of articulated skeleton joints are described in Table1.

In human model initialization process, the proportion of every skeleton is designed according to anthropometry. We estimate the shape parameters X_D through fitting the Equ.12 using human contour points on image if given a specific pose (It is assumed that we know the joint parameters). Trust region method [17][18] for nonlinear minimization methods are used to estimate the shape parameters.



a. b. c. d. e. f.

Fig. 3. a, b, c and d, e, f show human body with two gestures in different viewpoint

3 Foreground Segmentation and Boundary Smoothing

Background subtraction has been widely used in foreground detection and segment where the observed object moves in static background. Firstly, we model the background of video image sequence using the method LMedS (Least Median of Squares)

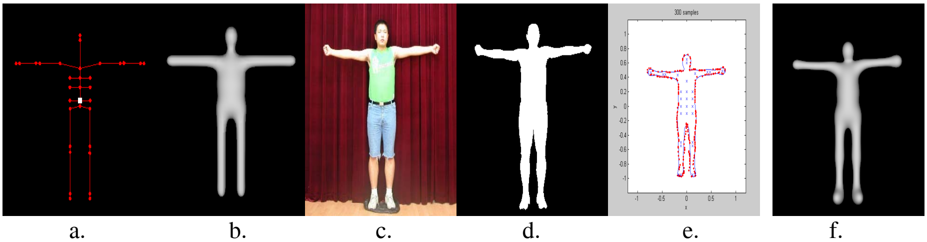
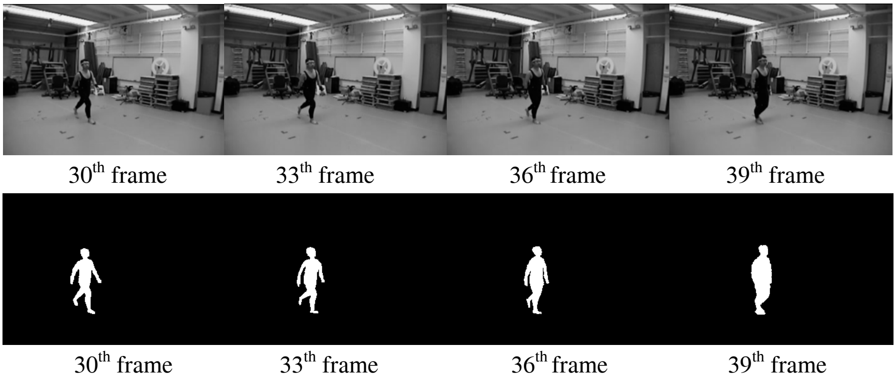


Fig. 4. a and b shows the skeleton and convolution surface model without image information; b,c,d and f show the initialization process with a specified gesture


30th frame 33th frame 36th frame 39th frame

30th frame 33th frame 36th frame 39th frame

Fig. 5. First row is the gray video sequences and second row is image after foreground segmentation

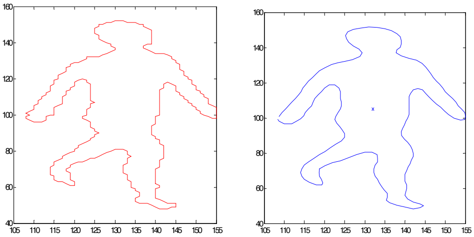


Fig. 6. The figure with red line is the rough contour and the figure with blue line is the contour after smoothing

method [21]. Then subtraction method in [22] is introduced to segment the foreground. Finally, we use morphological operators to eliminate the spurious pixels and small holes. The result is shown in Fig 5. In order to accelerate the parameter optimization, we must smooth the human contour to get rid of the jagged boundary after the boundary tracing. Here 5-neighborhood average method is used to smooth the contour, in Fig 6. Assumed that (x_i', y_i') $i \in 1,2,\cdots,M$ are points on the traced contour, the smoothed contour points are:

$$(x_i, y_i) = \left(\frac{\sum_{i-2}^{i+2} x_i'}{5}, \frac{\sum_{i-2}^{i+2} y_i'}{5}\right) \; i \in 1,2,\cdots,M \tag{13}$$

Table 1. The DOF numbers of the human model

Body part	Number of DOF
The whole body	4
Shoulder	1*2
Upper arm	3*2
Lower arm	1*2
Hand	1*2
Thigh	3*2
Knee	1*2
Foot	1*2
Neck	1

4 Joint Parameters Estimation

In this section, we estimate joint and global location parameters of the human pose from monocular image using deterministic nonlinear constraint optimization method. We do not care about the scene depth for the camera model we use is weak perspective projection.

4.1 Object Function

The projection contour of the human model is approximated with a convolution curve in Equ.12. If given

$$C(x,y,X)=\sum_{j=0}^{31}\Big(\int_0^{l_j'}\big(\sum_0^1 q_{ij}' t^i\big)\frac{1}{(1+s_j^2\left\|\vec{r}_j'\right\|^2)^2}\Big)dt-T \tag{14}$$

X is parameters vector to be estimatied. Given the contour point (x_k, y_k), $k \in 1,2,\cdots,M$, we attempt to describe the objective function as

$$F(X)=\min\sum_k C^2(x_k,y_k,X) \tag{15}$$

4.2 Joints and Skeletons Constrains

In high dimensional parameters optimization, it is very difficult to get global optimization solution. Constrains term must be given for reduce local minima. Here we give a joint and skeleton constrains term. It means the 2d points, which are the orthogonal projection points of 3D joint points on arms, hands, legs, feet and head, must approximate the human region skeleton closely as possible.

Firstly the skeleton of the human region in image should be subtracted. Assumed that Q is skeleton point set and P is the point set of 2D body joint points. We define the distance between a point $\vec{p}_i \in P$ and Q,

$$D(p_i,Q)=\min_j\left\|\vec{p}_i-\vec{q}_j\right\|^2 \quad \forall\vec{q}_j \in Q \tag{16}$$

The constraint term will be decriped as:

$$G(X) = \sum_{i} D(p_i, Q) < T \tag{17}$$

Equ.17 is the nonlinear constraint of objective function and T is an experimental number.

4.3 Nonlinear Optimization

Human pose estimation can be reduce to the following nonlinear optimization problem:

$$\begin{aligned} &\min F(X) \\ &\text{s t } G(X) < T \end{aligned} \tag{18}$$

Many classical methods can solve this kind of problem [17][18].

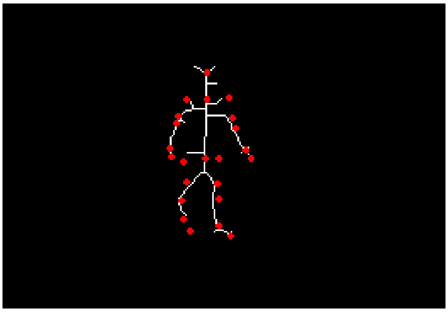


Fig. 7. The sketch map of the joint and human skeleton likelihood

4.4 Initial Value

The very important step is to estimate initial values of parameters. Our estimation has to start with an initial known pose in the first frame and the init joint value in next frame equals to the last estimation result for the correspondence joints have small change between two consecutive frame, while the init location value will be set to the center of the human region in image.

5 Experiments

This section experiments are given to show the performance of of our estimation methods with a sequence of video image from CMU motion data and we recover a walking human pose with 3D model. We provide many test result including the recovered 3D human model.

The video frame has the resolution of 352*240 pixels. First we show a sequence of video frames (after RGB convert to gray) in ath row of Fig.8. In bth row, blue line in Fig.8 is the human contour from image and the green line is convolution curve (Enlarged in Fig.9); blue points in figure are orthogonal projection of 3D joint points. In cth rows, we give the estimated human skeleton. In dth and eth row of Fig.8, we show the recovered pose of every frame.

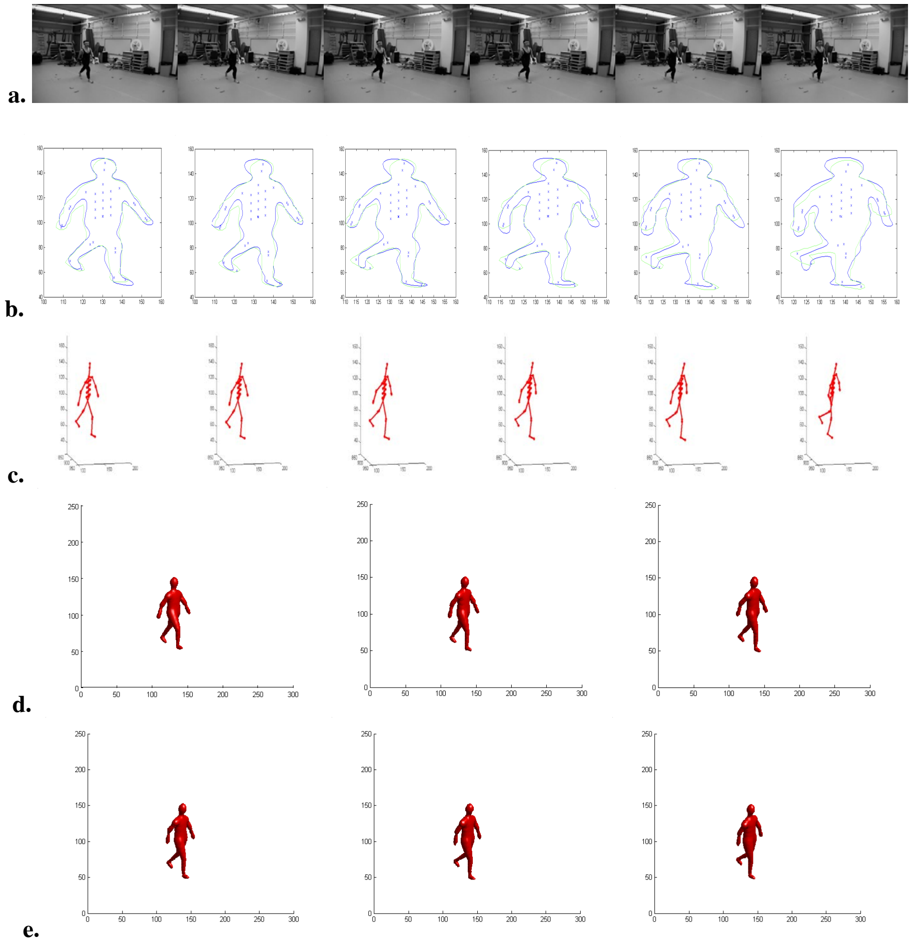


Fig. 8. Continuous 6 frame image and the recovered 3D model of a walking man from CMU video datasets. The ath row shows six video sequences of a person walkinge. The bth row shows the human contour and convolution curve. The cth row shows the estimated 3D skeleton. The dth and eth row show the recovered 3D walking pose

In the list experiments, our methods works well without severe self-occlusion because we can make use of the joint and image skeleton constrain. The tracking work under severe self-occlusion will be done in the future work.

We also do a contrast experiment in optimization algorithm, described in **Fig.9**. One group uses joint and skeleton constrain and another group does not. The first group shows a well estimation result and the second group is trapped in local minima.

6 Conclusions and Future Work

We have built a new human body model using convolution surface in this paper. 3D human model parameters determination method is also presented through convolution surface and curve correspondence. We extract silhouette of a human with specified gesture and initilize the deformable model. The convolution surface and curve correspondence theorem under weak pespective projection is proved, which brings a bridge between the 3D convolution surface and 2D convolution curve. We also present a new constraint (joint and skeleton distance likelihoods), which is very efficient without severe self-occlusion.

In future work we will give a new human tracking framework from monocular image and solve the problem of the joint angle ambiguity under monocular image Limb self-occlusion and tracking algorithm, not mentioned in current work will be also our aim.

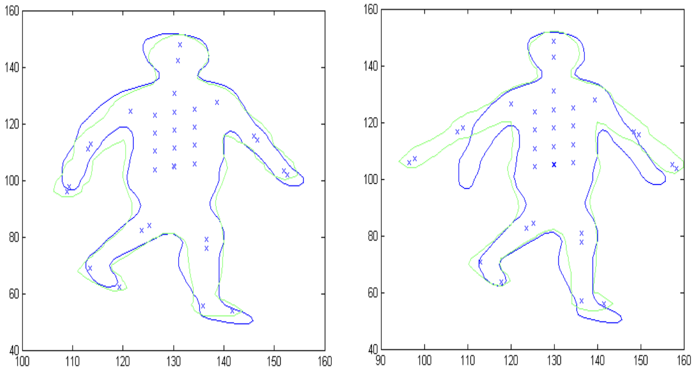


Fig. 9. Fist image shows us the optimization with joint and skeleton constrain and second image shows us the optimization without joint and skeleton constrain

Acknowledgement

Work is supported by China project NSFC 60375009.

References

1. J.Bloomenthal and K.Shoemake. Convolution surface. *Computer Graphics*, 25 (4): 251-256, 1991
2. Douglas DeCarlo and Dimitri Metaxas, Blended Deformable models, *PAMI* v18 (4): pp 443-448,1996
3. Xiaogang Jin and Chiew-Lan Tai. Convolution Surfaces for Line Skeletons with Polynomial Weight Distributions. *Journal of Graphics Tools ACM Press,* 6(3): 17-28, 2001.
4. Chiew-Lan Tai, Hongxin Zhang and Chun-Kin Fong. Prototype Modeling from Sketched Silhouettes based on Convolution Surfaces. Computer Graphics Forum, 23(1), 2004.
5. Thomas B. Moeslund and Erik Granum. A Survey of Computer Vision-Based Human Motion Capture. *Int. Journal of Computer Vision and Image Under- standing* 81(3): pp. 231 - 268, 2001.
6. J. K. Aggarwal, Q. Cai, W. Liao, and B. Sabatay. Non-rigid Motion Analysis: Articulated and Elastic Motion. *Int. Journal of Computer Vision and Image Understanding* 70(2), pp. 142156, 1998
7. D. M. Gavrila. The visual analysis of human movement: a survey. *Computer Vision and Image Understanding*, 73 (1): pp 82-98, 1999.

8. K. Shoji, A. Mito and F. Toyama, □Pose estimation of a 2-D articulated objects from its silhouette□. *Proc. 15th Int. Conf. On Pattern Recognition,* Vol. 3, pp.713-717, 2000.
9. S. X. Ju, M. J. Black and Y. Yacoob, □Cardboard people: a parameterized model of articulated image motion□. *Proc. 2nd Int. Conf. On Automatic Face and Gesture Recognition*, pp.38-44, 1996.
10. D. Gavrila and L. Davis, □3-D model-based tracking of human in action: a multi-view approach□. In *CVPR*. San Francisco, pp. 73-80, 1996.
11. C.Bregler and J. Malik. □Tracking people with twists and exponential maps□. In *CVPR*, pp. 8-15, 1998.
12. C.Bregler, H.Aaron and H.Biermann. □Recovering Non-Rigid 3D Shape from Image Streams□. In *CVPR* ,vol.2 2000.
13. P.Sand, L.McMillan, and Jovan Popovic. □Continuous Capture of Skin Deformation□. *In SIGGRAPH* 2003
14. R.Plankers and P.Fua. □Articulated soft objects for multi-view shape and motion capture□. *PAMI*, 25(10), 2003.
15. C. Sminchisescu and A. Telea. □Human pose estimation from silhouettes. a consistent approach using distance level sets□. *In WSCG International Conference on Computer Graphics, Visualization and Computer Vision*, 2002.
16. A.Sherstyuk. □Kernel functions in convolution surfaces: A comparative analysis□. *The Visual Computer*, 1999, 15 (4): 171- 182
17. Mor□, J.J. and D.C. Sorensen, □Computing a Trust Region Step□. *SIAM Journal on Scientific and Statistical Computing*, Vol. 3, pp 553-572, 1983.
18. T.Steihaug □The Conjugate Gradient Method and Trust Regions in Large Scale Optimization□. *SIAM Journal on Numerical Analysis,* Vol. 20, pp 626-637, 1983.
19. D.Tolani, A.Goswami and N.Badler. □Real-Time Inverse Kinematics Techniques for Anthropometric Limbs□. *Graphical Models*, 62:353-338,2000.
20. Xiaoyun Zhang, Yuncai Liu, and Thomas S. Huang. □Motion Estimation of Articulated Objects from Perspective View□. *AMDO*, Nov 2002, pp: 165-176.
21. Y. Yang and M. Levine, □The Background Primal Sketch: An Approach for Tracking Moving Objects□. *Machine Vision and Applications*, vol. 5, pp. 17-34, 1992.
22. Y. Kuno, T. Watanabe, Y. Shimosakoda, and S. Nakagawa. □Automated Detection of Human for Visual Surveillance System□. *Proc. Int'l Conf. Pattern Recognition*, pp. 865-869, 1996.

A Joint System for Person Tracking and Face Detection

Zhenqiu Zhang*, Gerasimos Potamianos, Andrew Senior,
Stephen Chu, and Thomas S. Huang*

IBM Thomas J. Watson Research Center
Yorktown Heights, NY 10598, USA
zzhang6@uiuc.edu, {gpotam,aws,schu}@us.ibm.com, huang@ifp.uiuc.edu

Abstract. Visual detection and tracking of humans in complex scenes is a challenging problem with a wide range of applications, for example surveillance and human-computer interaction. In many such applications, time-synchronous views from multiple calibrated cameras are available, and both frame-view and space-level human location information is desired. In such scenarios, efficiently combining the strengths of face detection and person tracking is a viable approach that can provide both levels of information required and improve robustness. In this paper, we propose a novel vision system that detects and tracks human faces automatically, using input from multiple calibrated cameras. The method uses an Adaboost algorithm variant combined with mean shift tracking applied on single camera views for face detection and tracking, and fuses the results on multiple camera views to check for consistency and obtain the three-dimensional head estimate. We apply the proposed system to a lecture scenario in a smart room, on a corpus collected as part of the CHIL European Union integrated project. We report results on both frame-level face detection and three-dimensional head tracking. For the latter, the proposed algorithm achieves similar results with the IBM "PeopleVision" system.

1 Introduction

Visual detection and tracking of humans in complex scenes is a very interesting and challenging problem. Often, input from multiple calibrated cameras with overlapping fields of view is available synchronously, and information about both the frame-view and space-level human location is desired. One such scenario of interest, considered in this paper, is human-computer interaction in smart rooms, where a speaker is presenting a seminar in front of an audience. The scenario is of central interest within the CHIL European Union integrated project, "Computers in the Human Interaction Loop" [1]. In data collected as part of the CHIL project, four fixed calibrated cameras located at the corners of a smart room capture video data, with the goal of locating and identifying the seminar presenter. Hence, both three-dimensional head position estimation, as well as

* Zhenqiu Zhang and Prof. Thomas Huang are with the Beckman Institute, University of Illinois, Urbana, IL 61801, USA. This work was performed while Zhenqiu Zhang was on a summer internship with the Human Language Technologies Department at the IBM Thomas J. Watson Research Center, and was supported by the European Commission under the integrated project CHIL, "Computers in the Human Interaction Loop", contract number 506909.

N. Sebe, M.S. Lew, and T.S. Huang (Eds.): HCI/ICCV 2005, LNCS 3766, pp. 47–59, 2005.

face detection at the available frame views is required. The information can be further utilized to obtain close-up views of the presenter, based on steerable pan-tilt-zoom cameras, in the seminar indexing and annotation, etc. Clearly therefore, in such a scenario, a visual system that combines face detection, tracking, and multicamera processing is feasible and desirable.

Significant research work has been devoted to the problems of face detection, single-camera and multicamera tracking. For face detection, a machine learning based approach has proved the most effective. For example, Rowley et al. [2] present a face detection system based on retinally connected neural networks (NNs), accepting as input the preprocessed image pixel values directly. Post-processing is then performed by appropriately combining the NN outputs by "and"/"or" operators, or by using an additional neural network to arbitrate them. Roth et al. [3] use a network of linear units. The SNoW learning architecture is specifically tailored for learning in the presence of a very large number of features. The system of Viola and Jones [4] makes a successful application of AdaBoost to face detection. The resulting system is an efficient, real-time frontal-view face detector.

The ability to detect faces under varying head pose is important in many real applications. A reasonable treatment for multiview face detection is a view-based method within the appearance-based framework. In the system of Schneiderman and Kanade [5], multiresolution information is used for different levels of a wavelet transform. The algorithm consists of an array of five face detectors in the view-based framework. Each is constructed using statistics of products of histograms computed from examples of the respective view. In general, while great success has been achieved for frontal-view face detection, much engineering work is needed for real-time multiview face detection.

For tracking, there also exist many successful algorithms proposed in recent years. In [6], Comaniciu et al. introduce an algorithm for real-time tracking of non-rigid objects seen from a moving camera. The central computational module is based on the mean shift iterations and finds the most probable target position in the current frame. In [7], Isard and Blake propose a new algorithm, called "condensation". The algorithm uses "factored sampling", previously applied to the interpretation of static images, in which the probability distribution of possible interpretations is represented by a randomly generated set. The method uses learned dynamical models, together with visual observations, to propagate the random set over time. The result is highly robust tracking of agile motion that runs in near real-time.

Multiple cameras provide additional information concerning the objects of interest, which could be used to improve the tracking system performance. Many multicamera tracking algorithms have been proposed in recent years. For example, in [8], Black and Ellis present such a method in the context of surveillance. Their approach exploits multicamera views to resolve object occlusion. Moving objects are detected by using background subtraction, and viewpoint correspondence between the detected objects is established by using ground plane homography. In [9], Hampapur et al. use two or more calibrated cameras to triangulate a moving object's position, originally obtained by background subtraction, and determine the steering parameters for a third, pan-tilt-zoom camera that is calibrated to the same coordinate system. The pan-tilt-zoom camera automatically acquires zoomed-in views of a person's head, while the person is in mo-

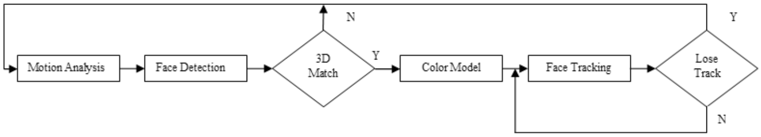


Fig. 1. Block diagram of the proposed multicamera face detection and tracking system

tion within the monitored space. An extended version of this system is also considered in this paper for the CHIL seminar presenter tracking task.

Even though there exit so many successful algorithms for face detection and tracking, how to efficiently combine their strengths, in order to obtain robust and real-time performance within a multicamera framework, is still an interesting and very important problem. Our work focuses on exactly this problem, in the context of the CHIL seminar task discussed at the beginning of this section. In particular, we propose to solve the problem by taking advantage of multiview face detection, color-based mean shift tracking, motion analysis, and utilizing calibration information for the available camera views. The overview diagram of our approach is depicted in Fig. 1: First, a three-dimensional face detection and tracking system is initialized by combining motion history, calibration information, and multiview face detection [10]. Subsequently, for tracking, a face model is constructed based on the color histogram of the face region in the hue/saturation/value (HSV) color space, and mean shift tracking [6] is applied to track the presenter's face in different views independently. At each frame, the tracking component is verified by local face detection and the calibration information. If it is determined that the tracking system has lost track, the detection and tracking system is re-initialized and the face model updated.

The paper is organized as follows: Section 2 presents FloatBoost [10] learning, which is applied to multiview face detection. The mean shift iteration algorithm [6] for face tracking is discussed in Section 3, and the proposed, combined vision system for multiview face detection and tracking is described in Section 4. For comparison purposes, an alternative approach based on the IBM "PeopleVision" system is described in section 5. Face detection and tracking experiments are presented in section 6, and conclusions are drawn in Section 7.

2 FloatBoost Learning for MultiView Face Detection

In [4], the AdaBoost algorithm was successfully applied to frontal face detection resulting in the first real-time frontal face detection system. However, if considered as a feature selection algorithm, AdaBoost is a sequential forward search procedure, which suffers from the so-called "nesting effect". Attempts to prevent the nesting of feature subsets have led to the development of floating search methods. FloatBoost [10] incorporates the idea of "floating search" [11] into AdaBoost to solve the "nesting effect" encountered in the sequential search of AdaBoost. A quality improvement of the selected features is gained at the cost of increased computation due to the extended search. In this paper, the FloatBoost learning algorithm is applied to multiview face detection in the smart room seminar scenario.

2.1 AdaBoost Learning

For two class problems, let us assume that a set of N labeled training examples is available, (x_1, y_1),..., (x_N, y_N), where $y_i \in \{+1, -1\}$ is the class label associated with example x_i. For face detection, x_i is an image sub-window of a fixed size (e.g. 20×20) containing an instance of the face ($y_i = +1$), or nonface ($y_i = -1$) pattern. In the notion of RealBoost (a real version of AdaBoost [4] as opposed to the original discrete one), a stronger classifier is a linear combination of M weak classifiers

$$H_M(x) = \sum_{m=1}^{M} h_m(x) \ , \tag{1}$$

where $h_m(x) \in \Re$ are weak classifiers. The class label for a test x is obtained as $H(x) = sign\{H_M(x)\}$ (an error occurs when $H(x) \ / \ \cancel{y)}$, while magnitude $|H_M(x)|$ indicates the confidence.

In boosting learning [12, 13], each example x_i is associated with a weight w_i, and the weights are updated dynamically using a multiplicative rule according to the errors in previous learning, so that more emphasis is placed on the examples that are erroneously classified by the weak classifiers learned previously. This way, the new weak classifiers will “focus more attention” to those examples. The stronger classifier is obtained as a proper linear combination of the weak classifiers.

The “margin” of an example (x, y) achieved by $H(x)$ (a single or a combination of weak classifiers) on the training examples can be defined as $yH(x)$. This can be considered as a measure of the confidence of the h’s prediction. The following criterion measures the bound on classification error [13]

$$J(H(x)) = E_w(e^{-yH(x)}) = \sum_{i=1}^{N} e^{-y_i H(x)}, \tag{2}$$

where $E_w(\bullet)$ stands for the mathematical expectation with respect to weight w over the examples (x_i, y_i).

The AdaBoost method constructs $h(x)$ by stage-wise minimization of (2). Given the current $H_{M-1}(x) = \sum_{m=1}^{M-1} h_m(x)$, the best $h_M(x)$ for the new strong classifier $H_M(x) = H_{M-1}(x) + h_M(x)$ is the one which leads to the minimum cost

$$h_M = arg \min_{h^+} J(H_{M-1}(x) + h^+(x)) \ . \tag{3}$$

2.2 FloatBoost Learning

AdaBoost is a sequential forward search procedure using the greedy selection strategy. Its heuristic assumption is the monotonicity. The premise offered by the sequential procedure can be broken down when the assumption is violated. FloatBoost [10] incorporates the idea of floating search [11] into AdaBoost to overcome the non-monotonicity problems associated with AdaBoost. The sequential floating search (SFS) method [11] allows the number of backtracking steps to be controlled instead of being fixed beforehand. Specifically, it adds or deletes $l = 1$ feature and then backtracks r steps, where

r depends on the current situation. It is such a flexibility that amends limitations due to the non-monotonicity problem. A quality improvement of the selected features is obtained at the cost of increased computation due to the extended search. The SFS algorithm performs very well in several applications [14]. The idea of floating search is further developed in [15], by allowing more flexibility for the determination of l.

These feature selection methods, however, do not address the problem of (sub-) optimal classifier design based on the selected features. FloatBoost combines them into AdaBoost for both effective feature selection and classifier design.

Let $H_M = \{h_1, ..., h_M\}$ be the so far best set of M weak classifiers; $J(H_M)$ be the criterion which measures the overall cost of the classification function $H_M(x) = \sum_{m=1}^{M} h_m(x)$ build on H_M; $J_m^{\min}$ be the minimum cost achieved so far with a linear combination of m weak classifiers for $m = 1, ..., M_{\max}$, where $M_{\max}$ denotes the maximum number of features (iterations) allowed in the boosting algorithm, initially set to a large value before the iteration starts. The FloatBoost learning [10] procedure involves training inputs, initialization, forward inclusion, conditional exclusion and output.

In the forward inclusion step, the currently most significant weak classifier is added one at a time, which is identical to AdaBoost. In the conditional exclusion step, FloatBoost removes the least significant weak classifier from H_M, subject to the condition that the removal leads to a lower cost than $J_{M-1}^{\min}$. Supposing that the removed weak classifier was the m'-th in H_M, then $h_{m'}, ..., h_M$ will be re-learned. The above steps are repeated until no more removals can be performed.

3 Mean Shift Tracking

Computational complexity is critical to most tracking applications, and therefore, exhaustive search in the neighborhood of the predicted target location for the best candidate is in most cases prohibitive. As a solution to this problem, mean shift tracking has been proposed [6]. Mean shift tracking is a real-time algorithm that aims to maximize the correlation between two statistical distributions.

The correlation or similarity between two distributions is expressed as a measurement derived from the Bhattacharyya coefficient [6]. Statistical distributions can be built using any characteristic discriminating to a particular object of interest. A model might use color, texture, or include both. In this paper, we model the target using the H channel in the HSV color space.

The discrete density q is estimated from the m-bin H-channel histogram of the HSV color in the face region, and p is estimated at a given location y from the m-bin histogram of the face candidate. The sample estimate of the Bhattacharyya coefficient is given by

$$\hat{\rho}(y) \equiv \rho(\hat{p}(y), \hat{q}) = \sum_{u=1}^{m} \sqrt{\hat{p}_u(y)\hat{q}_u} \; . \tag{4}$$

The distance between the two distributions can then be defined as

$$d(y) = \sqrt{1 - \rho(\hat{p}(y), \hat{q})} \; . \tag{5}$$

Starting at the predicted location y of the target computed by Kalman filtering [17], we search for the new target location in the current frame using mean shift iterations.

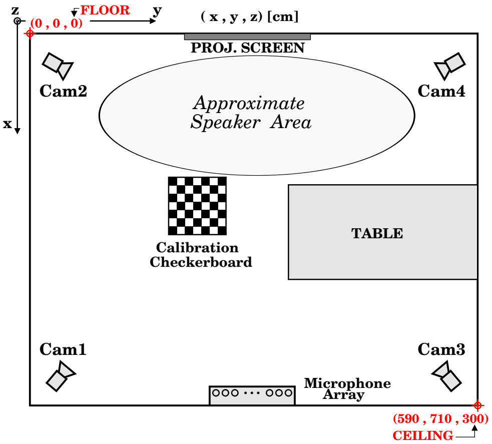


Fig. 2. Schematic diagram of the audio-visual sensors installed in the smart room at the University of Karlsruhe, Germany, as part of the CHIL project [1]. Data recorded by the four fixed cameras cam1–cam4 are used in our experiments

To minimize the distance (5), or, equivalently, maximize the Bhattacharyya coefficient, the Taylor expansion of p is used around the value of the coefficient at the target predicted position y_o. This yields

$$\rho[\hat{p}(y), \hat{q}] \approx \frac{1}{2}\sum_{u=1}^{m}\sqrt{\hat{p}_u(\hat{y}_o)\hat{q}_u} + \frac{1}{2}\sum_{u=1}^{m}\hat{p}_u(y)\sqrt{\frac{\hat{q}_u}{\hat{p}_u(\hat{y}_o)}} \, . \tag{6}$$

Position y_o can be just set to the current object position, or can be determined by Kalman filtering, as in our system.

The first term in (6) is independent of y, and the second term can be efficiently achieved based on mean shift iterations. This iterative optimization maximizes the value of Bhattacharyya coefficient in equation (4). At each iteration, a new location of the object is derived based on the mean shift vector [6]. Compared with exhaustive search, this iterative optimization algorithm is very efficient, with typically only several iterations needed to find the optimal target location.

4 Joint MultiCamera Face Detection and Tracking

As discussed in the Introduction, in this paper, we consider a particular scenario, where multiple calibrated cameras are set up in a smart room. The goal is to detect and track the presenter's face in the camera views (whenever visible), and to track the position of the presenter's head in the three-dimensional smart room space.

The setup of the room is as depicted in Fig. 2. This corresponds to the smart room located at the University of Karlsruhe, Germany, one of the CHIL project partners [1, 19]. Similar rooms are being set up in a number of CHIL project partners [1]. The room dimensions are 5.90×7.10 m, with a ceiling height of 3 m. A number of sensors are installed in the room which include the four fixed cameras, providing the data used in

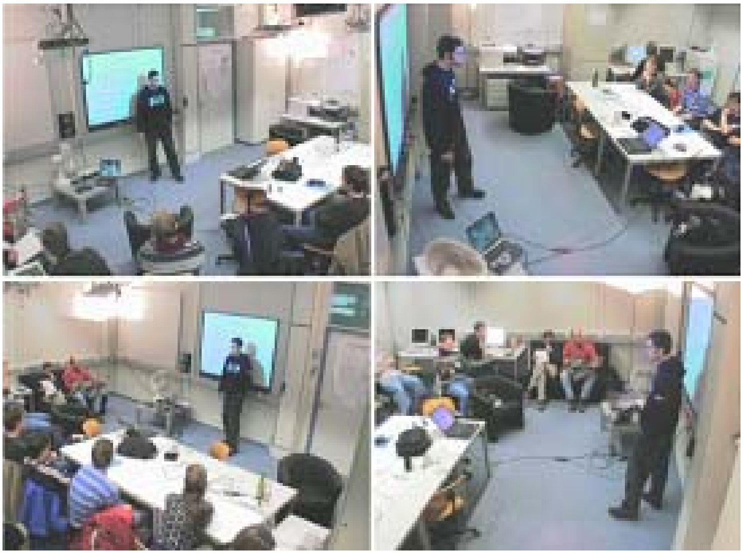


Fig. 3. Example frames captured by the four fixed cameras of the CHIL smart room

this paper. The cameras are located at the corners of the room, at about 2.7 m heigh (see also Fig. 2). They are SONY DFW-V500, and capture color data at a 640×480 pixel resolution and 15 frames per second through a firewire interface. For storage purposes and ease of access, JPEG compression is used for each frame. The cameras are synchronized and calibrated by the calibration toolbox [18], hence their relative position and orientation are known. A sample of synchronized images from the four camera views is shown in Fig. 3.

To detect and track the location of the presenter in the seminar room, three components are integrated in the proposed visual system: Initialization, tracking, and a reinitialization decision (see also Fig. 1). Notice that in the currently implemented algorithm, only two camera views are utilized. The three components, based on processing the two views, are described in more details in the following.

4.1 Initialization

In the initialization stage, we use three primary vision modules: Motion analysis, the camera model (based on calibration information), and face detection.

Motion History: First, independently for each camera view, motion history is estimated to rapidly determine where movement has occurred. The utilized algorithm is based on work by Davis and Bobick [16]. Obtaining a foreground silhouette is achieved through subtraction between two consecutive frames instead of background subtraction. As the person moves, the most recent foreground silhouette is copied as the highest value in the motion history image. The result is called the "motion history image" (MHI). MHI pixel values that fall below that threshold are set to zero. An example of the algorithm applied to two camera views is depicted in Fig. 4(a).

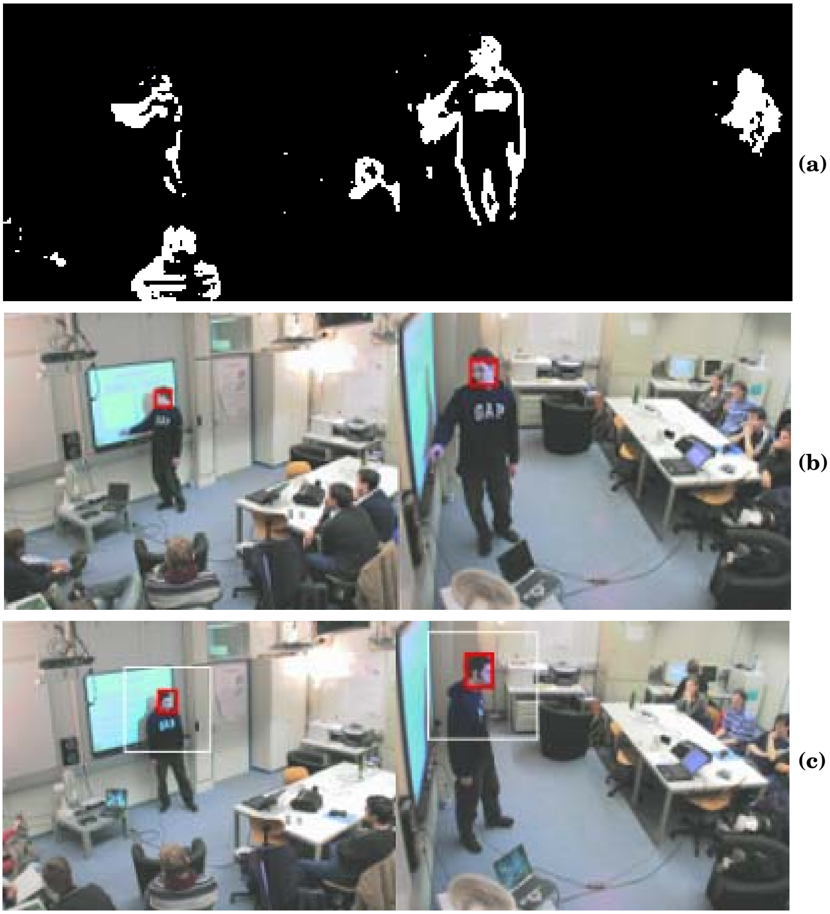


Fig. 4. (a) Motion history image (MHI) examples in two camera views; motion objects are segmented as foreground (white pixels). (b) Multiview face detection result when the detectors are applied locally around the foreground region. (c) Local multiview face detection in the tracking stage: Faces are detected within windows around the tracking results

Multi View Face Detection: Due to the size of the room, the resolution of the presenter's face is quite small in most of the CHIL data considered (typically, around 20×20 pixels). Robust face detection becomes difficult for such low resolutions, and as a result, the face detector threshold is adjusted to low values in order to keep high detection rate. This of course increases the false alarm rate. To deal with the problem, the multiview face detector is only applied to the foreground region, where motion occurred. Example detection results are depicted in Fig. 4(b). The detection results at each view can then be used to verify whether the detected faces belong to the same person. In particular, our system uses two trained face detectors: One for frontal view and the other for the left side view, since the right side face detector can be easily obtained by mirroring the left side one.

Camera Model: Recording images using a camera is equivalent to mapping three-dimensional object space points to image points in the film plane. For digitization, this recorded image will be projected again to the image in the projection plane. For simplicity, it is possible to directly relate the projected image and the object by a ray through the projection center.

In our system, given the results of face detection, rays are created by connecting the projection center with the head center at each camera view. The inter-ray distance is subsequently calculated. If the detection results are correct in both views, then they belong to the same person, and the inter-ray distance should be small; otherwise, the distance should be larger than a threshold, and the detection result erroneous. Based on this assumption, one can verify the detection results of the previous stage.

4.2 Tracking

Following the initialization component and the successful location of the presenter's face, the algorithm switches into its tracking mode. A color-based face model is first created of the detected face region for tracking in each of the two camera views. In this paper, the HSV color space is used for this purpose. In particular, the face model is created by the one-dimensional histogram of the H component in the HSV color space. The mean shift iteration is subsequently applied in the two view images separately to find the best target candidate.

4.3 Decision to Reinitialize

In our system, the re-initialization decision is based on local face detection and utilizes the camera model. In more detail, at each frame, a multiview face detector is applied around the tracking result to determine whether there is a face object in the local region, as shown in Fig. 4(c) (in our system, this is a 80×80 pixel region). If a face could not be detected in the local region for several frames, a re-initialization decision is taken. Similarly, if the inter-ray distance of the two-camera rays is larger than a predetermined threshold, this indicates that the two tracked results are inconsistent, hence prompting re-initialization. In our system, such decision is taken every 5 to 10 frames.

4.4 System Specifics

The resulting algorithm runs at approximately 5 frames per second on a Pentium four, 2.8 GHz desktop, with 512 MBytes of memory. Nevertheless, the current implementation is not optimal, and we believe that the algorithm speed can be substantially improved.

As already mentioned, the algorithm operates using two camera views, but of course can be readily extended to accommodate more camera inputs. For the CHIL data, it uses inputs from the cam1+cam2 or the cam3+cam4 cameras, depending on which set contains higher percentage of frontal faces, as determined on a development data set (see Section 6).

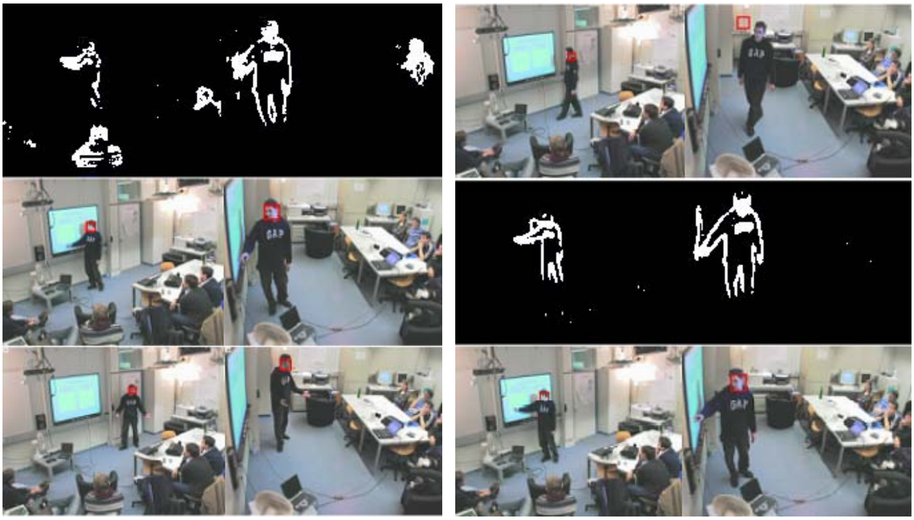


Fig. 5. Typical operation of the proposed multicamera face detection and tracking system. Depicted in left column: Frame 1: Motion analysis. Frame 5: Tracking initialization. Frame 283: Continued tracking. Right column: Frame 324: Lost tracking detected. Frame 331: Foreground segmented by motion analysis. Frame 334: Tracking re-initialization

An example of the tracking algorithm applied on the CHIL data is depicted in Fig. 5: In frame 5, motion analysis is applied and the foreground object is segmented. The presenter's face is located by the multiview face detector, and the detection result is verified by the camera model. Subsequently, the face color model is constructed in the detected face region. In the next hundreds of frames, color-based mean shift tracking successfully locates the presenter's face in the two camera views. However, by frame 324, the tracking has failed. Therefore, the system returns to its initialization component, and at frame 334 re-emerges in the tracking mode, after successfully applying re-initialization.

5 A Background Subtraction Tracker

For comparison purposes, we briefly present an alternative head tracking system, based on the IBM "PeopleVision" system [9], properly modified for use on the CHIL data. The system uses background subtraction based object detection, that utilizes a multiple Gaussian color model at each pixel, and object tracking based on the tracking method described in [20].

The system is first applied on each of the four camera views independently. At each frame, the 2D tracker is applied, and the resulting 2D probabilistic models are used to determine the top of the presenter's head. These 2D object points are then considered as hypotheses of the presenter's head top in the 3D space, based on camera calibration information and a minimum threshold for inter-ray distances between all candidate pairs in the four views. The best resulting 3D candidates at each frame are then considered to obtain the "optimal" temporal sequence / track of the presenter's head, using a Viterbi decoding approach.

6 Experiments on the CHIL Corpus

For our experiments, we use the CHIL seminar database collected at the University of Karlsruhe (UKA). There are a total of 12 video sequences of speakers holding seminars in front of audiences inside the UKA smart room, with the first 7 seminars collected in 2003 (referred to as the Sem03 data), and the remaining 5 recorded in 2004 (Sem04 data). For every seminar, segments are allocated to the development and the test sets, each containing approximately 8,000 frames for each of the four available camera views per seminar. Ground truth labels of the 2D face center coordinates (a point of center of the head) in individual camera images are provided every 10 frames, but only when faces are visible (i.e., when the nose is visible). For the 2004 seminars, a bounding box of each visible face is also provided.

For multiview face detection, and as mentioned in Section 4, two face detectors are trained: One for the frontal view and the other for the left side view (the right side face detector is obtained by mirroring the latter). A number of frontal and left-side view face images are cropped from selected images in the development set for this purpose. In addition, non-face training samples are cropped from an image database that does not include faces. The two face detectors are trained using the FloatBoost approach described in Section 2. For the frontal face detector, a cascade structure of 15 layers and 576 features is obtained, trained on 1606 annotated images. For the left side view, the cascade structure consists of 30 layers and 4330 features, trained on 1542 annotated images.

The face detection accuracy of the combined detection and tracking scheme is depicted in Table 1. Note, that in order to obtain face detection estimates at the two non-used camera views, the face detection step is also applied to them around the 2D camera view point that corresponds to the 3D estimate of the presenter's head. Notice that the face detection accuracy is defined as the percentage of frames that contain detected faces leading to errors between the face and the label centers less than half of the labeled face size. This is readily available for the labeled Sem04 data, but for the Sem03 data it is set to a default value of 30 pixels.

Table 1. Face detection accuracy of the proposed algorithm on the CHIL seminar data

Data	Sem03	Sem04
Face detection accuracy	76.16%	51.21%

The proposed algorithm also returns the 3D location of presenter's head based on the estimated 2D locations in the two calibrated cameras, using triangulation. The resulting estimates are compared to the ground truth, that is available every 10 frames (66 msec), for the cases where the presenter's head is visible by at least two cameras (i.e., has been labeled at two or more camera views, and thus can be obtained through triangulation). Four metrics are depicted in Table 2 for evaluating the performance of the system:

3D error: Mean Euclidean 3D distance in millimeters between the estimated and the ground truth position of the head center in 3D coordinates. In addition, "% err $<$ 300" is the percentage of time instants, where the 3D error is smaller than 300 mm.

Table 2. Head location performance by the proposed algorithm versus the background subtraction system (BGS) of IBM "PeopleVision", using the metrics discussed in Section 6

Data	Sem03		Sem04	
Method	Proposed	BGS	Proposed	BGS
3D error (mm)	253.9	278.4	467.4	480.3
3D err $<$ 300	84.6%	81.2%	78.9%	47.7%
2D err (mm)	228.3	204.7	441.1	436.9
2D err $<$ 300	85.3%	84.1%	80.7%	57.1%

2D error: Mean of Euclidean 2D distance in millimeters between the projection on the smart room floor of the estimated 3D head center and that of the 2D ground truth. Furthermore, "% err $<$ 300" is the percentage of time instants, where the 2D error is smaller than 300 mm.

In addition to the proposed system, the performance of the modified background subtraction based system (BGS) developed from the IBM "PeopleVision" system is also depicted. As it becomes clear, the two approaches achieve similar results on the CHIL seminar database test sets. Notably, the performance on the Sem04 dataset is worse. This is partially due to the more challenging nature of this set in terms of lighting and motion. For example, when the presenter moves in or outside the area near the screen, the face region skin color changes abruptly due to the projector illumination on the presenter's face. In this case, the proposed algorithm fails, due to the use of color based mean shift tracking and has to "wait" for the decision to re-initialize tracking with multiview face detection. An additional reason for the poor performance of the proposed system on the Sem04 set is that, due to lack of time, the multiview face detectors have only been trained on Sem03 data. Nevertheless, the proposed system is slightly more consistent than the BGS approach on the Sem04 set, as it achieves a higher percentage of instants where the tracking error is less than 300 mm.

7 Summary

In this paper, we proposed a novel system for joint face detection and head tracking in camera views and the three-dimensional space using multicamera input. The system combines the strengths of FloatBoost multiview face detection and mean shift tracking, with camera calibration information that is used to initialize and verify the returned results based on two camera views. We applied the algorithm on the CHIL seminar database, and we compared the system performance to that of a background subtraction based tracker. In future work, we will extend the system to deal with four cameras, and improve tracking by seeking additional cues such as joint contour and appearance information.

References

1. CHIL project web-site: `http://chil.server.de`
2. H.A. Rowley, S. Baluja, and T. Kanade, "Neural network-based face detection," *IEEE Trans. Pattern Anal. Machine Intell.*, 20(1):23–28, 1998.
3. D. Roth, M.-H. Yang and N. Ahuja, "A SNoW-based face detector," In *Proc. NIPS*, 2000.
4. P. Viola and M. Jones, "Rapid object detection using a boosted cascade of simple features," In *Proc. Conf. Computer Vision Pattern Recog.*, 2001.
5. H. Schneiderman and T. Kanade, "A statistical method for 3D object detection applied to faces and cars," In *Proc. Conf. Computer Vision Pattern Recog.*, 2000.
6. D. Comaniciu, V. Ramesh, and P. Meer, "Real-time tracking of non-rigid objects using mean shift," In *Proc. Conf. Computer Vision Pattern Recog.*, 2000.
7. M. Isard and A. Blake, "Condensation - conditional density propagation for visual tracking," *Int. J. Computer Vision*, 29(1):5–28, 1998.
8. J. Black and T. Ellis, "Multi camera image tracking," In *Proc. IEEE Work. on Performance Evaluation of Tracking and Surveillance*, 2001.
9. A. Hampapur, S. Pankanti, A.W. Senior, Y.-L. Tian, L. Brown, and R. Bolle, "Face cataloger: Multi-scale imaging for relating identity to location," In *Proc. IEEE Conf. Advanced Video Signal Based Surveillance*, pp. 13–20, 2003.
10. Z. Zhang, L. Zhu, and S. Li, "Real time multiview face detection," In *Proc. IEEE Int. Conf. Face Gesture Recog.*, 2002.
11. P. Pudil, J. Novovicova, and J. Kittler, "Floating search methods in feature selection," *Pattern Recog. Lett.*, 15:1119–1125, 1994.
12. J. Friedman, T. Hastie, and R. Tibshirani, "Additive logistic regression: A statistical view of boosting," *Technical Report*, Dept. Statistics, Stanford Univerity, Palo Alto, CA, 1998.
13. R.E. Schapire and Y. Singer, "Improved boosting algorithms using confidence-rated predictions," *J. Machine Learning*, 37(3):297–336, 1999.
14. A. Jain and D. Zongker, "Feature selection: Evaluation, application, and small sample performance," *IEEE Trans. Pattern Anal. Machine Intell.*, 19(2): 153–158, 1997.
15. P. Somol, P. Pudil, J. Novovicova, and P. Paclik, "Adaptive floating search methods in feature selection," *Pattern Recog. Lett.*, 20:1157–1163, 1999.
16. A. Bobick and J. Davis, "The representation and recognition of action using temporal templates," *IEEE Trans. Pattern Anal. Machine Intell.*, 23(3):257–267, 2001.
17. G. Welch and G. Bishop, "An introduction to the Kalman Filter," *Technical Report* TR 95-041, Computer Science Dept., Univ. of North Carolina, Chapel Hill, NC, 1995.
18. J.-Y. Bouguet, Camera calibration toolbox, `http://www.vision.caltech.edu/bouguetj/calib_doc/`.
19. D. Macho, J. Padrell, A. Abad, et al., "Automatic speech activity detection, source localization, and speech recognition on the CHIL seminar corpus," In *Proc. Int. Conf. Multimedia Expo*, 2005.
20. A. Senior, "Tracking with probabilistic appearance models," In *Proc. Int. Work. on Performance Evaluation of Tracking and Surveillance Systems*, 2002.

Perceptive User Interface, a Generic Approach*

Michael Van den Bergh[1,3], Ward Servaes[2], Geert Caenen[1],
Stefaan De Roeck[1], and Luc Van Gool[1,3]

[1] ESAT-PSI, University of Leuven, Belgium
{mvandenb,caeneng,sderoeck}@esat.kuleuven.be
[2] PHILIPS
ward.servaes@philips.com
[3] Computer Vision Group (BIWI), ETH Zuerich, Switzerland
vangool@vision.ee.ethz.ch

Abstract. This paper describes the development of a real-time perceptive user interface. Two cameras are used to detect a user's head, eyes, hand, fingers and gestures. These cues are interpreted to control a user interface on a large screen. The result is a fully functional integrated system that processes roughly 7.5 frames per second on a Pentium IV system. The calibration of this setup is carried out through a few simple and intuitive routines, making the system adaptive and accessible to non-expert users. The minimal hardware requirements are two web-cams and a computer. The paper will describe how the user is observed (head, eye, hand and finger detection, gesture recognition), the 3D geometry involved, and the calibration steps necessary to set up the system.

1 Introduction

With the abundance of multimedia information and the increasing ubiquity of computing power, the number of applications for perceptive user interfaces is growing. Examples are a security control room, browsing a multimedia database as seen in the motion picture *Minority Report*, controlling a home entertainment system, the gaming industry... Vision allows us to detect a number of interesting cues like location, identity, a user asking for attention, expression, emotion, a user pointing at objects or a user gesturing. This paper summarizes the development of a perceptive user interface that allows a user to point at objects and manipulate them with hand gestures.

There are a few existing non-commercial systems with similar objectives. One example is *PFinder*, developed by C. Wren, A. Azarbayejani and A. Pentland at the Massachusetts Institute of Technology (MIT), USA [1]. This system analyzes color and shape to detect the head and the hands of the user. A body contour is extracted by comparing the current image with a stored background image. Subsequently a number of coordinates in the image are selected with high probability of being a head or a hand. The method is very fast, but an interface as proposed in this paper requires a more detailed human model. Other systems

* This work was supported by GOA/2004/05, Research Fund K.U.Leuven and the ETH project Blue C II

N. Sebe, M.S. Lew, and T.S. Huang (Eds.): HCI/ICCV 2005, LNCS 3766, pp. 60–69, 2005.

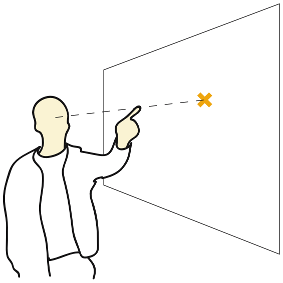


Fig. 1. User pointing at a point on screen

attempt to build a more articulated and dynamic body model, e.g. the system developed by R. Plänkers and P. Fua at the École Polytechnique Fédérale de Lausanne EPFL, Switzerland [2]. They built a body model from *soft objects*. The model is iteratively fit to the camera images. This approach yields a high level of detail, but is too complex for a real-time implementation.

In this paper we describe a system that is both fast enough to implement a real-time perceptive user interface, and sufficiently detailed to allow a user to point at objects on a screen and to recognize the user's hand gestures. The rest of this paper is organized as follows. First the user's head and hands are extracted through skin color analysis (section 2.1). Next, the user's fingers and eyes are located and the user's gestures are processed (sections 2.2-2.4). If the user is pointing, an imaginary line is constructed between the user's eyes and his finger tip (figure 1) and intersected with the screen plane (section 3). The system is designed to be robust and adaptive to new environments, which requires a number of calibration steps (section 4). Finally, we describe the demonstration GUI and the integrated setup (section 5). Considering the real-time nature of the system, all algorithms need to be selected with speed of execution in mind.

2 Observing the User

2.1 Skin Color Analysis

This section addresses the detection of skin pixels in a camera image. Skin and non-skin pixels are modeled as distributions in the *rg*-color space. Based on these distributions a MAP rule is derived for the probability of a color pixel being skin or not. The results are further improved in a post-processing step.

First the image is converted to the *rg*-color space, which is an intensity normalized color space ($r = R/(R + G + B)$ and $g = G/(R + G + B)$). The benefits of this transformation are a lighting invariant, compact representation of skin colors and a speedy computation. To model skin and non-skin pixels in this color space, a *gaussian mixture model* (GMM) is implemented similar to [3]. The result are two distributions, $p(\mathbf{c}|skin)$ and $p(\mathbf{c}|non\text{-}skin)$. The model for the probability that a given color is skin is constructed with Bayes' rule,

$$P(skin|\mathbf{c}) = \frac{p(\mathbf{c}|skin)}{p(\mathbf{c}|skin) + p(\mathbf{c}|non\text{-}skin)} \tag{1}$$

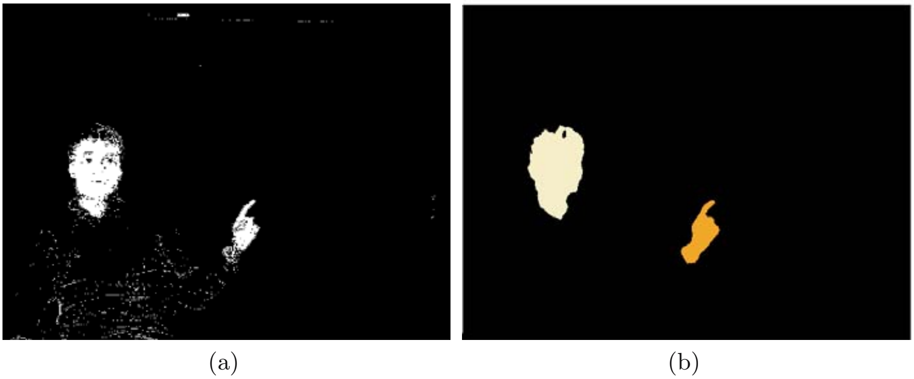


Fig. 2. (a) Result of skin color analysis without post-processing. (b) Result of skin color analysis after post-processing

where the prior $P(skin)$ is discarded for the time being. Figure 2a shows the result of this detection model, after thresholding. To speed up the detection, the results of (1) are computed offline and are stored in a *lookup table* (LUT). To further speed up the detection, it is applied to a decimated image first, and then detailed detection is applied only to a region of interest (ROI).

The resulting detection still suffers from a lot of noise and holes. To improve the quality, we take another look at the prior that was discarded in (1). It is obvious that a pixel has a higher probability of being a skin pixel if it is surrounded by skin pixels. This information can be included in the prior. [4] explains how such a prior can be modeled with median filtering, requiring very few CPU cycles. Next, a connected components analysis is applied to select only sufficiently large objects (hand, face). Through background segmentation [5] objects from the background can be discarded. The result of these post-processing steps is shown in figure (2b). Note the level of detail of the hand segmentation. This quality will be very important for the robustness of the finger detection and gesture recognition in the following subsections.

Our current implementation uses a fast area-based measure to distinguish between head and hand objects (figure 2b). Though functional, future revisions might be improved with a basic face detector and a more detailed arm model.

2.2 Finger Detection

This subsection will describe the detection of fingers on the detected hand, and the localization of the fingertip of the pointing finger. With speed in mind a simple yet very effective technique from [6] was implemented. The first step is to perform erosion on the hand object until all fingers are removed, and to dilate it back to get a good approximation of the palm of the hand. Subtracting this object from the original observed object leaves only the fingers (figure 3). Using connected component analysis we can remove noise and count the number of observed fingers. For a hand with up to three extended fingers the accuracy of

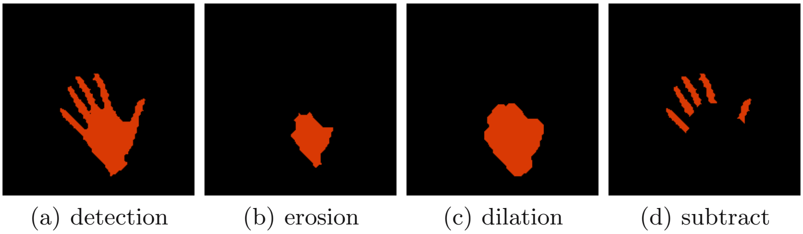


Fig. 3. Finger detection

the method exceeds 95%. This means the method is robust enough to decide whether the user is pointing at an object (one finger) or gesturing (zero or multiple fingers). If exactly one finger is counted, the finger tip will be located as the furthest point from the gravity center of the palm of the hand.

2.3 Gesture Recognition

If zero or multiple fingers are detected, gesture recognition is used to understand which action the user is trying to invoke. The starting point is a hand object as segmented in figure 3a. The object is normalized for rotation and size through eigen analysis. Next, the outline shape is extracted with a 3 × 3 edge filter. The similarity between two edge maps is measured by the Hausdorff-distance [7].

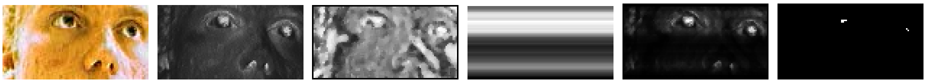
In our case the gesture recognition subsystem can use two camera images. Therefore we can train two edge maps with each gesture, one from each camera. During detection the sum of both Hausdorff-distances is minimized to achieve an effective stereo detection. If there is doubt in one camera image, this ambiguity can be resolved through the second one. The accuracy depends entirely on the chosen gestures. In our experiments we achieved 95% accuracy distinguishing 4 gestures (pointing, scissors, gun and open hand).

2.4 Eye Localization

Localization of the eyes consists of two steps. First the results of the skin color analysis are used to estimate a region of interest (ROI). Using eigen analysis an ellipse is fit to the detected face object. Given this ellipse, a small rectangle can be cut out from the upper part of the face and normalized (figure 4). The second step is to perform a precise detection only on this region. To detect the eyes a number of features are extracted and based on each feature a pseudo-probability is calculated. A combination of these probabilities leads to a more robust detection (figure 4). The first probability is constructed based on luminance, as the eyes (pupils) are distinctly darker spots in the image,

$$p_L(\mathbf{p}) = \exp\left(\frac{-I(\mathbf{p})}{s}\right) \tag{2}$$

where $I(\mathbf{p})$ is the luminance of point p and s is a scaling factor. Experiments under various lighting conditions and using different cameras have shown 100 to



(a) (b) (c) (d) (e) (f)

Fig. 4. (a) region of interest, (b) probability based on luminance (p_L), (c) probability based on color (p_S), (d) probability based on integrated luminance (p_I), (e) product of all probabilities, (f) detected eye components

be an appropriate value for s. The second probability is based on color, as the eyes are distinctly not skin-colored,

$$p_S(\mathbf{p}) = p(\textit{non-skin}|c_{\mathbf{p}}) \tag{3}$$

Finally a third probability can be extracted from the horizontally integrated luminance. Both the eyes and the shadow region around them present a much area region inside the ROI. These differences can be enhanced through horizontal integration,

$$S_I(\mathbf{p}) = \int_{-w}^{+w} I(t, y)\, dt \tag{4}$$

where p_I is obtained as a normalized version of S_I. To improve the detection an additional probability p_R is introduced to eliminate non-skin colored objects at the sides of the ROI (e.g. hairs, ears). Combining the probabilities yields,

$$p(\mathbf{p}) = p_L(\mathbf{p}) \cdot p_S(\mathbf{p}) \cdot p_I(\mathbf{p}) \cdot p_R(\mathbf{p}) \tag{5}$$

Finally, the eyes are located using connected component analysis, looking for the two components with highest probability. Unlikely detections are discarded after two geometric tests. The distance between the eyes needs to be likely with respect to the size of the face, and the eyes should be horizontal with respect to the rotation of the face. The method's accuracy exceeds 95% during frontal observation. It remains robust with respect to head rotations around the vertical axis until approximately 30° of rotation.

3 3D Extraction

The goal is to figure out which point on the screen the user is pointing at, which should be robust and unambiguous. We chose to draw an imaginary line from the user's eyes, through his fingertip, onto the screen (figure 1). In what follows the 3D coordinates of the eyes and fingertip are written as $\mathbf{X}^o$ and $\mathbf{X}^v$ respectively. The pointing direction can thus be written as $\mathbf{X}^v - \mathbf{X}^o$. The camera matrices P and P' and the screen plane $\mathbf{U}$ are obtained in a calibration step which is described in section 4.

First, we want to translate the couples of 2D coordinates in the camera images into 3D coordinates in space. The relation between a 2D point $\mathbf{x}$ and corresponding 3D point $\mathbf{X}$ in homogeneous coordinates is [8]:

$$\mathbf{x} = P\mathbf{X} \tag{6}$$

Applying (6) to both cameras (P and P' respectively) we can construct a homogeneous system of four equations,

$$\begin{bmatrix} wP_1 - xP_3 \\ wP_1 - yP_2 \\ w'P_1' - x'P_3' \\ w'P_1' - y'P_2' \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ W \end{bmatrix} = \mathbf{0} \tag{7}$$

which we can use to calculate the points $\mathbf{X}^o$ and $\mathbf{X}^v$. Finally, the intersection of the line through these two points and the screen plane needs to be calculated. The equation of this line is

$$L \leftrightarrow \mathbf{X} = \mathbf{X}^o + k(\mathbf{X}^v - \mathbf{X}^o) \tag{8}$$

Together with the screen plane (11) we can find

$$\begin{bmatrix} k \\ x \\ y \end{bmatrix} = \begin{bmatrix} \mathbf{X}^v - \mathbf{X}^o & -\mathbf{U}_x & -\mathbf{U}_y \end{bmatrix}^{-1} (\mathbf{U}_0 - \mathbf{X}^o) \tag{9}$$

This equation can be used to compute the 2D pixel coordinate (x, y) on screen directly from the original 2D eye and finger coordinates.

4 Calibration

A number of calibration steps are necessary to get the system running. These steps are designed to be simple and fully automatic, as described in the following subsections.

4.1 Camera Calibration

The *internal camera parameters* can be assumed to be constant for fixed camera settings. The parameters are calibrated once (off line) using a semi-automatic calibration program with calibration object [9, 10]. The internal camera parameters are stored in the calibration matrices K and K' for both cameras respectively. The calibration of the *external camera parameters* (position, viewing direction) should be easy enough to be performed by the end user. To achieve this the calibration only requires the user to point his finger randomly in front of the cameras. The calibration data is recorded by registering correspondences in both camera images as the system tracks the user's fingertip. The relation between a set of correspondences $\mathbf{x}$ en $\mathbf{x}'$ and the camera pair's fundamental matrix F in homogeneous coordinates can be written as

$$\mathbf{x}'^t F\mathbf{x} = 0 \tag{10}$$

F can be determined using several correspondences. A robust method to estimate F is the Ransac method [11]. The camera matrices P and P' can then be derived from F via the essential matrix E and the calibration matrices K and K' as described in [12].

4.2 Screen Calibration

To calibrate the position of the screen, the system will display a sequence of points on the screen one by one, and ask the user to point his finger at them as they appear. Note that this procedure can also be used for the calibration in section 4.1. The 2D coordinates of the displayed points are denoted as (λ_i, μ_i). The 3D positions of the corresponding finger and eye positions are denoted as $\mathbf{X}_i^v$ and $\mathbf{X}_i^o$ respectively. The screen plane α is described by,

$$\alpha \leftrightarrow \mathbf{X} = \mathbf{U}_0 + x(\mathbf{U}_x) + y(\mathbf{U}_y) \tag{11}$$

where $\mathbf{U}_0$ is the upper left corner of the screen and $\mathbf{U}_x$ and $\mathbf{U}_y$ are the horizontal and vertical axes of the screen. The line projected through the 3D finger and eye coordinates is denoted as L_i,

$$L_i \leftrightarrow A_i\mathbf{X} + \mathbf{b}_i = \mathbf{0} \tag{12}$$

The rows of A_i form a basis for the orthogonal complement of the subspaces reproduced by the vector $\mathbf{X}_i^v - \mathbf{X}_i^o$. If an orthonormal basis is chosen, the following equation describes the perpendicular distance of any $\mathbf{X}$ to L_i,

$$d^2(\mathbf{X}, L_i) = (A_i\mathbf{X} + \mathbf{b}_i)^t(A_i\mathbf{X} + \mathbf{b}_i) \tag{13}$$

Every projected point (λ_i, μ_i) in the screen plane corresponds with a line L_i. In order for this point to be equal to the point the user is pointing at, L_i should intersect the screen plane α in that point. Thus, for every i,

$$A_i\left(\mathbf{U}_0 + \lambda_i(\mathbf{U}_x) + \mu_i(\mathbf{U}_y)\right) + \mathbf{b}_i = \mathbf{0} \tag{14}$$

Given a number of correspondences $(\lambda_i, \mu_i) \leftrightarrow L_i$, it is possible to determine α. In order to deal with unavoidable measurement errors, a least squares solution will be computed based on N correspondences. Using (13) we look for the plane α that puts the screen points $\mathbf{u}(\lambda_i, \mu_i) \in \alpha$ as closely as possible to their corresponding L_i.

$$\sum_i (A_i\mathbf{u}(\lambda_i, \mu_i) + \mathbf{b}_i)^t(A_i\mathbf{u}(\lambda_i, \mu_i) + \mathbf{b}_i) \tag{15}$$

which can be rewritten,

$$\sum_i \left(A_i\Lambda_i\mathbf{u} - \mathbf{b}_i\right)^t \left(A_i\Lambda_i\mathbf{u} - \mathbf{b}_i\right) \text{ where } \mathbf{u} = \begin{bmatrix} \mathbf{U}_0 \\ \mathbf{U}_x \\ \mathbf{U}_y \end{bmatrix} \text{ and } \Lambda_i = \left[\, I_3 \middle| \lambda_i I_3 \middle| \mu_i I_3 \,\right] \tag{16}$$

Minimization with respect to $\mathbf{u}$ yields,

$$\underbrace{\sum_i (A_i\Lambda_i)^t(A_i\Lambda_i)}_{S}\mathbf{u} + \underbrace{\sum_i (A_i\Lambda_i)^t\mathbf{b}_i}_{\mathbf{g}} = \mathbf{0} \Leftrightarrow S\mathbf{u} = -\mathbf{g} \tag{17}$$

The solution to this equation is not unique. The screen can still be shifted away from the user. This leaves one free parameter for the screen plane. As we only need to determine the 2D screen coordinates based on the 3D finger and eye coordinates (9), this free parameter is irrelevant and can be discarded.

4.3 Color Calibration

The quality of the skin color analysis in section 2.1 depends a great deal on the white balance parameters of the cameras and the chosen threshold for (1). To improve the robustness and usability of the system we have developed an automatic system to adapt these parameters to the given camera and lighting conditions. We present a simple setup where the system imitates a mirror with a small rectangle overlaid in the middle of the picture. The user needs to sit and position his head exactly inside this rectangle before calibration begins. The following measure of skin color analysis quality is proposed,

$$q = n_+ - n_- \tag{18}$$

where n_+ denotes the number correctly classified pixels, the number of skin pixels inside the rectangle and the number of non-skin pixels outside, and n_- the number of skin pixels detected outside of the rectangle. The white balance is varied until a maximum is found for q. Given this white balance, the same quality measure is used to determine the optimal threshold. In most cases the color calibration will yield a natural white balance and a neutral threshold. Only with very strange backgrounds or lighting conditions the calibration will result in a compensating white balance.

5 Integrated System

5.1 Graphical User Interface

Until now we have described methods to detect skin, eyes, hands, fingers and gestures. Based on the coordinates of the eyes and fingertip we can determine which point on the screen the user is pointing at. Now it's important to illustrate the functionality in a clear and useful interface (figure 5). The goal is to build an environment where the user can point at objects (icons) to highlight them, grab them, drag them and activate them. Each icon has three states: normal, highlighted and grabbed. When the icons appear on the screen they are in the normal state. When the user points at an icon it will grow and become highlighted. The user can now gesture to grab the icon. The icon will go to the grabbed state and its highlight will change color. Finally the user can gesture to activate the icon. In our implementation activating an icon opens a hyperlink to a new interface with new icons. All gestures are trained during a training procedure in advance and they can be chosen freely by the user. The only gesture that's fixed is the pointing gesture, where the user only extends one finger. The icons, their location on screen and their hyperlinks are defined in an XML file.

5.2 Integrated Setup

Our test setup consists of a large screen (3 m by 2 m) with two Sony DFW-VL500 cameras placed alongside it. The cameras are positioned near the border of the

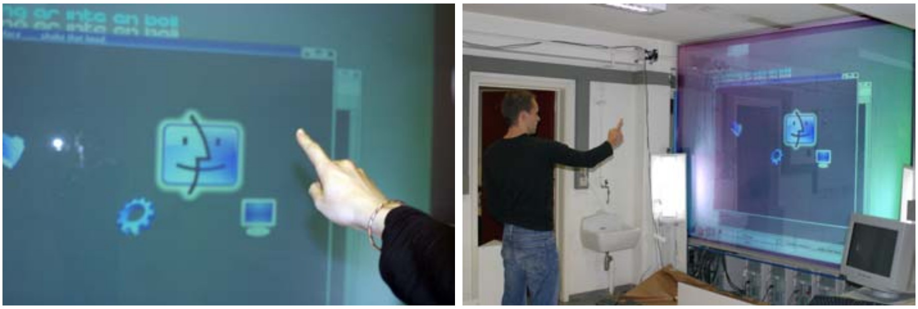


Fig. 5. Using the interface

screen, in the middle of the vertical sides. The cameras are pointed directly at the user. The cameras could in fact be positioned anywhere, as long as the eyes and the hands are visible inside the picture. Both cameras continuously grab images. These images are passed through the skin color analysis. The grabbed images and their binary skin images are fed to the subsystems described in section 2. Depending on the state (pointing or gesturing) the system applies finger tip detection or gesture recognition. The gesture recognition system analyzes the stereo images and returns the ID of the recognized gesture and the hand's center of gravity. The eye detector locates the coordinates of both eyes and returns the middle point. Based on two coordinates (either fingertip and eyes, or hand center and eyes) an imaginary 3D line is constructed. The intersection of this line with the 3D screen plane gives us the 2D coordinate on the screen. The interface processes this 2D screen coordinate along with the ID of the recognized gesture. If the user points at an icon this icon will be highlighted. The gestures are interpreted to perform actions on the highlighted icon.

The result of the integrated setup is illustrated in figure 5. The figure shows a user actively demoing the user interface. Our system is implemented in C++ and achieves approximately 7.5 frames per second on a Pentium IV. The average pixel deviation is 15 pixels on a resolution of 1024x768, which boils down to a pixel deviation of about 2%. The icons in our system are 168x168 pixels in size, comfortably bigger than the pixel error.

6 Conclusion

We have selected several state-of-the-art techniques, keeping the speed of execution in mind. The skin color analysis was improved drastically with a fast post processing step. We implemented a robust stereo gesture recognition system. A fast and efficient eye detection system was implemented. We proposed a set of simple and intuitive calibration routines to calibrate the cameras, the screen and the skin color analysis, making the system adaptive and accessible to non-expert users. Each component was tested for quality and robustness by setting a ground truth and comparing the detection results. As described in the previous sections each component exceeded a 95% detection accuracy in our experiments.

Finally we combined all these components to build a working prototype. Several movies demonstrating the work presented in this paper can be downloaded at `http://www.esat.kuleuven.ac.be/~mvandenb`. The functionality of the various subsystems from section 2 are demonstrated in these movies. The final movie shows the working prototype and is a clear proof of concept of our perceptive user interface.

References

1. Wren, C.R., Azerbayejani, A., Darell, T., Pentland, A.P.: Pfinder: Real-time tracking of the human body. IEEE Transactions on Pattern Analysis and Machine Intelligence **19** (1997) 780–785
2. Plänkers, R., Fua, P.: Articulated soft objects for video-based body modeling. Proceedings 8th International Conference on Computer Vision (2001)
3. Jones, M.J., Rehg, J.M.: Statistical color models with application to skin detection. Cambridge Research Laboratory Technical Report Series CRL98/11 (1998)
4. Jedynak, B., Zheng, H., Daoudi, M., Barret, D.: Maximum entropy models for skin detection. Technical Report publication IRMA **57** (2002)
5. Mester, R., Aach, T., Dümbgen, L.: Illumination-invariant change detection using a statistical colinearity criterion. Pattern Recognition: Proceedings 23rd DAGM Symposium (2001) 170–177
6. Hung, Y.P., Yang, Y.S., Chen, Y.S., Hsieh, I.B., Fuh, C.S.: Free-hand pointer by use of an active stereo vision system. Proceedings of Third Asian Conference on Computer Vision **1** (1998) 632–639
7. Sánchez-Nielsen, E., Antón-Canalís, L., Hernández-Tejera, M.: Hand gesture recognition for human-machine interaction. Journal of WSCG **12** (2004)
8. Gool, L.V.: Beeldinterpretatie en Computer Vision II. Visics, KULeuven (2002-2003)
9. Koninckx, T., Griesser, A., Van Gool, L.: Real-time range scanning of deformable surfaces by adaptively coded structured light. Fourth International Conference on 3-D Digital Imaging and Modeling (3DIM03), S. Kawada, ed. (2003) 293–302
10. Koninckx, T., Van Gool, L.: High-speed active 3d acquisition based on a pattern-specific mesh. SPIE's 15th annual symposium on electronic imaging - videometrics VII **5013** (2003) 26–37
11. Fischler, M., Bolles, R.: Random sampling consensus: a paradigm for model fitting with application to image analysis and automated cartography. Commun. Assoc. Comp. Mach. **24** (1981) 381–395
12. Hartley, R.I., Zisserman, A.: Multiple view geometry in computer vision. (2000) 239–240

A Vision Based Game Control Method

Peng Lu, Yufeng Chen, Xiangyong Zeng, and Yangsheng Wang

Institute of Automation, Chinese Academy of Sciences,
Beijing 100080, China
luchenpeng@msn.com

Abstract. The appeal of computer games may be enhanced by vision-based user inputs. The high speed and low cost requirements for near-term, mass-market game applications make system design challenging. In this paper we propose a vision based 3D racing car game controlling method, which analyzes two fists positions of the player in video stream from the camera to get the direction commands of the racing car.

This paper especially focuses on the robust and real-time Bayesian network (BN) based multi-cue fusion fist tracking method. Firstly, a new strategy, which employs the latest work in face recognition, is used to create accurate color model of the fist automatically. Secondly, color cue and motion cue are used to generate the possible position of the fist. Then, the posterior probability of each possible position is evaluated by BN, which fuses color cue and appearance cue. Finally, the fist position is approximated by the hypothesis that maximizes a posterior. Based on the proposed control system, a racing car game, "Simulation Drive", has been developed by our group. Through the game an entirely new experience can be obtained by the player.

1 Introduction

Recent advances in various signal-processing technologies, coupled with an explosion in the available computing power, have given rise to a number of novel human computer interface (HCI) modalities: speech, vision-based gesture recognition, etc.

Up to now many vision based interfaces (VBI) have been proposed. But the systems [1] [2], which are designed for game control, are very few. For improving the robust of algorithm most video recognition algorithms are time-consuming, it isn't fit for game control, which requires fast response.

How to find a robust and real time fist tracking algorithm is the key problem of our vision based game control method. Currently, many researches have been done on this area. However tracking objects efficiently and robustly in complex environment is still a challenging issue in computer vision. Particle filter [3] [4] and mean shift[5] [6]are two successful approaches taken in the pursuit of robust tracking. Particle filters, to apply a recursive Bayesian filter based on propagation of sample set over time, maintain multiple hypotheses at the same time and use a stochastic motion model to predict the position of the object. Maintaining multiple hypotheses allows the tracker to handle clutters in the

N. Sebe, M.S. Lew, and T.S. Huang (Eds.): HCI/ICCV 2005, LNCS 3766, pp. 70–78, 2005.

background, and recover from failure or temporary distraction. However, there are high computational demands in the approach, and this is the bottleneck to apply particle filtering in real time systems. On the other hand, mean shift explores the local energy landscape, using only a single hypothesis. This approach is computational effective, but it is susceptible to converge to local maximum.

In this paper we propose a real time and robust fist tracking algorithm. Comparing with conventional algorithms, our algorithm has three advantages. First, the proposed algorithm uses more hypotheses to overcome the shortcoming of local optimal of the mean shift algorithm. Second, in order to reduce the computation, motion cue and color cue are used to reduce the number of hypotheses. Third, BN is used to fuse the appearance cue and color cue, which makes the tracking results more robust.

Based on the proposed algorithm, we design a vision based racing car game control system, which analyzes two fists positions of the player in video stream from the camera to get the direction commands of the racing car. In the traditional keyboard control approach, driving car is an integral process. That is to say the larger angle the player wants to turn, the longer time the player needs to push the key down. In our game, if the player wants to change the direction at a large scale, what he should do is just pose his fists at a large angle. That is to say, for a normal camera this process just needs 33 millisecond. Experimental results show that the frame rate of a normal camera, which is about 30 frames per second, is enough for controlling the car.

The remainder of this paper is organized as follows: Section 2 describes the novel fist tracking algorithm. section 3 describe our 3D game control system. Some experiments are shown in section 4.

2 BN Based Fist Tracking Method

This section explains how to track the player's fists. It's very important for our racing car control system to efficiently and robustly tracking objects in complex environment. In other words, the response time of the vision interface should be less than a video frame time. And at the same time, the recognition algorithm should be more robust. The proposed tracking algorithm includes three steps. First, some hypotheses about the fist's position are generated based on the motion cue and color cue. In this way, the number of the hypotheses is very limited. Then all the hypotheses are evaluated by the BN, which fuses appearance cue and color cue. Finally, the fist position is approximated by the hypothesis that maximizes a posterior.

2.1 Hypothesis Generation

Color Cue. Skin is arguably the most widely used primitive in human image processing research, with applications ranging from face detection and person tracking to pornography filtering. Color is the most obvious feature of the fist. It indicates the possible position of the fist. In order to use color cue of the fist ,a probability distribution image of the desired color must be created firstly. Many algorithms employ a manual process to extract color information in their initialization stage, such as CAMSHIFT. By this way an accurate skin color model can be obtained. But semi-automation is the main shortcoming of these algorithms. For achieving automation, the most popular method is to learn the skin color distribution from a large number of training samples. However, due to the different illumination and different camera lens, this color distribution is not always exact in real condition.

In this paper, we proposed a novel scheme to acquire the color distribution of the fist. Generally speaking, the skin color of hand and face, which belong to the same person, are the same or similar. For face detection, there are many successful algorithms. So we gain the skin distribution from the face of the player instead of the fist. In our scheme, three steps are needed for creating color model. The first stage is face detection. In this stage, our Haar-Sobel-like boosting [7] algorithm is used. Haar and sobel features are used as feature space, and GentleBoost is used to select simple classifiers. Haar features are used to train the first fifteen stages. And then sobel features are used to train the rest fourteen stages. The second stage is face alignment. At this stage, active shape model (ASM) [8] is used. The last stage is creating color model. The hues derived from the pixels of the face region are sampled and binned into an 1D histogram, which is used as the color model of the fist. Through this histogram, the input image from the camera can be convert to a probability image of the fist.

Motion Cue. In order to deal with the skin-colored objects in the background, motion cue is used for our algorithm. We differentiate the current frame with the previous frame to generate the difference image using the motion analysis method in [9]. The method is to compute the absolute value of the differences in the neighborhood surrounding each pixel. When the accumulated difference is above a predetermined threshold, the pixel is assigned to the moving region.

Since we are interested in the motion of skin-colored regions, the logical AND operator is applied between the color probability distribution image and the difference image. And as a result the probability distribution image is obtained.

Hypothesis Generation. Suppose human hand is represented by a rectangle window, the possible position of the fist is gained as follows:

1, we sample the image from 320x240 to 160x120.

2, a subwindow x, y, w, h, where x and y are the left-top coordinate, w and h are the size of the rectangle, moves on the probability distribution image. And the sum of the pixels in subwindow is calculated. In our experiment, the w is 28 and the h is 35.

3, if the sum is below a certain threshold T, which is in a direct ratio to $w \times h$, it returns to step 2. And if the sum is above the threshold, the CAMSHIFT [10] is applied to getting the local maximum and the local color region size. And the center point of CAMSHIFT region is saved.

4, The pixels in CAMSHIFT region are set to zero. And it goes to step 2.

Based on these saved points, multiple hypotheses of the position and size of the fist are generated. For each saved point (x, y), four rectangle regions are taken out as four hypotheses. These regions have the same center (x, y), but differ in size. So the total number of the hypotheses is $Num \times 4$, where Num is the number of saved point.

2.2 Inference in the Bayesian Network

In this section, we mainly discuss how to compute the posterior probability of each hypotheses.

For robust tracking, we must eliminate the effect of other skin-color objects, which are also in motion, such as face. So besides skin-color feature more features should be taken into consideration. Fist appearance is the most significative feature and it can be used for differentiating fist from other objects. In our algorithm, two main features, color feature and appearance feature, are employed. Bayesian network [12] is useful when we are trying to fuse more cue for tracking fist. Thus the posterior probability of fist region given observations of the other variables can be computed as follows:

$$P(X_k|C, A, X_{k-1}, X_{k-2}) \tag{1}$$

where C denotes color cue, A denotes appearance cue, X_{k-1} and X_{k-2} are the previous object state, X_k is the current object state.

The Bayesian network is shown in Fig.1.

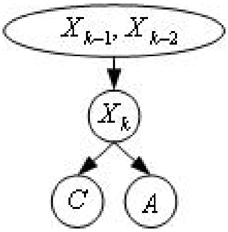


Fig. 1. The Bayesian network

By using conditional independence relationships we can get

$$\begin{aligned} &P(X_k|C, A, X_{k-1}, X_{k-2}) \\ &\propto P(X_k, C, A, X_{k-1}, X_{k-2}) \\ &\propto P(C|X_k)P(A|X_k)P(X_k|X_{k-1}, X_{k-2}) \end{aligned} \tag{2}$$

Prior Model. The prior model $P(X_{k-1}, X_{k-2})$ is derived from the dynamics of object motion, which is modelled as a simple second order autoregressive process(ARP).

$$X_k - X_{k-1} = X_{k-1} - X_{k-2} + W_k \tag{3}$$

where W_k is a zero-mean Gaussian stochastic component.

The parameters of ARP model are learned from a set of pre-labeled training sequences.

Computation of the Color Marginal Likelihood. We define the color likelihood as follows:

$$P(C|X) = \frac{1}{n_c} \sum_{i,j} P_c(i,j) \tag{4}$$

where n_c is the scale of the likelihood, and $P_c(i,j)$ is the pixel in color cue image.

Computation of the Appearance Marginal Likelihood. Based on assumption of a Gaussian distribution, the probability of input pattern A, which belongs to fist class X can be modelled by a multidimensional Gaussian probability density function:

$$P(A|X) = \frac{exp[-\frac{1}{2}(A-\mu)^T \Sigma^{-1}(A-\mu)]}{(2\pi)^{N/2}|\Sigma|^{1/2}} \tag{5}$$

where μ is the mean vector of class X, Σ is the covariance matrix of class X.

By using PCA to reduce the dimension of X, the $P(A|X)$ is approximately estimated by equation (6), more detail about equation (6) can be found in [11].

$$\hat{P}(A|X) = exp\left[-\frac{1}{2}\sum_{i=1}^{M}\frac{y_i^2}{\lambda_i}\right] exp\left[-\frac{\epsilon^2(x)}{2\rho}\right] \tag{6}$$

where $\hat{P}(A|X)$ is the estimation value of $P(A|X)$, $\epsilon^2(x)$ is the residual error, λ_i is eigenvalue of Σ, M is the dimensional of principal subspace, N is the dimension of total subspace.

In our experiment, the appearance parameters μ and Σ are learned from more than 5000 labelled images, which are collected from one hundred persons with three kinds of illumination. Some samples are shown in the Fig.2.

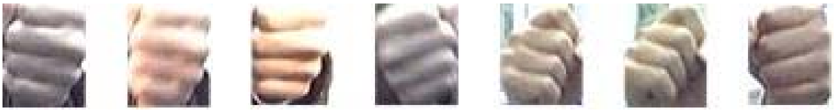


Fig. 2. Some samples used for training

3 Racing Car Game Control System

Based on the proposed algorithm, we design a vision based racing car game control system, as show in Fig. 3. It is made up of two parts: vision based interface, which is used for car's direction controlling, and voice based interface,which is used for car's status controlling (such as startup, acceleration or deceleration).

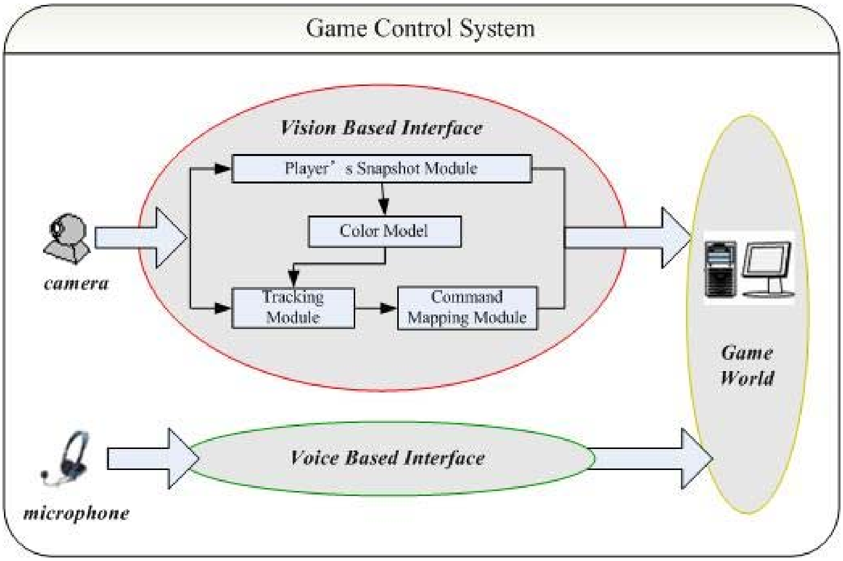


Fig. 3. Game control system

For the purpose of this paper, we mainly focus on the vision based interface and the fist tracking algorithm. Vision based interface is made up of three modules: snapshot module, tracking module and command mapping module. When a new player enters into the game, a snapshot of the player is cropped from the video frame by snapshot module automatically. And this snapshot is used as an individual player ID in our racing car game. Then tracking module starts tracking the player's fists. Finally the tracking results will be changed into game command by command mapping module to control the game. In this way the player can control the car's direction by his fists.

3.1 Snapshot Module

In order to add more individual to our system, we design a snapshot module, which can create an individual ID of the player. At the same time, another purpose of this module is to create an accurate skin color model, which will be used by the proposed fist tracking, from the player's face without an initializing by the player. In this module the player's face can be got and the player's skin color model can be created. More detail about the algorithm of this module can be found in section 2.1.

3.2 Tracking Module

Tracking module is the core of the VBI, and it is used to analyze the fist position of the player. In this module the proposed fist tracking method is implemented.

3.3 Command Mapping Module

Through tracking the fists' position can be obtained and the steering wheel direction can be computed as follows:

$$corner = atan((y_r - y_l)/(x_r - x_l)) \tag{7}$$

where $corner$ is the direction of the car, (x_l, y_l) is the position of the left hand and (x_r, y_r) is the position of the right hand.

Smooth control is very important to well designed game, so the recognition result can't be used directly. A four order smooth process is applied by our system.

$$\Theta = \omega_0 \times corner_k + \omega_1 \times corner_{k-1} + \omega_2 \times corner_{k-2} + \omega_3 \times corner_{k-3} \tag{8}$$

where Θ is the final output control information at time k, $corner_k$ is the recognition result at time k, $corner_{k-1}, corner_{k-2}, corner_{k-3}$ correspond to recognition results at time $k-1, k-2, k-3$ respectively.

In our experiment $\omega_0 = 0.5$, $\omega_1 = 0.3$, $\omega_2 = 0.15$, $\omega_3 = 0.05$.

4 Experimental Result

4.1 Algorithm Performance

In order to compare with the mean shift algorithm, two experiments are done. In the first experiment, the color-motion based mean shift algorithm is used. When player's fist and face overlap, the tracker loses the fist. Some key frames of this experiment are shown in Fig.4. And it has no chance to recover. This is because the mean shift is a local optimal algorithm. In the second experiment, the proposed algorithm is applied. Based on multi-hypotheses it overcomes the local optimal. And using more cues, our algorithm becomes more robust. Some key frames of this experiment are shown in Fig.5. All the experiments are done on a P4 1.7G machine with $512M$ memory. The normal recognition speed of our algorithm is about 29 fps.

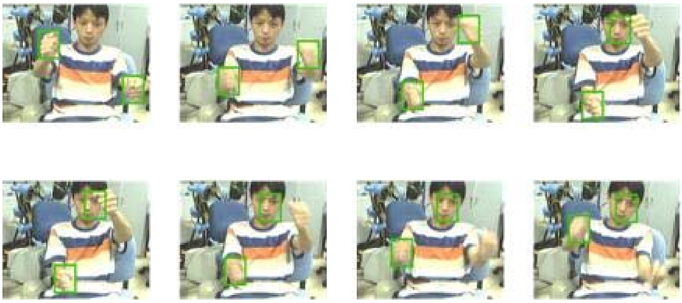


Fig. 4. Tracking failure by color and motion based Mean Shift algorithm

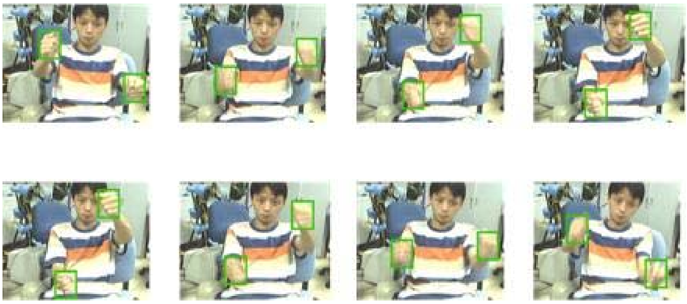


Fig. 5. Tracking by Bayesian network

4.2 3D Game Example

In the racing car game, an extra thread is created for vision based interface and the average fps is 28. That is to say 28 direction commands can be sent to the game and the experiment results show that these are enough for controlling the car's direction in real time. There are no distinct drop in game's fps. Some pictures of our game ,"Simulation Drive" are shown in Fig.6.

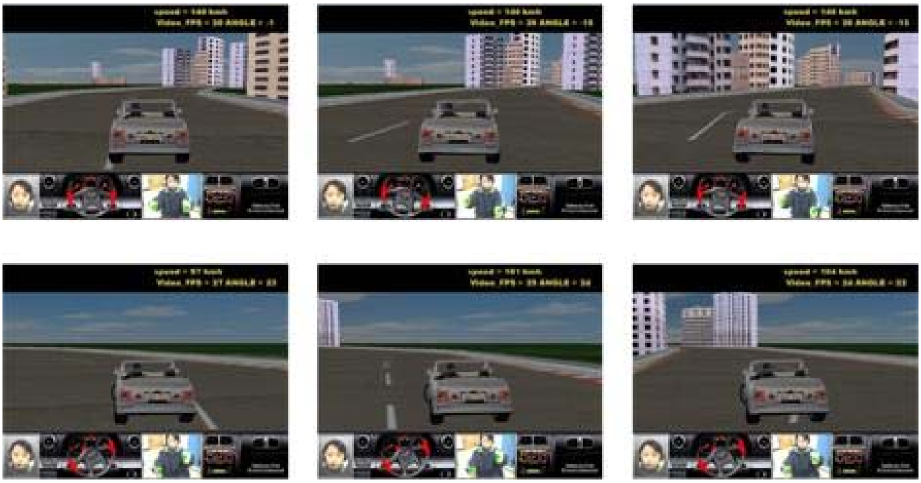


Fig. 6. Some frames of our game

5 Summarize

In this paper,a Bayesian network based fist tracking algorithm is introduced. Comparing with particle filter, the proposed algorithm, which uses more information to generate hypotheses, reduces significantly the number of hypotheses needed for robust tracking. And at the same time it overcomes the shortcoming of local optimal of the mean shift algorithm. Based on this algorithm we design a vision based 3D racing car game control system, which migrate the "natural" means to the game control.

References

1. W.T. Freeman, K. Tanaka, J. Ohta, "Computer vision for computer games", Proceedings of the Second International Conference on Automatic Face and Gesture Recognition, pp.100-105,1996.
2. Y. Akazawa, Y. Okada, "Video Based Motion Capture System as Intuitive Interface for interactive 3D Games", International Journal of Intelligent Games and Simulation (IJIGS), Vol.2, No.2, pp. 64-71, November 2003.
3. S. Maskell and N.Gordon, "A Tutorial on Particle Filters for On-line Nonlinear/Non-Gaussian Baysian Tracking", in Proc. IEE Workshop "Target Tracking: Algorithms and Applications", Oct. 2001
4. C.F. Shan, Y.C Wei, T.N. Tan, "Real Time Hand Tracking by Combining Particle Filtering and Mean Shift", The 6th International Conference on Automatic Face and Gesture Recognition (FG2004).
5. D. Comaniciu and V. Ramesh, "Mean Shift and Optimal Prediction for Efficient Object Tracking", ICIP, Vancouver, Canada, 2000, pp. 70-73.
6. D. Comaniciu, P. Meer, "Mean Shift Analysis and Applications", Proc. Seventh Int'l Conf. Computer Vision, pp. 1197-1203, Sept. 1999.
7. P.Lu, X.S.Huang, Y.S.Wang,"A New Framework for Handfree Navigation in 3D Game," Proceedings of the International Conference on CGIV04.
8. T.F. Cootes, C.J.Taylor "Statistical Models of Appearance for computer vision," Imaging Science and Biomedical Engineering, University of Manchester, Manchester M13 9PT, U.K. March 8, 2004.
9. H. P. Graf, E. Cosatto, D. Gibbon, M. Kocheisen, and E. Petajan, "Multi-Modal System for Locating Heads and Faces", AFG, Killington, Vt, 1996, pp. 88-93.
10. G. R. Bradski, "Computer Vision Face Tracking For Use in a Perceptual User Interface", Intel Technology Journal Q2, 1998.
11. B. Moghaddam and A. Pentland, "Probabilistic visual learning for object representation", IEEE Transactions on Pattern Analysis and Machine Intelligence, 19(7):696-710, July 1997.
12. D. Heckerman, "A Tutorial on Learning With Bayesian Networks", Microsoft Research Technical Report,MSR-TR-95-06

Mobile Camera-Based User Interaction

Antonio Haro, Koichi Mori, Tolga Capin, and Stephen Wilkinson

Nokia Research Center, 6000 Connection Drive, Irving, TX 75039, USA
{antonio.haro,koichi.mori,tolga.capin,stephen.wilkinson}@nokia.com

Abstract. We present an approach for facilitating user interaction on mobile devices, focusing on camera-enabled mobile phones. A user interacts with an application by moving their device. An on-board camera is used to capture incoming video and the scrolling direction and magnitude are estimated using a feature-based tracking algorithm. The direction is used as the scroll direction in the application, and the magnitude is used to set the zoom level. The camera is treated as a pointing device and zoom level control in applications. Our approach generates mouse events, so any application that is mouse-driven can make use of this technique.

1 Introduction

Mobile devices currently support navigation through a joypad/direction keys or scroll bars on touch-sensitive screens using a stylus-based pen. Although these modes of interaction are sufficient for small sized content, more intuitive techniques are required for navigating more complex data. It is difficult to use these techniques to navigate a full-sized Web page or to select an item from dozens of choices in a list box of messages, photos, audio files, phonebook entries or other mobile content. On devices with larger form-factors, additional keys provide a better user experience since keys can be dedicated to specific tasks such as page up/down and zoom level. Smart phones cannot make use of such keys due to limited physical space. Stylus-based interaction for navigation is an alternative, but requires two-handed interaction and has been shown to cause additional attentional overhead in users [1]. Consequently, alternative interaction techniques are desired. Other sensors could be added to mobile devices such as accelerometers (*e.g.,* Samsung's SCH-S310 smartphone), but these can be difficult to integrate into existing consumer-level devices at both the software and hardware level. In addition, such sensors are known to have error buildup over time since some infinitesimal acceleration is always measured.

To address these problems, we propose using the camera sensor as the input device. Feature-based tracking of the incoming video is used to estimate both motion direction and magnitude. The direction estimates are used for scrolling while the magnitude of the physical movement can drive the current zoom level in an application or be used for shake detection. This approach provides a more natural user interaction maximizing the use of the display, minimizing attentional overhead to the user, and permitting one-handed interaction. This approach does not preclude the use of a joypad, and can be used as an extension

N. Sebe, M.S. Lew, and T.S. Huang (Eds.): HCI/ICCV 2005, LNCS 3766, pp. 79–89, 2005.

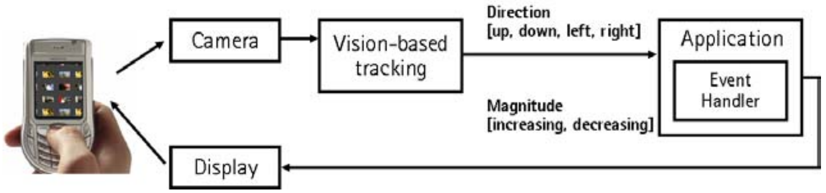

Camera
Vision-based tracking
Direction [up, down, left, right]
Magnitude [increasing, decreasing]
Application
Event Handler
Display

Fig. 1. A tracking algorithm is used to determine movement direction and magnitude. These are then treated as mouse events by the application's event handler, and the user's view is updated

of joypad-based interaction, where the joypad could be used for fine-grained selection. We tested our approach on an image-browsing task in a photo browsing application, as well as in a document viewer and in games. In informal tests, users preferred our solution to a joypad-based navigation. Joypads and scroll buttons are adequate for navigation of small datasets on limited sized displays, but not for large or complex data.

2 Related Work

Mobile camera-based tracking has been researched by several groups. Rohs *et al.* [2] perform tracking based on dividing incoming camera frames into blocks and then determine how the blocks move given a set of discrete possible translations and rotations. Our algorithm is instead based on tracking individual corner-like features observed in the entire incoming camera frames. This allows our tracker to recognize sudden camera movements of arbitrary size, as long as at least some of the features from the previous frame are still visible, at the trade-off of not detecting rotations.

Augmented reality research on mobile camera-based tracking systems includes that of Möhring *et al.* [3], who track a color-coded 3D marker to estimate 3D camera pose, after an initial calibration step. Our work is instead focused on new user interfaces using computed 2D motion, so we do not require markers or user calibration for tracking. Drab *et al.* [4] present a computationally inexpensive tracking system, however their system has problems with repeating textures and requires scenes with high dynamic range which ours does not. Beier *et al.* [5] present a marker-less tracking algorithm, however it does not run on mobile devices as ours does and also requires matching with known 3D models, which we do not require.

Additional related work includes the Mosquito game available for the Siemens SX1 mobile phone, among others that have been created since then for many smart phone platforms. While camera motion is indeed estimated in these games to translate sprites accordingly, it should be noted that the detected motion does not need to be exact as the sprites are rendered on top of the video but not attached to any feature. As such, only approximate motion is required. Since our tracked motion needs to match the user's physical motion exactly, a higher degree of accuracy is required which from our testing is not present in current

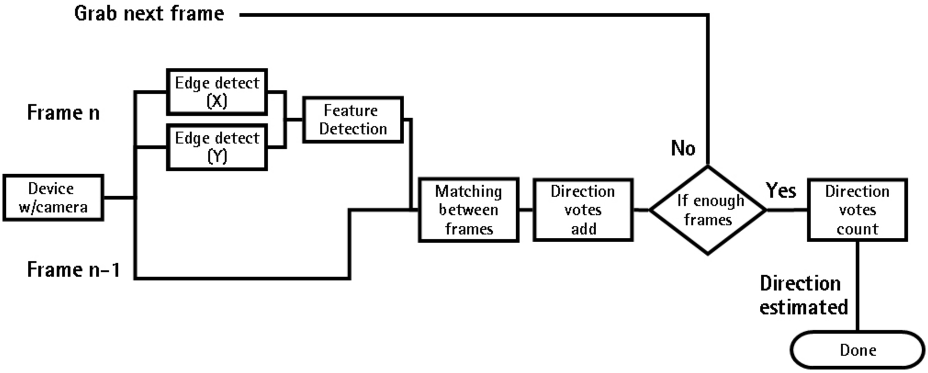


Fig. 2. Tracking algorithm and direction estimation flowchart

commercial camera motion tracking-based games. Recently [6] have created the first Kalman-based tracker for mobile devices. Kalman tracking yields higher quality motion estimates but has higher computational requirements and needs a more complex implementation than in our work.

Prior work on speed-dependent automatic zooming on mobile devices has focused on performing zoom and/or panning using either additional hardware or sensors. Igarashi and Hinckley [7] performed speed-dependent automatic zooming by creating equations based on mouse motion which determine whether to zoom in or out. Recently, Cockburn *et al.* [8] have performed extensive user studies to find empirical values for the equations relating speed and zoom levels for mouse motion. Other pointer device based scrolling techniques include the Alphaslider [9], the FineSlider [10] and the Popup Vernier [11]. Those works focused on how to effectively select an item from a list of a large number of items. An extended scroll bar component that allows the user to change the scrolling speed was used. Our approach can be used as an alternative in instances where a pointing device is not available, such as on a mobile device.

Our work is also similar to the scroll [12], and peephole [1] displays works and work on tilt-based interaction [13, 14]. In these works, the goal was to perform scrolling on mobile devices in a more intuitive fashion by using additional sensors. Scroll-detection sensors that were used included both mechanical and optical mouse sensors, position and orientation sensors, and ultrasonic transmitters/receivers. While additional sensors were required in those works, we use only the camera as the direction sensor instead of adding new sensors. Doing so allows for regular camera-equipped smart phones to have an additional interaction modality without modifying the phone.

Other hardware-based solutions to scrolling come from the commercial domain. Apple's iPod, while not performing zooming, makes use of a touch-sensitive scroll wheel whose scrolling speed depends on the number of songs in a play list, to maximize display usage. On other mobile devices such as cell phones, touch screens are commonly used to address display size limitations. Touch screens can allow users to interact and scroll through their data more effectively than using buttons as the stylus can just be dragged down a scrollbar. However, touch screens have the disadvantage of not permitting one-handed operation.

3 Camera-Based Movement Estimation

Our approach is to use the mobile phone's onboard camera as the source of input. The user's physical movement of the device is captured in incoming video, which is analyzed to determine scroll direction and magnitude. The detected direction is sent to the event handler exactly as a corresponding mouse event, while the magnitude is used to specify the current scroll level. Figure 1 shows an overview of our system.

Correctly interpreting the observed motion from the camera's incoming video requires accurate tracking. To determine the motion direction, a feature-based tracking algorithm is used. To determine the magnitude of the physical movement, motion history images (MHI) [15] are used, which were originally used for performing action and gesture recognition. Our tracking algorithm provides four directions as application-level events, similar to mouse movement: up, down, left and right, in the camera plane. The magnitude is also passed as an event, where two states are possible: motion magnitude increasing or decreasing. The rest of the application remains the same as the only changes are the cause of the events passed to the event handler. This allows applications to use the camera easily, without any knowledge of the underlying tracking algorithms or camera hardware.

High-Level Algorithm Description: The tracking system was implemented on the Symbian OS. The process diagram of the tracker is presented in Figure 2. Two frames are grabbed, n and $n-1$. Edge detection is performed on both frames using the Sobel filter. The thresholded absolute values of the x and y derivatives are used as features as they peak in corner-like regions. Feature matching is performed between frames using template matching with 15x15 search windows. Direction voting is performed using variables, and the final decision on motion estimation is performed every 4 frames. This allows several frames to 'vote' on the motion, keeping the scrolling from being incorrect due to any errors in other parts of the system.

Feature Detection: Traditional features include edges and corners. However, edges are not significantly temporally coherent and corner features are too computationally expensive to find at many image locations while retaining real-time performance. Instead, corner-like features are detected using image gradient information (Equation 1).

$$S(x,y) = (G_x^2 + G_y^2) \tag{1}$$

$$G_x(x,y) = \frac{\partial I}{\partial x} \approx sobel_x(x,y),\ \ G_y(x,y) = \frac{\partial I}{\partial y} \approx sobel_y(x,y) \tag{2}$$

$S(x,y)$ is the Sobel operator and the Sobel functions denote convolution with the x and y components of the Sobel kernel. All corners cannot be detected using the Sobel operator; however, it provides a useful first-step culling of pixels for additional processing. Frame n is filtered using the Sobel x and y filters. The Sobel operator is then applied to every pixel in scanline order. If $S(x, y)$ is greater than an edge threshold, the pixel at (x, y) in frame n is labeled a feature. Once k features are detected the Sobel operator is no longer applied, with $k = 50$ providing good results. The list of detected features is then passed onto the next step, template matching.

Template Matching: Template matching alone is not reliable since only image pixel difference errors are used and neither sensor noise nor lighting variations are modeled. Template matching is used in this algorithm because it is computationally inexpensive and provides useful match estimates. Matching is performed for each feature detected in Frame n. For each feature, the 15x15 pixel neighborhood around the feature is tested for image similarity using the sum of squared differences (SSD). 15x15 sized features were chosen as this size is large enough to capture visually distinct regions and significant intra-frame physical motion. Let t_f denote a 15x15 pixel sized template image consisting of the pixel neighborhood at (i, j), where feature f was detected in Frame n. Then, to find the closest match in Frame $n - 1$, we can use the following equation (equivalent to SSD in the case of a non-changing image and template):

$$\min_{(x,y)\in N} M(x, y) = \sum_{k=-7}^{7} \sum_{l=-7}^{7} t_f(k + 7, l + 7) f(x + k, y + l) \tag{3}$$

where N is the 15x15 pixel neighborhood around (i, j). The location of the closest match is found by testing every offset around location (i, j) and comparing the 15x15 sub-image there with the 15x15 sub-image from the feature's pixel neighborhood. The matching is performed from the current frame to the previous frame instead of vice versa since a feature detected in Frame $n - 1$ may not be detected in Frame n.

Direction Estimation: The direction cannot be estimated by simply counting the most dominant template matching direction amongst all features. Such estimation would be temporally incoherent since neither the feature detection nor template matching component is perfectly coherent. To remove temporal incoherencies, the estimated directions of the matched feature locations are temporally filtered. For each frame, the most dominant direction is computed and a counter for that direction is incremented. For each direction, a counter is initialized at zero. After m frames, where m is typically between $3 - 5$ frames, the counter with highest count is chosen as the estimated direction with other counters reset to zero. Only a small amount of temporal filtering is needed since the features are individually robust. The direction estimation fails if the camera is moved largely between frames since at least one feature from the previous frame must be visible, as in other template matching based algorithms.

Determining Camera Motion Magnitude: The directions of dominant camera motion are computed using the tracking algorithm, but their magnitudes are not known accurately. Camera motion magnitude must be calculated accurately to determine how to adjust the scroll speed in applications that need zoom control. We use motion history images (MHI) [15] to estimate camera motion magnitude. Motion histories are encoded in single images such that a single image can be used for simple, robust and computationally inexpensive gesture recognition. An MHI is computed by performing background subtraction between the current and previous frames. At locations where the pixel values change, the MHI is updated by decrementing by a pre-defined constant amount. By averaging the intensity values of the MHI, the average camera motion magnitude is estimated. The following equation calculates the MHI's value at position (x, y) in the camera image at time t:

$$H_T(x,y,t) = \begin{cases} \tau & \text{if } \ D(x,y,t) = 1, \\ \max(0, H_T(x,y,t-1) - 1) & \text{otherwise} \end{cases} \tag{4}$$

where H_T is initialized to be 0 in the first frame and $D(x, y, t)$ denotes an image difference between frame t and $t-1$, with τ being the number of frames of motion that the MHI should represent. In this manner, the MHI compactly represents the motion magnitude from the incoming video, with areas 'lighting up' when significant motion is detected and the whole MHI fading to black if no motion is detected. The simplicity of the MHI calculation makes it amenable for use in driving the scroll level and velocity as well as camera shake detection (Section 5).

4 Results

Our algorithm's frame rate is 10fps on our test hardware, a Nokia 6630 smartphone, with 1 motion magnitude and motion direction update every $3-5$ frames, with more frequent updates possible at the expense of accuracy. In our experimentation, only smooth walls result in complete tracking failure since temporally coherent features are not found. Otherwise, the tracker's performance matches users' physical motion at a responsive rate with no errors in typical indoor and outdoor environments.

Motion estimates computed are accurate for user interaction, never estimating the wrong motion direction even when the device is abruptly switched in directions. The motion magnitude is also accurate, mainly representing the immediate past since is is not recomputed often, which turns out to be suitable for automatic zooming. The biggest limitation of our algorithm is that detectable physical motion speed is limited since intra-frame matching is performed and part of the previous frame must be visible in the current frame to establish feature correspondences.

To measure the accuracy and performance of our algorithm, we compared our tracking algorithm with the Kalman filter-based tracker from [6]. The Kalman tracker has higher motion estimation accuracy, as expected, since the Kalman filter greatly improves the quality of intra-frame matching. However, the computational requirements are significantly greater since several matrices must be multiplied and inverted per frame. On devices with limited computational resources, our algorithm provides sufficient motion and velocity accuracy for user

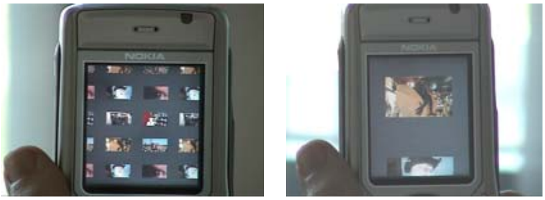


Fig. 3. Picture browser application. The application automatically adjusts the zoom level to help the user browse

interaction with many leftover cycles for intense applications such as 3D games (Section 5) at the trade-off of more limited accuracy since the temporal filtering in our algorithm cannot match a Kalman filter.

A general issue with camera-based mobile device user interfaces is that the user's physical environment may have very limited space. In such situations, it may be advantageous to provide a 'clutch' to turn the tracking on/off. This would emulate the act of lifting a mouse once the edge of a desk is reached in traditional desktop interaction. In our informal testing we did not provide a clutch, however in commercial implementations this is a consideration to keep in mind. Another general issue is that all camera-based user interfaces require adequate light for tracking. Problems can arise particularly for a mobile device in low lighting conditions if the device has an automatic 'night mode' as incoming images will already be processed and may be too noisy. In dark environments, applications should default to joypad-based interaction.

5 Applications

We implemented several test applications to demonstrate our algorithm's strengths and limitations. In general, any mobile application that requires scrolling and/or zooming could make use of our approach provided that an onboard camera is present.

Zooming Photo Browser: As cameras become more widespread on mobile phones and storage size increases, managing photos becomes a more difficult task for the user as large amounts of information must be viewed with limited input modalities. Current typical photo viewer applications show photo thumbnails as lists, grids, or 3D carousels. Since image selection and scrolling are done with the joystick, the amount of time a user needs to browse their images is directly related to the number of images that they are browsing.

Our photo browser test application (Figure 3) shows thumbnails of the user's photos in a grid layout. The user can scroll in four directions (up, down, left, right, in the camera plane) by physically moving the mobile device. In this case it is difficult to view all the images as some zoom control is required when looking for a particular image. If the zoom level is not properly set, it is difficult for a user to select a particular image from the set as the scrolling will be too fast. To

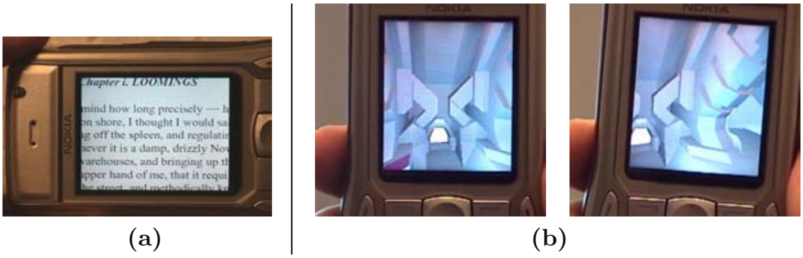

hapter i. LOOMINGS
mind how long precisely — h
on shore, I thought I would sa
g off the spleen, and regulati
never it is a damp, drizzly No
arehouses, and bringing up t
pper hand of me, that it requi
NOKIA
NOKIA
NOKIA
(a)
(b)

Fig. 4. (a) Camera-based interaction in a document viewing application. (b) Mapping physical motion to viewing direction creates the illusion of a window into an environment

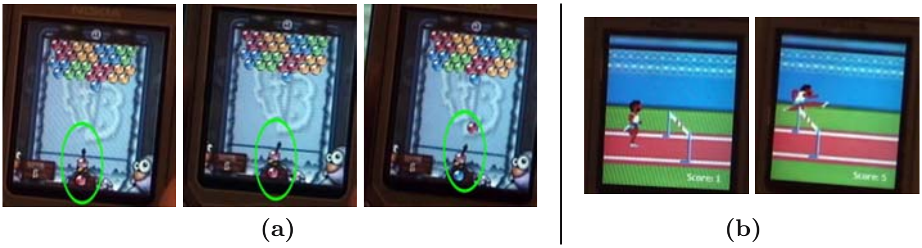
address this problem, we used the technique introduced in [7]. Adaptive zooming based on the magnitude of the user's physical movement keeps the scroll speed virtually consistent, allowing the user to browse more thumbnails by only moving the device faster.

Document Viewer: Scrolling a document is a commonly difficult task on mobile devices. For instance, web content designed for desktop computers is much vertically longer since mobile devices have narrower screens. In addition, joystick scrolling is especially difficult when scrolling line-by-line. An alternative is to add an extra hardware button for scrolling. However, an extra button is not a preferable solution for mobile device manufacturers due to the lack of extra physical space on the device along with additional manufacturing costs.

In the document viewer prototype application we implemented (Figure 4 (a)), the user can vertically scroll documents by moving the device. The scroll speed depends on how fast the user moves the device, which is much more intuitive than changing scrolling speed depending on how long the user presses the joystick or via menu options and settings. One issue we identified in this application is that at some point, the user has to move the device more than they can reach. For example, if the user is scrolling to the right, at some point they will reach the physical limit of their arm's motion. To address this problem, we use the joystick as a 'carriage return', which scrolls the document to the beginning of the next line and allows the user to move their arm back to the left again. After a carriage return, all tracked motion except movement to the right is ignored.

3D Game Interaction: Creating an immersive 3D experience is difficult on mobile devices due to the limited display size. The most immersive experiences are typically created using a combination of large displays reducing peripheral vision as much as possible and/or virtual environment navigation tied to the user's physical motion. In our prototype (Figure 4 (b)), we map the user's physical motion to the view-point to create the illusion of a window into a 3D world.

Our renderer loads standard Quake III$^{\text{TM}}$ or Quake III Arena$^{\text{TM}}$ maps. Textures, light maps, curved surfaces and lighting calculations are disabled for performance. The rendering is done using the OpenGL ES implementation available in the latest Symbian OS-based Series 60 SDK. Pre-computed vertex lighting and



(a)

(b)

Fig. 5. (a) Players move the device left and right to aim, and shake the device to launch a bubble. (b) A jump command is issued by shaking the device at the correct time to avoid tripping on the hurdle

fixed point calculations are used to improve performance due to the lack of a floating-point unit on our test hardware. The renderer is able to realistically render lit virtual environments with several thousand polygons per scene at $3 - 10$ frames per second, depending on the environment that is chosen.

Navigation of the virtual environments is performed with a combination of physical motion and keypad presses. Actual movement in the environment is controlled by the keypad. The user looks around in the scene by physically moving the device around their body in the directions that they would like to look. We map the tracked camera motion directions to a trackball as in traditional mouse-based 3D interaction. The combination of detailed environments, camera-based control and interactive frame rate create a mobile user experience closer to that using additional hardware or larger displays.

2D Game Interaction: Camera-based user interaction can be used to enhance 2D games as well as those that are 3D. Camera motion can be used to add an additional element of interaction in games that require precise movements or very well-timed button presses. We created puzzle and action game prototypes to investigate these ideas using the camera motion and shake detection algorithms presented.

We modified the open source Series 60 port of the 'Frozen Bubble' puzzle game (http://fb–s60.sourceforge.net/), switching the game control from using the keypad to using the camera (Figure 5 (a)). In our version, the user moves their device left and right to aim and performs sudden shakes to launch their bubble. This has the effect of significantly changing gameplay as careful arm motions are now required to aim, instead of a number of button presses, which increases the excitement as the game is now more physically-based.

We created a camera-based action game prototype as well. Using sprites and artwork from Konami's 'Track and Field$^{\mathrm{TM}}$' game for the Nintendo Entertainment System, a new game (Figure 5 (b)) was created. A runner must jump over a never-ending number of approaching hurdles. To jump, the player must time the shaking of their device correctly so that the character does not crash into hurdles. Relying on the camera exclusively for input results in a game that is very simple to learn and understand but difficult to master, providing a new type of game. Shake detection is performed by thresholding the average intensity of the computed MHI (Section 3).

6 Conclusions and Future Work

We introduced a new approach to improve the user experience on camera-equipped mobile devices. A feature-based tracking algorithm was presented to detect both physical motion direction and magnitude to permit one-handed physical movement based interaction. A camera was chosen since cameras are now widely available on mobile devices and are very powerful sensors that can be used without introducing new sensors. We demonstrated our approach in several applications including a document viewer, photo collection browser and some games. In the future, we would like to perform user studies to determine how to improve user interaction further using mobile cameras.

While our tracking algorithm is computationally efficient and works well in practice, some situations cannot be handled. Severe lighting differences will cause the template matching to stop working properly. Motion in front of the camera is ambiguous and can affect tracking results as it is impossible to tell whether the camera is moving or not. Shadows may confuse the tracking system, but there are known techniques for robust tracking in the presence of shadows that will be incorporated into the tracking algorithm once additional processing speed is available.

References

1. Yee, K.P.: Peephole displays: pen interaction on spatially aware handheld computers. In: Proceedings of the SIGCHI conference on Human factors in computing systems. (2003) 1–8
2. Rohs, M.: Real-world interaction with camera-phones. In: International Symposium on Ubiquitous Computing Systems. (2004)
3. Möhring, M., Lessig, C., Bimber, O.: Optical tracking and video see-through ar on consumer cell phones. In: Proceedings of Workshop on Virtual and Augmented Reality of the GI-Fachgruppe AR/VR. (2004) 193–204
4. Drab, S., Artner, N.: Motion detection as interaction technique for games & applications on mobile devices. In: Extended Abstracts of PERVASIVE: Workshop on Pervasive Mobile Interaction Devices. (2005) 48–51
5. Beier, D., Billert, R., Bruderlin, B., Stichling, D., Kleinjohann, B.: Marker-less vision based tracking for mobile augmented reality. In: Proceedings of the Second IEEE and ACM International Symposium on Mixed and Augmented Reality. (2003) 258–259
6. Hannuksela, J., Sangi, P., Heikkila, J.: A vision-based approach for controlling user interfaces of mobile devices. In: IEEE Workshop on Vision for Human-Computer Interaction. (2005)
7. Igarashi, T., Hinckley, K.: Speed-dependent automatic zooming for browsing large documents. In: Proceedings of the ACM symposium on User interface software and technology (UIST). (2000) 139–148

8. Cockburn, A., Savage, J., Wallace, A.: Tuning and testing scrolling interfaces that automatically zoom. In: Proceedings of the SIGCHI conference on Human factors in computing systems. (2005) 71–80
9. Ahlberg, C., Shneiderman, B.: The alphaslider: a compact and rapid selector. In: Proceedings of the SIGCHI conference on Human factors in computing systems. (1994) 365–371
10. Masui, T., Kashiwagi, K., George R. Borden, I.: Elastic graphical interfaces to precise data manipulation. In: CHI'95: Conference companion on Human factors in computing systems. (1995) 143–144
11. Ayatsuka, Y., Rekimoto, J., Matsuoka, S.: Popup vernier: a tool for sub-pixel-pitch dragging with smooth mode transition. In: Proceedings of the ACM symposium on User interface software and technology (UIST). (1998) 39–48
12. Siio, I.: Scroll display: Pointing device for palmtop computers. In: Asia Pacific Computer Human Interaction. (1998) 243–248
13. Bartlett, J.: Rock 'n' scroll is here to stay. IEEE Computer Graphics and Applications **20** (2000) 40–45
14. Rekimoto, J.: Tilting operations for small screen interfaces. In: Proceedings of the ACM symposium on User interface software and technology (UIST). (1996) 167–168
15. Davis, J., Bobick, A.: The representation and recognition of human movement using temporal templates. In: Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR). (1997) 928–934

Fast Head Tilt Detection for Human-Computer Interaction

Benjamin N. Waber, John J. Magee, and Margrit Betke

Computer Science Dept., Boston University
{bwabes,mageejo,betke}@cs.bu.edu

Abstract. Accurate head tilt detection has a large potential to aid people with disabilities in the use of human-computer interfaces and provide universal access to communication software. We show how it can be utilized to tab through links on a web page or control a video game with head motions. It may also be useful as a correction method for currently available video-based assistive technology that requires upright facial poses. Few of the existing computer vision methods that detect head rotations in and out of the image plane with reasonable accuracy can operate within the context of a real-time communication interface because the computational expense that they incur is too great. Our method uses a variety of metrics to obtain a robust head tilt estimate without incurring the computational cost of previous methods. Our system runs in real time on a computer with a 2.53 GHz processor, 256 MB of RAM and an inexpensive webcam, using only 55% of the processor cycles.

1 Introduction

Many existing face analysis systems require an upright face as input to function correctly and often assume that no head tilt occurs (e.g., [1–5]). A head tilt detection system could serve as a pre-processor to these systems by rotating the input image by the estimated head tilt angle and thus facilitate interaction in circumstances when the head is not upright. This would be particularly important for people with severe motion impairments, e.g., due to cerebral palsy or multiple sclerosis, who often have difficulties holding their heads straight. They could then use video-based assistive technology that assumes an upright face, such as EyeKeys [4], and thus gain access to communication software. Head tilt detection can also enhance human-computer interaction by providing an additional mechanism to select commands. Left and right head tilts could be used to rotate a 3D model, control a video game, tab through a web page, or select letters in a scan-based text-entry program.

Among previous face detection algorithms, CAMSHIFT [6] is particularly geared towards real-time human-computer interaction. It requires a color model of the face that it will be tracking, and it finds the center, elongation, and tilt of the face by determining the likelihood that pixels in the input image belong to the face based on a comparison of their color values with the prior face color model. Other face tracking methods also use color and motion information [7, 8].

N. Sebe, M.S. Lew, and T.S. Huang (Eds.): HCI/ICCV 2005, LNCS 3766, pp. 90–99, 2005.

Face detection systems that require extensive training use neural networks [9] or AdaBoost [10–12]. These systems are very effective in detecting upright faces, but give only coarse estimates of head tilt and put severe computational and memory strains on a computer that, if used for human-computer interaction, must at the same time provide computational resources for the application program.

Motion of the head, the foreground object, in a video sequence causes pixels to transition from background to foreground or foreground to background, depending on the direction of motion. We observed that an image representation of these transitions contains distinctly grouped pixels created by the motion of the head, in particular by the change of its occluding boundary. Our contribution is a method of estimating head tilt from the size, relative location, and orientation of these groups. A second contribution is our method to estimate head tilt from the motion of the center of the face. Assuming head rotation is parallel to the image plane, a circle can be fit to the sequence of face centroids as the head rotates. This second estimate of head tilt is combined via a weighting function with our estimate based on motion of the occluding boundary to yield an estimate that is comparable to the most powerful methods to date (e.g., [9, 12]).

Not only is our method a novel technique, but its real-time performance on common desktop computers with inexpensive cameras makes it immediately applicable to human-computer interaction, unlike some previous approaches (e.g., [9, 12]). Our method can be integrated as part of a larger interaction system; here we show its performance as a stand-alone system.

2 Head Tilt Estimation Algorithm

Our head tilt estimation algorithm has four steps:

1. Foreground and background segmentation,
2. Analysis of the motion of the occluding boundary of the face, which provides head tilt estimate θ_b,
3. Analysis of the motion of the center of the face to compute angle estimate θ_c,
4. Analysis of the confidence factor w that determines the weighting of the two estimates θ_b and θ_c in computing the final head tilt estimate $\theta = (1-w)\,\theta_b + w\,\theta_c$.

We assume that our algorithm has access to a foreground-detection method. Ideally, the segmented foreground image I_f contains the user's head alone, but our method can also handle foreground segmentations that include the user's neck and shoulders and regions in the background (e.g., it works for the poor foreground segmentation in Fig. 1C). In our experiments, we used a simple foreground estimation method that subtracted the current frame (Fig. 1B) from the initial frame without the user (Fig. 1A). Our method also relies on the mild assumptions that movements in the background are not correlated with the user's head motion and affect a smaller number of pixels in the video than the user's head motion.

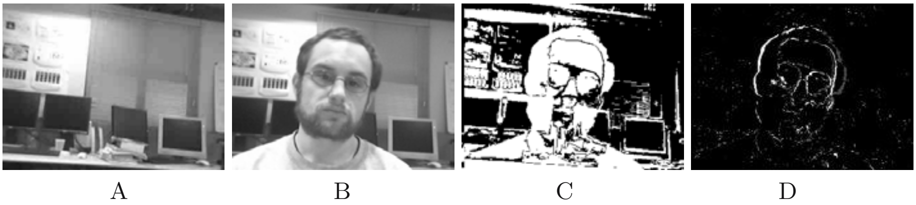


Fig. 1. Images used by our system. A) Background image. B) Example input frame. C) Foreground image $I_{f,t}$ computed from differencing images in A and B (here noisy due to camera shift). D) The signed difference $I_{f,t} - I_{f,t-1}$. The gray pixels represent $I_{b\rightarrow f}$, the white pixels represent $I_{f\rightarrow b}$

Foreground pixels in I_f are set to 1, while background pixels are set to 0. A representation of motion is obtained by subtracting the foreground image of the previous frame from the foreground image of the current frame, taking care to preserve the sign of the result (Fig. 1D). The image is then separated into two frames: binary image $I_{b\rightarrow f}$ to represent pixels that changed from belonging to the background at time $t-1$ to belonging to the foreground at time t, and $I_{f\rightarrow b}$ to represent pixels in the foreground at time $t-1$ and in the background at time t.

2.1 Angle Estimation Based on Occluding Boundary of Face

We now explain how head tilt angles are estimated in the binary motion images $I_{b\rightarrow f}$ and $I_{f\rightarrow b}$. Three filtering steps are performed on both $I_{b\rightarrow f}$ and $I_{f\rightarrow b}$ to find connected components that represent the motion of the occluding boundary of the face well, in particular, at the sides of the face. First, connected components too large or too small to represent the face are removed (Fig 2A). Then components too far from the centroids of the respective components of $I_{b\rightarrow f,t-1}$ and $I_{f\rightarrow b,t-1}$ are removed (Fig. 2B). Furthermore, components whose orientation is too far from the estimated orientation of the face in the previous frame are removed (Fig. 2C). To perform this last filtering step, each remaining component is processed as follows:

Pixels with high curvature at the top and bottom of the filtered components are removed since they do not represent the motion of the sides of the face well (Fig. 2D). For each connected component i, its lowest pixel $\mathbf{x}_{i,o}$ is used as the origin of a local polar coordinate space. We ignore additional portions of the connected component above the origin with lower, but still relatively high curvature within this connected component (orange in Fig. 2D) to allow for more accurate angle estimation. The angle $\alpha_{i,j}$ representing pixel $\mathbf{x}_{i,j}$ is thus the angle between the line $\mathbf{x}_{i,j} - \mathbf{x}_{i,o}$ and the horizontal axis. If the face is tilting left, which appears as a right rotation in the image, the pixel with the largest angle $\theta_{L,i}$ is roughly equivalent to the head tilt angle estimated in the previous frame, and the pixel with the smallest angle $\theta_{R,i}$ can be used to describe the head tilt in the current frame:

$$\theta_{L,i} = \max \{ \, \alpha_{i,j} \mid \forall \, j \text{ in component } i \, \}, \tag{1}$$

$$\theta_{R,i} = \min \{ \, \alpha_{i,j} \mid \forall \, j \text{ in component } i \, \}. \tag{2}$$

Angle $\theta_{R,i}$ is a particularly good approximation of the head tilt if component i represents the motion of the right side of the face (shown left) in $I_{f \to b}$ (Fig. 2D) or the motion of the left side of the face (shown right) in $I_{b \to f}$. Vice versa, if the face is tilting right, angle $\theta_{L,i}$ is a particularly good approximation of the head tilt if component i represents the motion of the right side of the face (shown left) in $I_{b \to f}$ or the motion of the left side of the face (shown right) in $I_{f \to b}$.

If angles $\theta_{L,i}$ or $\theta_{R,i}$ are not within a small γ_T of the head tilt estimate from the previous frame, then component i is discarded. The threshold γ_T was experimentally set to 10 degrees and found to work quite well.

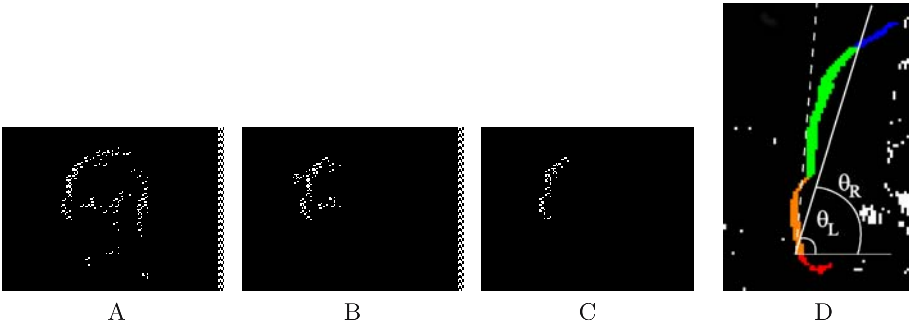


Fig. 2. Results of filtering steps on image $I_{f \to b}$ in Fig. 1D. A) Size filter. B) Filter on distance from centroid (cross) in previous frame. C) Filter on on difference to estimated angle in previous frame. Here, but not generally, only one component remained. D) Zoomed-in sub-image of $I_{f \to b}$ containing the filtered component in C (colored pixels) and the discarded components (white). The pixels at the top and bottom of the filtered component (blue, red, and orange) are disregarded in estimating the orientation of the occluding boundary. Angle $\theta_{L,i}$ is approximately the head tilt in the previous frame (dashed line) and angle $\theta_{R,i}$ the head tilt in the current frame (solid line)

At this point in the process, our method has eliminated all components that do not represent the motion of the sides of the face well and created a list of angles $\theta_{L,i}$ and $\theta_{R,i}$ for both $I_{b \to f}$ and $I_{f \to b}$. Our algorithm compares these two lists by only considering those pairs of angles $\theta_{b \to f}, \theta_{f \to b}$ that are within a threshold τ of each other. Threshold τ was experimentally set to 20 degrees. If there is more than one pair remaining, pairs are eliminated if their size difference is too large. In particular, a component that is less than 2/3 the size of its paired component can be safely eliminated. If, however, no components meet this criterion and there is only a single candidate component remaining in either $I_{b \to f}$ or $I_{f \to b}$, then no pairs are eliminated in this way. This deals with cases when components of motion that should be considered are broken into smaller pieces as a result of poor foreground estimation. If there are still multiple pairs remaining, the pair $\theta_{b \to f}, \theta_{f \to b}$ that has the largest combined area is chosen.

The estimate of head tilt based on the analysis of the occluding boundary of the sides of the face is then computed by the average

$$\theta_b = \frac{\theta_{b\to f} + \theta_{f\to b}}{2}. \tag{3}$$

We also define a measure of confidence p_t that $\theta_{b,t}$ represents head tilt in the current frame t:

$$p_t = 1 - \frac{|\theta_{b\to f} - \theta_{f\to b}|}{\tau}. \tag{4}$$

The confidence is high if the angle estimates based on binary motion images $I_{b\to f}$ and $I_{f\to b}$ are similar and low otherwise.

2.2 Angle Estimation Based on Circular Face Motion

The angle estimate θ_b can be further refined by comparing the location $\mathbf{c}_{t,\mathrm{meas}}$ of the center of the face in current frame t, measured in foreground image I_f, with its predicted location $\mathbf{c}_{t,\mathrm{circle}}$. The prediction is based on a motion model that assumes that the center of the face follows a circular arc in the video as the face tilts sideways, and the center of rotation is a point on the neck (Fig. 4A). A pixel $\mathbf{x}_0$ is used as the center of rotation. It has the same x-coordinate as the center of the face $\mathbf{c}_{1,\mathrm{meas}}$ when the face is in an upright position. The initial radius of the circle is the distance $\| \mathbf{c}_{1,\mathrm{meas}} - \mathbf{x}_0 \|$ between the initial center of the face and its projection (Fig. 4A). The location of $\mathbf{c}_{t,\mathrm{circle}}$ on the circular arc is determined by the intersection of the line between $\mathbf{x}_0$ and $\mathbf{c}_{t,\mathrm{meas}}$ with the arc. If a large distance $\| \mathbf{c}_{t,\mathrm{circle}} - \mathbf{c}_{t,\mathrm{meas}} \|$ between predicted and measured face center occurs in subsequent frames, the radius is updated by the distance $\| \mathbf{x}_0 - \mathbf{c}_{t,\mathrm{meas}} \|$.

The estimate of head tilt based on the average of the observed and predicted positions of the face center is

$$\theta_{c,t} = \frac{\| \mathbf{c}_{t,\mathrm{meas}} - \mathbf{c}_{t,\mathrm{circle}} \|}{2}. \tag{5}$$

As a measure of confidence in the angle estimate, we use the ratio e_t of distance between the observed and predicted center positions to the maximum possible distance T_t that the two points could be apart, i.e.,

$$e_t = 1 - \frac{\| \mathbf{c}_{t,\mathrm{meas}} - \mathbf{c}_{t,\mathrm{circle}} \|}{T_t}, \tag{6}$$

where T_t is the distance to the farthest corner of the image from $\mathbf{c}_{t,\mathrm{circle}}$.

2.3 Weighted Angle Estimation

The head tilt estimate θ_b, computed by analyzing the occluding boundary of the face, and the estimate θ_c, computed by analyzing the circular motion of the face, can be combined to compute the weighted angle estimate

$$\theta_t = (1 - w_t)\,\theta_{b,t} + w_t\,\theta_{c,t}, \tag{7}$$

where the weighting factor

$$w_t = \frac{e_t^2}{e_t^2 + p_t^2} \tag{8}$$

is computed from e_t and p_t, which represent the respective confidences in estimates θ_b and θ_c. To prevent one measure from being the sole source of our final angle estimate, we set a lower bound of 0.2 for the weight of $\theta_{b,t}$ and 0.1 for the weight of $\theta_{c,t}$.

A summary of the head tilt estimation algorithm is given in Figure 3.

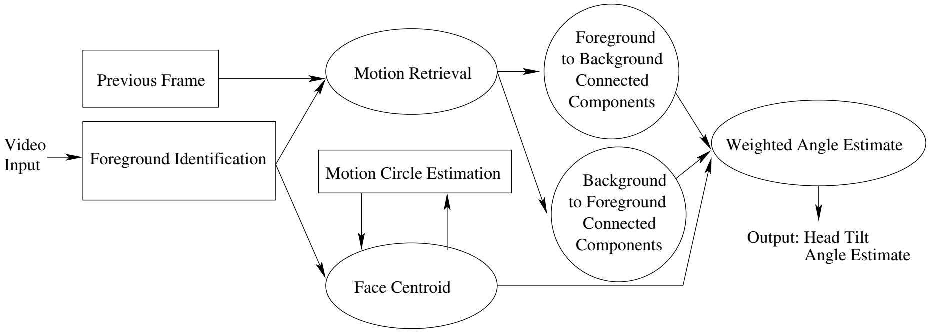

Previous Frame
Video Input
Foreground Identification
Motion Retrieval
Motion Circle Estimation
Face Centroid
Foreground to Background Connected Components
Background to Foreground Connected Components
Weighted Angle Estimate
Output: Head Tilt Angle Estimate

Fig. 3. Flowchart of head tilt estimation method

3 Human-Computer Interaction Experiments

To evaluate our method we performed three experiments. In experiment 1, we compared our system's head tilt estimation to that of two volunteers, who labeled each frame of five 320-frame video segments with their estimate of head tilt angle (Fig. 4B). Each of these video segments contained 5–7 separate head tilts with motion varying from slow and smooth to quick and erratic for a total of 31 distinct head tilts across 1320 images. In two of the video sequences, people were walking through the background. To verify that all the steps in our algorithm are necessary, we also tested our method in three other configurations: with the weighting component removed, with the centroid angle estimation removed, and with only centroid angle estimation.

In the second experiment, we tested users' ability to play the BlockEscape game [4] using our interface. We tested 5 users who each played 4 games. The game BlockEscape was developed as a tool to test the performance of human-computer interfaces. In the BlockEscape game, the screen contains a number of horizontal walls that are separated vertically by a fixed distance (Fig. 4C). Each wall has one or more holes of various widths. At the beginning of the game, a block appears on top of the screen. The goal of the game is for the user to lead the block through the holes in the walls until it reaches the bottom of the screen. The game was implemented so that the user only needs to initiate the block's horizontal movement by issuing "move block left" or "move block right" commands. The block continues to move in the initiated direction until the user issues a command for movement in the opposite direction. The block's vertical motion is automatic – when a hole is reached, the block "falls through" to the next wall or bottom of the screen. When a wall reaches the user's block, it pushes the block upwards. The user wins if he or she leads the block through the holes in the walls to the bottom of the screen and loses if a wall pushes the block up to the top of the screen. During playing, usage statistics, in particular, the departure of the user-controlled block from an optimal path, were computed based on the positions of the block, walls, and holes.

Our last experiment tested users' ability to navigate web pages with our head tilt estimation method. Test subjects were repeatedly shown web pages consisting of three links and directed to select a particular sequence of links (on average, the 2nd link on the page). A left head tilt directed the web browser to highlight the next link, while a right head tilt directed the browser to follow the currently selected link. The same sequence of links was used for all participants. We recorded the number of links each participant successfully followed before following an incorrect link.

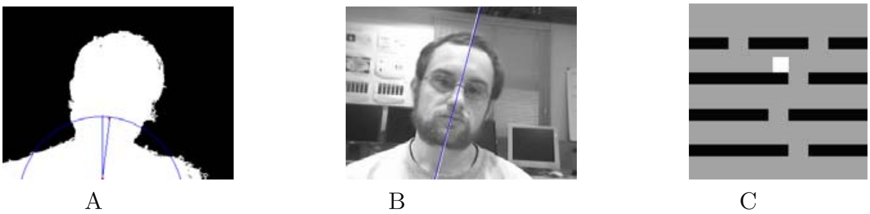


Fig. 4. A) Centroid angle detection binary image showing the centroid circle, circle center, and the original and current frame centroid positions. B) Experimental system output. Our system's head tilt estimate is represented by the white line, while the blue line represents the average of the two volunteers' estimates. C) Screenshot of the game BlockEscape. As the white block falls towards the bottom of the screen, the player navigates it through the holes in the black walls, which move upwards, by initiating "move block left" or "move block right" commands

4 Results of Human-Computer Interaction Experiments

Our system runs in real time on a computer with a 2.53 GHz processor, 256 MB of RAM and in these experiments used less than 55% of the processor cycles. The comparison of our system's head tilt estimations to those made by two volunteers is given in Table 1. A screenshot of the system's estimate and that of the volunteers shown in Fig. 4B. If we consider human observation to be ground truth, our system has exhibited a good performance. Each component of our method, angle estimation based on the analysis of the motion of the occluding boundary of the face, analysis of the moving center of the face, and the weighting scheme, is vital to the success of our interface. It is the combination of these components that yields a valuable HCI tool, and removing one of them yields a decline in performance (Table 1).

Table 1. Accuracy-Evaluation Experiment: Angle differences between four versions of our method and human observation, using 2 volunteers and five 320-frame video segments. The estimates of degrees were rounded to whole numbers because precision beyond this level in human observations is unlikely

	Weighted Estimate θ		**Equally Weighted Estimate** $\theta_b + \theta_c$		**Boundary-based Estimate** θ_b		**Center-based Estimate** θ_c	
Subject	Median Ang. Diff.	Std. Dev.	Median Ang. Diff.	Std. Dev.	Median Ang. Diff.	Std. Dev.	Median Ang. Diff.	Std. Dev.
1	6	7	9	8	14	9	15	10
2	8	6	11	9	13	9	17	9
Avg.	7	6	10	8	14	9	16	10

The results for our BlockEscape experiment are summarized in Table 2. The subjects were able to use our head-tilt interface with similar success as the EyeKeys [4] or Camera Mouse [13] interfaces. All but one subject using our interface won their last three games. The other won their last two games, which indicates that practice improved the quality of the human-computer interaction. If we eliminate each subject's first game from the analysis, the average path deviation is 1.4 and the median is 0.6.

Table 2. Game-Interaction Experiment: Five users played 4 games of BlockEscape, issuing commands by head tilt and with three other interfaces [4]. Rows 1–3: The number of deviations of the block from the optimal path during gameplay, which are assumed to be due to false detections of the interface rather than misjudgments of the player

Interaction Method	Head Tilt	EyeKeys	Camera Mouse	Keyboard
Path Deviation: Median	2.2	2.5	0	0
Average	2.6	2.9	2.3	0
Std. Dev.	2.8	4.0	2.7	0
Wins	16/20 (80%)	10/12 (83%)	10/12 (83%)	12/12 (100%)

In the games that resulted in losses, the users tilted their heads too far sideways in executing a command. This motion and the return motion to the neutral upright state both took time. As a result, it took the subjects too long to issue the sequence of commands needed to navigate the block through the holes in the moving walls before the game ended. After gaining experience with the interface, the users soon became aware of this issue and played much better.

In the third experiment, users were able to follow an average of 5.9 links to new web pages, indicating that 11.8 head tilts were correctly detected for each incorrect detection. Individual results are in Table 3. These results indicate that our method may be used in a real-world context of surfing the Internet.

Table 3. Web Browsing Experiment: Number of web links correctly followed until a link was followed by mistake

Participant	Trial 1	Trial 2	Trial 3	Trial 4	**Mean**
1	3	5	7	2	4.3
2	6	8	9	8	7.8
3	7	7	6	9	7.3
4	4	3	5	2	3.5
5	8	7	7	4	6.5

5 Discussion and Future Work

Through our experimental results, we have shown that our system can act as a fast, responsive, and usable interface on its own. Our results are very promising since our system operated in less than ideal conditions: we used background subtraction for crude foreground estimation, and there were objects moving in the background. Since our method is not bound to a specific foreground model, and any background and foreground information collected by a program can be used as an input to our system, computational load may be significantly reduced, which is quite important in the context of human-computer interaction.

In our experiments, users were able to play the BlockEscape game and browse web pages effectively, demonstrating that our method works well in a real world context. Our method also offers new opportunities for people with disabilities. Used as a stand-alone interface or with another computer vision system, our technique could help facilitate a richer interactive experience for quadriplegic users who can control their head motion by enabling them to browse the web, enter text, or play a computer game. This is an important aspect of our contribution, since people with disabilities are dependent upon interface systems for their interaction with computers.

Using the eyes as possible cues to our method could make it even more robust. The eyes can be detected with a variety of methods (e.g., [4]), and we could use their axis of orientation to provide additional estimates to our system. To combine these estimates, the use of democratic integration [14] rather than equation 7 may be desirable in this case, since democratic integration can weigh system components relative to their current performance in an extremely effective fashion. These ideas could extend to tracking other facial features, such as nostril tracking. Other extensions would be to handle out of plane head rotation by fitting conic sections rather than circles. However, modeling such head turns is not as relevant to human-computer interaction, since users typically do not wish to interact with a computer that is not the focus of their attention.

In summary, we have introduced an efficient, accurate method for fast head tilt angle estimation. Our system operates on video data in real time with minimal computational requirements. Experiments with our system have shown that it is easy to use as an input and control device on its own and it has the potential to become an important part of a robust human-computer interaction system.

Acknowledgment

The authors would like to thank the subjects for their enthusiasm and patience in testing our method. Funding was provided by the National Science Foundation (IIS-0308213, IIS-039009, IIS-0093367, P200A01031, and EIA-0202067).

References

1. I.A. Essa and A.P. Pentland. Coding, analysis, interpretation, and recognition of facial expressions. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 19(7):757–763, July 1997.
2. J.-J. J. Lien, T. Kanade, J. Cohn, and C.-C. Li. Automated facial expression recognition based on FACS action units. In *Proceedings of the Third IEEE International Conference on Face and Gesture Recognition*, pages 390–395, April 1998.
3. M.J. Lyons. Facial gesture interfaces for expression and communication. In *Proceedings of the 2004 IEEE International Conference on Systems, Man and Cybernetics*, volume 1, pages 598–603, October 2004.
4. J.J. Magee, M.R. Scott, B.N. Waber, and M. Betke. Eyekeys: A real-time vision interface based on gaze detection from a low-grade video camera. In *Proceedings of the 2004 Conference on Computer Vision and Pattern Recognition Workshop (CVPRW'04)*, volume 10, pages 159–166, July 2004.
5. A. Raouzaiou, N. Tsapatsoulis, V. Tzouvaras, G. Stamou, and S. Kollias. A hybrid intelligent system for facial expression recognition. In *Proceedings of the European Symposium on Intelligent Technologies, Hybrid Systems and their implementation on Smart Adaptive Systems (Eunite 2002)*, September 2002.
6. G. R. Bradski. Computer vision face tracking for use in a perceptual user interface. *Intel Technology Journal*, Q2:15pp, 1998.
7. B. Schiele and A. Waibel. Gaze tracking based on face color. In *International Workshop on Automatic Face- and Gesture-Recognition*, pages 344–349, 1995.
8. K. Schwerdt and J. L. Crowley. Robust face tracking using color. In *Proceedings of the Fourth IEEE International Conference on Automatic Face and Gesture Recognition*, Grenoble, France, March 2000.
9. H. Rowley, S. Baluja, and T. Kanade. Rotation invariant neural network-based face detection. In *Proceedings of the 1998 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'98)*, page 38, June 1998.
10. G. Shakhnarovich, P. A. Viola, and M. Baback. A unified learning framework for real time face detection and classification. In *Proceedings of the Sixth IEEE International Conference on Automatic Face and Gesture Recognition*, pages 16–26, May 2004.
11. J. Sochman and J. Matas. AdaBoost with totally corrective updates for fast face detection. In *Proceedings of the Sixth IEEE International Conference on Automatic Face and Gesture Recognition*, pages 445–450, May 2004.
12. B. Wu, H. Ai, C. Huang, and S. Lao. Fast rotation invariant multi-view face detection based on real AdaBoost. In *Proceedings of the Sixth IEEE International Conference on Automatic Face and Gesture Recognition*, pages 79–84, May 2004.
13. M. Betke, J. Gips, and P. Fleming. The Camera Mouse: Visual tracking of body features to provide computer access for people with severe disabilities. *IEEE Transactions on Neural Systems and Rehabilitation Engineering*, 10(1):1–10, March 2002.
14. J. Triesch and C. von der Malsburg. Democratic integration: Self-organized integration of adaptive cues. *Neural Computation*, 13(9):2049–2074, September 2001.

Attention Monitoring Based on Temporal Signal-Behavior Structures

Akira Utsumi, Shinjiro Kawato, and Shinji Abe

ATR Intelligent Robotics and Communication Laboratories, Kyoto 619-0224, Japan

Abstract. In this paper, we discuss our system that estimates user attention to displayed content signals with temporal analysis of their exhibited behavior. Detecting user attention and controlling contents are key issues in our "networked interaction therapy system," which effectively attracts the attention of memory-impaired people. In our proposed system, user behavior, including facial movements and body motions ("beat actions"), is detected with vision-based methods. User attention to the displayed content is then estimated based on the on/off facial orientation from a display system and body motions synchronous to auditorial signals. This attention monitoring mechanism design is derived from observations of actual patients. Estimated attention level can be used for content control to attract more attention of the viewers to the display system. Experimental results suggest that the content switching mechanism effectively attracts user interest.

1 Introduction

For a computer system to efficiently support human activities, it has to recognize users' behaviors and understand their intentions. Therefore, the ability to recognize human behavior by using sensors embedded in living environments is becoming an increasingly important research endeavor.

In human-computer interaction tasks, for instance, attracting and keeping the motivation of users become significant for extracting positive reactions. To achieve this, the system has to estimate individual concentration levels and control the style and amount of displayed information. Since reactions vary from user to user, such control should occur dynamically.

The same situation prevails in our "networked interaction therapy system," which entertains and effectively attracts the attention of memory-impaired people and lightens the burden of helpers or family members [1]. "Networked interaction therapy" requires that the system provide remote communication with helpers and family members as well as video contents and other services. To attract the attention of users for long periods of time, the system has to detect user behaviors and control the order and timing of provided services based on estimates of concentration levels.

Various sensory devices can be used to detect user behaviors. Vision-based detection of human behavior fits our needs since it does not require any special

N. Sebe, M.S. Lew, and T.S. Huang (Eds.): HCI/ICCV 2005, LNCS 3766, pp. 100–109, 2005.

attachments [2–5]. The system can remotely detect human motions, and users do not need awareness of the sensory systems.

In this paper, we describe a method to estimate user attention based on facial orientation and body motions related to visual/auditorial stimuli in displayed video/audio contents detected with a vision-based system. In the proposed method, if user attention strays from the target medium, the system changes the contents of the displayed information or prompts the user to lure him or her back. We believe that such content control enables users to receive services in a manner appropriate to their interests or preferences.

In the next section, we briefly introduce our "networked interaction therapy project." Section 3 outlines our observations. Section 4 summarizes the framework of our attention estimation, and Section 5 shows the experimental results. In Section 6, we give our conclusions.

2 Networked Interaction Therapy Project

Memory is frequently impaired in people with such acquired brain damage problems as encephalitis, head trauma, subarachnoid haemorrhage, dementia, cerebral vascular accidents, etc. Such people have difficulty leading normal lives due to memory impairment or higher brain dysfunction; consequently, the requirement for constant care and attention creates a heavy burden on their family. Networked interaction therapy, a term that denotes our method for relieving the stress suffered by memory-impaired people and their family members, creates easy access to the services of networked support groups [1]. There has been a recent spate of similar studies in this field by several researchers [6, 7].

The main goal of networked interaction therapy is to support the daily activities of memory-impaired people by enhancing their quality of life and reducing the burden on their family. For this purpose, we give networked interaction therapy the capability to automatically detect the intentions of a memory-impaired person. This provides the necessary information/entertainment for guiding the individual to a more comfortable condition on behalf of his or her family before

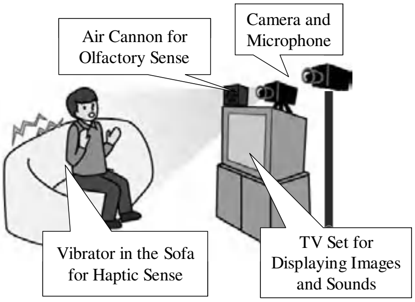

Air Cannon for Olfactory Sense
Camera and Microphone
Vibrator in the Sofa for Haptic Sense
TV Set for Displaying Images and Sounds

Fig. 1. Networked Interaction Therapy System

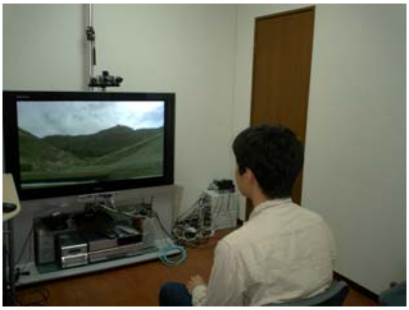


Fig. 2. Experimental Environment

the occurrence of such behavioral problems as wandering at night, incontinence, fits of temper, and so on. Figure 1 illustrates an example of a terminal used for networked interaction therapy. Currently, we plan to provide our service by using a large screen TV and a TV-top box that controls Internet communication, cameras, a microphone, and sensory media. We set up an experimental room that imitates a normal living environment with a TV, an audio set, normal lighting equipment, doors, windows etc. and installed a prototype system (Figure 2). All experiments in Section 5 were performed in this room.

3 Observing the Behavior of Memory-Impaired People Watching Videos

Prior to the system's actual implementation, we observed the behavior of actual memory-impaired patients watching pre-recorded video programs on TV. Observations were performed with a total of eight subjects in two groups: group 1: three subjects at their own home, and group 2: five subjects at a medical facility ("care house")). Table 1 briefly profiles the eight subjects.

Table 1. Brief profile of eight subjects

group	Subjects	Age	Case History	Preference
1	A	62	cerebral contusion	board game ("go"), songs (oldies)
	B	81	Alzheimer's disease	train travel, songs (oldies)
	C	69	cerebral infarction	baseball games, songs (oldies)
2	D	83	cerebral dementia	children's songs
	E	90	senile dementia	N/A
	F	89	Alzheimer's disease	movie
	G	89	Alzheimer's disease	songs (oldies)
	H	92	senile dementia	N/A

We prepared four to ten video contents for each subject and showed them by switching from one to the next every two to eight minutes (four contents every seven to eight minutes for group 1, ten contents every two minutes for group 2).

Table 2 shows some major results in group 1, where it can be seen that behavior varies for each video program in terms of total watching time and

Table 2. Observation Results for group 1

	video contents	memoir video	music clip	hobby	news shows
Watching time (sec)	Subject A	115	46	109	5
	Subject B	120	111	115	110
	Subject C	105	120	120	100
Frequency of looking away from TV (number)	Subject A	4	many	4	many
	Subject B	0	2	3	3
	Subject C	4	2	0	3
Total time of hand moving (sec)	Subject A	0	92	0	0
	Subject B	0	97	0	0
	Subject C	0	0	0	0

the number of “looking in another direction” actions. This should reflect user preference for each program, and the results suggest that the facing (or gazing) direction is a significant cue for estimating user attention. In addition, such hand motions as “keeping time with hand(s)” were observed for multiple subjects, especially when they watched music programs. Therefore, synchronous actions to auditorial signals are also considered indications of user attention.

Although the behavior of group 2 was not as remarkable as group 1, two subjects (D and H) did display similar preferential reactions.

Other frequently observed reactions were as follows:

Positive reactions:

- pointing or gesturing to the TV
- laughing at presented TV programs
- singing with presented music

Negative reactions:

- looking at different orientation than TV
- grasping surrounding object(s) and moving it (them)

According to the above observations, in this paper we focus on extracting user attention based on facial orientations and synchronous body motions into auditorial signals.

4 Attention Monitoring

Since the human vision system plays an important perceptual and communicative role in our daily life, gaze can be a strong cue for estimating user attention. In our attention monitoring, we first focus on the orientation of the face. Strictly speaking, facial orientation is different from human gaze due to the lack of eye information. However, in most cases, a loss of visual attention is accompanied by head movement. Therefore, we still consider facial orientation information useful for attention estimation.

Another means for estimating human attention is body motions, especially synchronous motions in visual/auditorial signals that appear in displayed content, which should represent user absorption with the content. In this paper, we deal with the extraction of synchronous beat actions from music programs.

Detected attention can be used for controlling audio/video contents displayed to users. For example, such positive synchronous reactions as looking at the TV and/or keeping time with their body can be used as the basis to make a system that shows the contents longer. In contrast, such negative behavior as turning his/her head away from the TV can be used as an trigger for the system to switch contents.

4.1 Vision-Based Behavior Extraction

Face Orientation Tracking

Vision-based face tracking is a popular research field that many researchers have investigated [8–10]. However, few systems have achieved both robustness and high-speed processing. We developed a real-time and stable face tracking system via SVM-based face detection coupled with prefiltering with SSR filters.

In our system we employ a pattern matching algorithm using rectangle templates called SSR filters for face candidate extraction. This process can operate quite fast with integral images. Since we do not use color information, the algorithm is not affected by the color temperature of the lighting. A Support Vector Machine (SVM) is applied to determine whether a candidate is a face. To minimize the effects of hair styles, beards, and other facial hair, the forehead and mouth regions are excluded from training patterns for SVM. Once located, we track them with a "between-the-eyes" template [11] instead of tracking the eyes themselves. The resulting pattern is fairly stable despite changes in facial expression. Facial orientation is then estimated based on the relative locations of the eyes and the tip of the nose [12], which roughly estimates the direction in which the subject is looking.

Figure 3 shows an example of face tracking. Graphics are overlayed on the facial image, indicating detected eyes and the tip of the nose. The detected facial orientation is indicated by the heavy line in the narrow region along the top of the image. Figure 4 shows the results of face orientation estimation (horizontal). Here ±1 in the vertical axis denotes the limit of detection, and for this subject it corresponds to about ±45 degrees.

Detecting Body Motion

For detecting body motion, we currently employ an interframe subtraction scheme. Here, the magnitude of motion is measured as the number of moving pixels N.

$$N_t = \sum_i s_{i,t}, \tag{1}$$

$$s_{i,t} = \begin{cases} 0 \ ((\boldsymbol{c}_{i,t} - \boldsymbol{c}_{i,t-1})'(\boldsymbol{c}_{i,t} - \boldsymbol{c}_{i,t-1}) < \text{threshold}) \\ 1 \ (\text{otherwise}) \end{cases} \tag{2}$$

Here, $\boldsymbol{c}_{i,t}$ denotes a color vector of the i-th pixel in an observed image at time t.

This method is simple but still valid for extracting temporal patterns of motion magnitude. For instance, "keeping the beat" action can be detected as

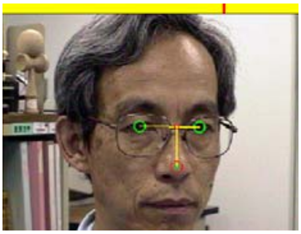


Fig. 3. Example of facial orientation estimation

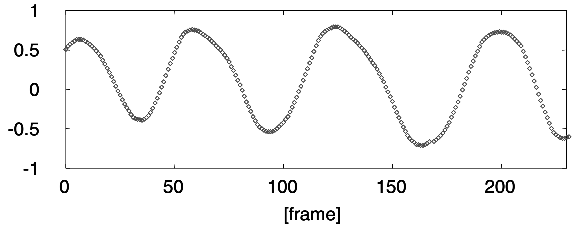
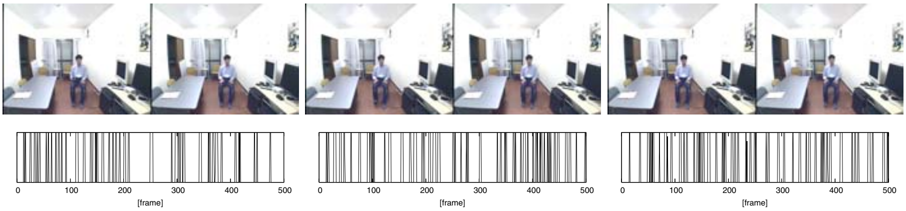


Fig. 4. Estimation of facial orientation

"frozen" points where the motion suddenly stops in a motion magnitude sequence.

$$\text{beat} = \begin{cases} 1 \ (\frac{dN_{t-1}}{dt} < 0 \text{ and } \frac{dN_t}{dt} \simeq 0) \\ 0 \ (otherwise). \end{cases} \qquad (3)$$

Figure 5 shows an example of extracted beat action. Here, the person was expressing the beat by hand (left figure), foot (middle figure), and nodding (right figure). Our method can extract all of them in the same manner.



Hand Motion Foot Motion Head Motion

Fig. 5. Example: Extracted Beat Actions

4.2 Contents Control

Figure 6 shows a process diagram of our contents control mechanism. Here, the display system shows video contents to users whose behavior (facial orientation and body motions) is observed by cameras and other sensory systems. Simultaneously, displayed content characteristics are extracted by the system. Currently, only the beat structures of audio signals are extracted in this process. After beat structure extraction, the temporal correlation between the beat structure and user motion is evaluated. User attention to the displayed contents is then estimated from the correlation and facial orientations.

If the system recognizes a loss of attention to the presented medium, it switches contents to regain user attention. Figure 7 shows the results of such content switching for a sample sequence, where the system controls the video contents according to facial orientation.

4.3 Beat Tracking

"Beat" is one of the most fundamental properties employed by humans for recognizing music. Therefore, a significant relation should exist between beat and the behavior of people listening to music. For instance, recently Shiratori et al. analyzed motion-captured dance movements in conjunction with musical rhythm analysis [15]. In our perspective, we focus on user actions synchronous to the rhythmic property of displayed music.

Beat tracking is a major topic of research interest among member of the acoustic audio processing community, and many trials have examined it. One major approach is based on frequency analysis [13, 14]. In this work, we also

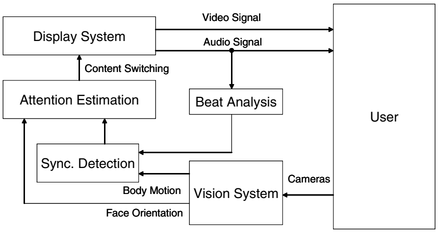


Fig. 6. Attention Estimation using Vision-Based Tracking

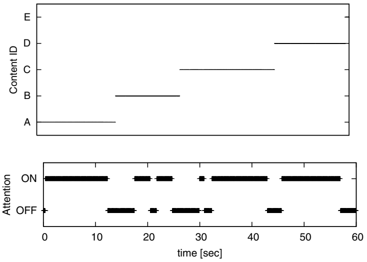


Fig. 7. Content Switching Based on Facial Orientation

formalized beat structure detection based on a set of signal power banks consisting of several different frequencies. Input signals are first fed into a frequency analyzer (FFT) and separated into discrete filter banks. In our current implementation, guard frequencies are 0-250 Hz, 250-500 Hz, 500-1 kHz, 1-2 kHz, and 2-4kHz.

For each bank, the power transition is examined, an envelope signal is produced, and the beat structures are extracted as a rising point using differential operations. Figure 8 shows an example of structure extracted from a piece of popular music.

5 Experiments

We conducted the following experiments to confirm the feasibility of our proposed method. In contrast with the observation experiment in Section 3, all experiments described in this section were performed using healthy adults.

First, we applied our facial orientation detection to a sequence of head rotation images. Figure 9 denotes the results of attention estimation. Here, the subject mainly watched our TV system and sometimes looked down at a magazine on his lap. Figure 10 shows typical sample images of the two states: “attention on (looking at the TV set)” and “attention off (looking in other directions).” The results indicate that in this case, our system accurately detected the loss of attention away from the TV system.

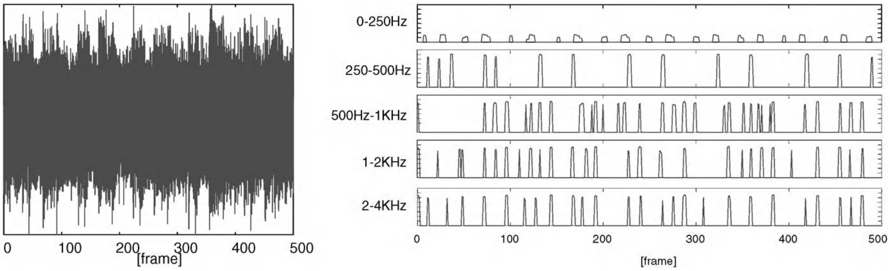


Fig. 8. Example: Input Audio Signal (left) and Extracted Beat Structure (right)

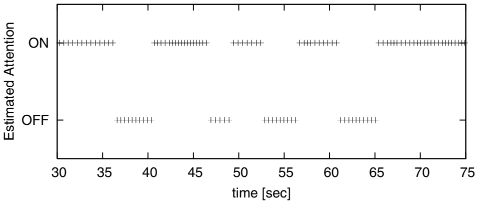


Fig. 9. Results of Attention Estimation

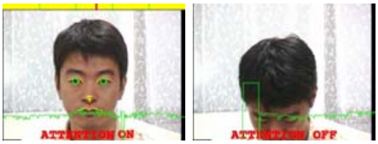


Fig. 10. Two states: "attention on (looking at TV set)" and "attention off (looking at other objects)"

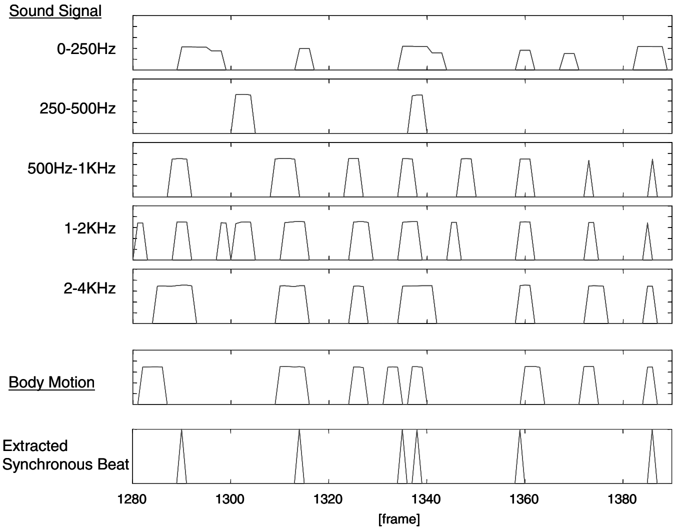


Fig. 11. Synchronous 'beat' behavior extraction

Next we performed detection of synchronous beat behavior related with displayed musical signals. Figure 11 shows both the detected beat from the displayed musical signal and the human beat motions. Here, several popular songs were used for the experiment. At the bottom of Figure 11, we find the extracted synchronous motion (beat action). Furthermore, Fig. 12 shows the result of synchronous/non-synchronous behavior detection. Here on the left side, a subject is listening to music and is instructed to keep the beat. On the right side, he moves his hand randomly, indicating that our system can clearly distinguish between the two states.

Finally, we examined the effect of content control based on attention monitoring (here, only face orientation is used.) We placed two TV sets in front of a subject. One employed content switching mechanism based on facial orientation. The other had no attention monitoring; video contents were just switched at constant intervals in a predefined order. Table 3 compares the total viewing time of the two TV sets for six subjects. As we can see, subjects watched the controlled TV significantly longer, suggesting that content switching can effectively attract user interest.

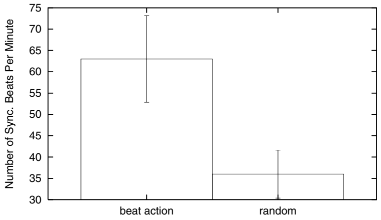


Fig. 12. Distinguishing Beat Action and Non-Beat Action

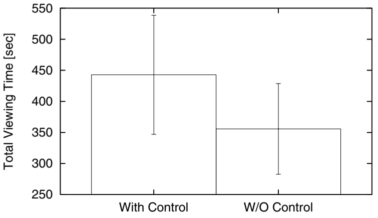


Fig. 13. TV Watching Time with/without Content Control

Table 3. Watching time and number of missed detections

Subject		a	b	c	d	e	f	Avg.
Watching Time (sec)	with control	528	526	439	422	444	294	442*
	without control	289	314	391	288	336	426	340*
detection error	false-negative	0	3	6	0	3	1	2.2
	false-positive	2	0	0	4	0	0	1

*p=0.06

6 Conclusions

We described vision-based techniques for estimating human attention based on body motions and facial orientation. We employed an SVM to model human faces and determined facial orientation by using the geometrical relation among the eyes and the tip of the nose. Body motions ("beat actions") were also extracted and examined in terms of synchronization with the beat elements of displayed musical signals. We applied these techniques to estimate the attention of users. In the experiments, we controlled the timing of content switching to attract user attention to the presented video contents. Experimental results suggest that the content switching mechanism effectively attracts user interest. Such capabilities are very promising for our "networked interaction therapy system," which effectively attracts the attention of memory-impaired people and lightens the burden on helpers or family members.

Future works include enhancing the face and body tracking algorithms and carrying out studies that involve a large number of participants.

This research was supported in part by the National Institute of Information and Communications Technology.

References

1. N. Kuwahara, K. Kuwabara, A. Utsumi, K. Yasuda, and N. Tetsutani. Networked interaction therapy: Relieving stress in memory-impaired people and their family members. In *Proc. of IEEE Engineering in Medicine and Biology Society*, 2004.

2. Akira Utsumi and Jun Ohya. Multiple-camera-based human tracking using non-synchronous observations. In *Proceedings of Fourth Asian Conference on Computer Vision*, pages 1034–1039, 2000.
3. D. M. Gavrila and L. S. Davis. 3-d model-based tracking of humans in action: a multi-view approach. In *Proc. of Computer Vision and Pattern Recognition*, pages 73–80, 1996.
4. Q. Cai and J. K. Aggarwal. Tracking human motion using multiple cameras. In *Proceedings of 13th International Conference on Pattern Recognition*, pages 68–72, 1996.
5. Jakub Segen and Sarma Pingali. A camera-based system for tracking people in real time. In *Proceedings of 13th International Conference on Pattern Recognition*, pages 63–67, 1996.
6. Low J., Schietecat T., Kwok T F., and Lindeboom L. Technology applied to address difficulties of alzheimer patients and their partners. In *Proceedings of the conference on Dutch directions in HCI, ACM*, page 18, 2004.
7. http://www.digitalangelcorp.com/default.asp.
8. J. Yang, R. Stiefelhagen, U. Meier, and A. Waibel. A real-time face tracker. In *Proc. 3rd IEEE Workshop on Application of Computer Vision*, pages 142–147, 1996.
9. J. C. Terrillon, M. David, and S. Akamatsu. Automatic detection of human faces in natural scene image by use of skin color model and invariant moment. In *Proc. of IEEE Int. Conf. on Automatic Face and Gesture Recognition*, pages 112–117, 1998.
10. J. Heinzmann and A. Zelinsky. 3-d facial pose and gaze point estimation using a robust real time tracking paradigm. In *Proc. of IEEE Int. Conf. on Automatic Face and Gesture Recognition*, pages 142–147, 1998.
11. S. Kawato and N. Tetsutani. Real-time detection of between-the-eyes with a circle frequency filter. In *Proc. of ACCV 2002*, pages II:442–447, 2002.
12. D. O. Gorodnichy. On importance of nose for face tracking. In *Proc. of IEEE 5th Int. Conf. on Automatic Face and Gesture Recognition*, pages 188–193, 2002.
13. E. D. Scheirer. Tempo and beat analysis of acoustic musical signals. *Journal of Accoustical Society of America*, 103(1):588–601, 1998.
14. M. Goto. An audio-based real-time beat tracking system for music with or without drum-sounds. *Journal of New Music Research*, 30(2):159–171, 2001.
15. T. Shiratori, A. Nakazawa, and K. Ikeuchi. Detecting dance motion structure through music analysis. In *Proc. of Sixth International Conference on Automatic Face and Gesture Recognition*, pages 857–862, 2004.

Action Recognition with Global Features

Arash Mokhber, Catherine Achard, Xingtai Qu, and Maurice Milgram

Laboratoire des Instruments et Syst□mes d□Ile de France (LISIF)
Universit□ Pierre et Marie Curie, 4 place Jussieu, Case courrier 252,
75252 Paris cedex 05
arash.mokhber@etu.upmc.fr, {achard,milgram}@ccr.jussieu.fr,
xingtai.qu@lisif.jussieu.fr

Abstract. In this study, a new method allowing recognizing and segmenting everyday life actions is proposed. Only one camera is utilized without calibration. Viewpoint invariance is obtained by several acquisitions of the same action. To enhance robustness, each sequence is characterized globally: a detection of moving areas is first computed on each image. All these binary points form a volume in the three-dimensional (3D) space (x,y,t). This volume is characterized by its geometric 3D moments. Action recognition is then carried out by computing the Mahalanobis distance between the vector of features of the action to be recognized and those of the reference database. Results, which validate the suggested approach, are presented on a base of 1662 sequences performed by several persons and categorized in eight actions. An extension of the method for the segmentation of sequences with several actions is also proposed.

1 Introduction

Human activity recognition has received much attention from the computer vision community since it leads to several applications such as video surveillance for security, human-computer interaction, entertainment systems and monitoring of patients or old people, in hospitals or in their apartments. The activity recognition problem is generally divided in two steps. The first, consists of detecting and tracking the person in motion while the second concerns the recognition.

The focus of this work is on the second problem which is recognizing actions of everyday life, such as walking, sitting on a chair, jumping, bending or crouching, using an appearance-based model. Invariance to the view point is carried out by multiple views of the same action.

The recognition system developed in this work is based on a global representation of the actions by 3D volumes (*x,y,t*) and a characterization of these volumes by 3D geometrical moments, which will be used as input for the recognition process. This global representation of actions has some advantages. In this method all information concerning an action is included in only one vector, that allows to recognize actions without systems such as finite state machines [6] or hidden Markov models [15]. In addition, information related to the dynamics of the action is contained in this 3D volume. There is no need to extract motion vectors in each image of the sequence, which is a difficult stage to realize due to the non-rigidity of the human body.

Human activity recognition can be used in many applications of Human Computer Interactions and particularly those concerning the surveillance of human. A system raising the alarm when a person is falling or presenting an unusual activity (being in a

N. Sebe, M.S. Lew, and T.S. Huang (Eds.): HCI/ICCV 2005, LNCS 3766, pp. 110□119, 2005.

lying position all the day) can thus be developed. Electronic equipments can also be set up in public areas to detect crowd movement or fight and thus attract the attention of a human operator assigned to the video surveillance.

2 Related Work

Human action recognition is an important area of research that has led to different surveys [3 and 13]. Different approaches have been employed. They can be divided in four groups:

- 3D approaches without shape model;
- 3D approaches with volumetric models such as elliptical cylinders, and spherical models;
- 2D approaches with explicit shape model such as stick figure, and 2D ribbons; and
- 2D approaches without explicit shape model.

Since the human body is not a rigid object and may present a multitude of postures for the same person, a robust modeling is difficult to obtain. Therefore, appearance models are utilized rather than employing geometric models.

Among all existing 2D approaches without models, the ones that have considered the sequence as a succession of images are cited. Martin and Crowley [6] recognize hand gestures using three modules: 1) tracking, 2) hand classification in various postures, and 3) gesture recognition with a finite state machine. In Ref. [12], motion parameters are computed on each image, the recognition is then carried out with the hidden Markov models. This method was first used by Yamato et al. [15] to recognize tennis actions from binary image sequences.

Another approach utilizes more complex classifiers and models of action. For example, Hongeng et al. [4] proposed an automatic surveillance system with the application of Bayesian network. While taking image features of the tracked moving regions as input, mobile object properties are computed, and used to determine the probability that a scenario occurs due to several layers of na□ve Bayesian classifiers.

In the previous approaches, actions are seen as a sequential set of separated images and are not considered globally. Recently, Bobick and David [1] have presented a view-based approach to the recognition of human movement. They first construct a binary Motion Energy Image (MEI) which represents places, where motion has occurred in an image sequence. Next, they generate a scalar-valued Motion History Image (MHI), where intensity is a function of how recent the motion is. Both characteristics are constructed globally throughout the sequence. Given a set of MEIs and MHIs for each view/movement of aerobic exercises, they compute statistical descriptions of these images using 7 Hu moments.

The global study of the sequence can also be conducted by examining the empirical distributions of some characteristics. Chomat and al. [2] use space-time local appearance to recognize actions. The exit of a bench of 12 space-time filters provides a vector of measurement in each pixel. Joint statistics on these vectors represented by a multidimensional histogram make it possible to carry out the recognition of the actions. In Ref. [14], actions are also considered as long-term temporal objects and are characterized on several temporal scales using motion detection. The measurement between two actions, is therefore, the distance between empirical distributions of these features. Masoud et al. [7] extract from each image of the sequence features relative to movement with the help of an infinite impulse response filter. From these images an analysis in principal component is performed to reduce the dimension of the problem for the recognition stage.

The system of recognition proposed in this work utilizes a global representation of actions by 3D volumes (*x,y,t*) and a characterization of these volumes by geometrical moments. This avoids extracting information relating to the dynamics of the scene, since it is implicitly present in the vector of features. The first stage of the system consists of creating these 3D volumes and thus, extracting moving regions in each image.

3 Motion Detection

As previously mentioned, the first stage of the activity recognition process consists of detecting moving areas. Therefore, the current image is compared at each instant with a reference image continuously updated. A second stage is also necessary to remove shadows that eventually are present in the scene.

To authorize multi-modal, the history of each pixel of the reference image is modeled by a mixture of K Gaussian distributions [8 and 11]. The probability of observing the value of the current pixel X_t is then given by:

$$P(X_t) = \sum_{i=1}^{K} w_{i,t} * \mathcal{N}(X_t, \mu_{i,t}, \Sigma_{i,t}) \tag{1}$$

where, for i^{th} Gaussian at time t, $w_{i},_{\mathrm{t}}$ is the weight of the Gaussian, $\mu_{i,t}$ is its average value and $\Sigma_{\mathrm{i,t}}$ its covariance matrix. $\mathcal{N}(\ )$ is the Gaussian density probability function which is defined, as follows:

$$\mathcal{N}(X, \mu, \Sigma) = \frac{1}{(2\pi)^{n/2} |\Sigma|^{1/2}} \exp(-\frac{1}{2}(X-\mu)^T \Sigma^{-1} (X-\mu)) \tag{2}$$

where n is the dimension of the vector (3 in this case because this work is based on color images with 3 channels).

Initialization of the Gaussian mixture is carried out by the K-means algorithm on the first 40 images of the sequence where it is assumed that no movement occurs. Each pixel of the background is modeled by K=2 Gaussian. It appears that this is a good compromise between computing time and the quality of results. For each new pixel X_t, its nearest Gaussian is searched. If the distance between this Gaussian and the current pixel is lower than a threshold, the latter is assigned to the background, otherwise, it is classified as a pixel belonging to a moving object. To deal with lighting changes during the process of acquisition, pixels labeled, as background, are used to update the reference image and thus, the Gaussian they are closest to:

$$\begin{aligned} \mu_t &= (1-\alpha)\mu_{t-1} + \alpha X_t \\ \Sigma_t &= (1-\alpha)\Sigma_{t-1} + \alpha(X_t - \mu_t)(X_t - \mu_t)^T \end{aligned} \tag{3}$$

where α was empirically fixed at 0.1.

This method leads to good results of detection. However, shadows are often detected as a moving object. As a result, the shapes of the detected silhouettes are significantly deteriorated and this disturbs the algorithm of action recognition. A second stage is then employed to solve this problem. In this work it is assumed, as in Ref. [8], that shadows decrease brightness of pixels but do not affect their color.

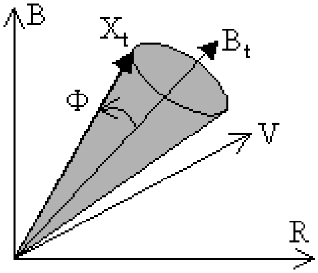
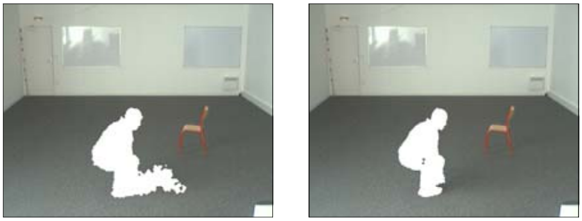


Fig. 1. Shadow is defined as a cone

Thus, the angle *Φ* between the color vector of the current pixel X_t and that of the corresponding background pixel B_t (average of Gaussian nearest to the pixel) is an effective parameter to detect shadows. Note that if *Φ* is below a threshold, and the brightness of the current pixel is smaller than the brightness of the background, it is assumed that the pixel corresponds to shadows. Shadow is thus defined as a cone around the color vector corresponding to the background, as shown in Figure 1.

At the end of the process, only pixels detected as moving by the mixture of Gaussian and which do not correspond to shadow are preserved. Several morphological operations end this stage and lead to a binary map of moving pixels, for each image. As can be seen in Figure 2, good detection results are obtained. However, in some images of our sequences, the detection is not as good due to the close similarity between background colors and those of the moving person. It is observed that the space-time characterization of these binary images, presented in Section 4, is robust enough to lead to good action recognition results.



(a) Image difference (b) With shadow modeling

Fig. 2. Motion detection

4 Features Extracted from Sequences

The features representative of the sequence from all the binary images need to be extracted. A global representation of actions is selected to simplify the recognition process and to enhance robustness during this stage. Therefore, an action is first represented by the 3D volume comprised of all moving points detected (*x,y,t*). This space-time volume contains a lot of information, such as, the silhouette of the person in each image or the action dynamics. To characterize this volume, without extracting (and separating) different information (posture, movement, etc), 3D geometrical moments are considered.

Let $\{x,y,t\}$ be the set of points of the binary volume where x and y represent the space coordinates and t, the temporal coordinate. The moment of order $(p+q+r)$ of this volume is determined by:

$$A_{pqr} = E\left\{x^p y^q t^r\right\} \tag{4}$$

where E$\{x\}$ represents the *expectation* of x. In order to use features invariant in translation, the central moments are defined by:

$$AC_{pqr} = E\left\{(x-A_{100})^p (y-A_{010})^q (t-A_{001})^r\right\} \tag{5}$$

These moments must also be invariant to the scale to preserve invariance with the distance of action or with the size of people. This invariance could be obtained using the Hu moments [5]. In the present study these moments were not retained because they lead to invariance in rotation which is not desirable in this application. For example, a person being upright or lying on the ground would then have the same features. A direct normalization on the different axes, by dividing each component by the corresponding standard deviation is not desirable because it leads to an important loss of information, that is, the shape of the binary silhouettes appears to be rounder. Also, an identical normalization is carried out on the first two axes, while the third (time) is normalized, separately. The normalization performed by preserving the ratio of width-to-height of the binary silhouettes is thus obtained by the following relation:

$$M_{pqr} = E\left\{\left(\frac{x-A_{100}}{A_{200}^{1/4} A_{020}^{1/4}}\right)^p \left(\frac{y-A_{010}}{A_{200}^{1/4} A_{020}^{1/4}}\right)^q \left(\frac{t-A_{001}}{A_{002}^{1/2}}\right)^r\right\} \tag{6}$$

5 Presentation of the Sequence Database

A sequence database comprising 8 actions is considered:

(1) "to crouch down",
(2) "to stand up",
(3) "to sit down",
(4) "to sit up",
(5) "to walk",
(6) "to bend down",
(7) "to get up from bending", and
(8) "to jump".

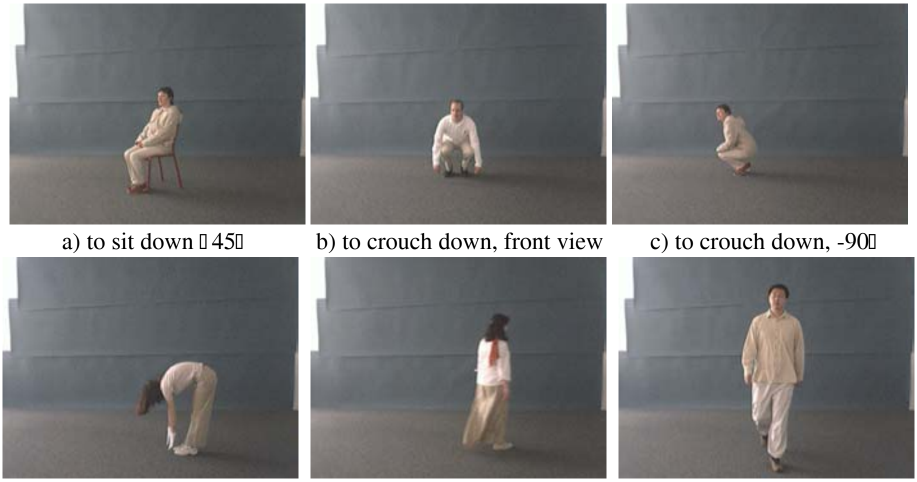
Various viewpoints were acquired for each action. The front, 45° and 90° views were captured while others were synthesized from the sequences already recorded (at -45°, at -90°) in order to allow later segmentation of the sequences in actions (Section 7). Each action was executed by 7 people, and repeated 230 times on average. The database comprises 1662 sequences. Presented Figure 3 are some examples of images of the database representing various actions and silhouettes of actors.

6 Recognition Results

For the recognition, the sequence database is divided into two disjointed sets: 1) a reference database, and 2) a test database. To test invariance of the method compared to people morphology, the reference database is made up of actions carried out by six people. The sequences achieved by the last person were assigned to the test database. The recognition is done by searching for the vector of features corresponding to the action to recognize the nearest vector in the reference database, using the Mahalanobis distance. The action to be recognized is then assigned to the class of the nearest vector. The method is examined by using a vector of features O composed of the 14 moments of 2nd and 3rd order:

$$O = \{M_{200}, M_{011}, M_{101}, M_{110}, M_{300}, M_{030}, M_{003},$$
$$M_{210}, M_{201}, M_{120}, M_{021}, M_{102}, M_{012}, M_{111}\}.$$

Note that the moment M_{020} is not calculated. This is due to the normalization which makes M_{020} inversely proportional to M_{200}. In addition, the moment M_{002} is always equal to 1.



a) to sit down □45□ b) to crouch down, front view c) to crouch down, -90□

d) to bend down □90□ e) to walk 135□ f) to walk, front view

Fig. 3. Some images of the database

Presented in Table 1 are the seven recognition rates obtained by placing each of the 7 persons in the test database, one by one. The average recognition rates, on the 8 actions, vary from 93.3% to 100% depending on the person. Thus, one may conclude that actions are well recognized, regardless of the person. Note that the person present in the test database is not at any time present in the base of reference. This shows that the characterization is relatively invariant to the silhouette of the person. The worst recognition rate (93.3%) is obtained for person 7. This is not surprising because this person presents a particular binary silhouette due to her clothing, as shown in Figure 4. This person wears a long skirt (and it is the only person with a skirt in the base). In spite of this characteristic, the recognition rate is still good, which demonstrates that the global characterization of actions is robust. An extension of the number of actors in the base is envisaged in order to improve classification results.

Table 1. Recognition rate using a features vector composed by 14 moments

Person	1	2	3	4	5	6	7
Rate	95.5	99.3	96.8	98.0	100	99.0	93.3

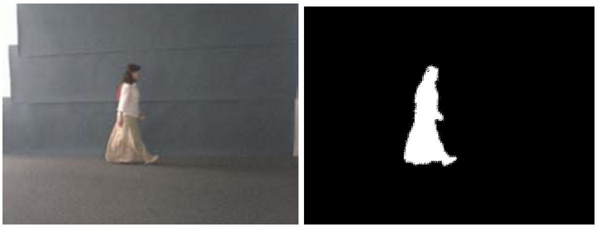


Fig. 4. Detection of a particular silhouette

Table 2 presents the average confusion matrix obtained by averaging the 7 confusion matrices corresponding to the different people. The most poorly recognized action, (92.5%), is action 6 ("to bend down",) sometimes confused with action 1 ("to crouch down",). Other actions ("to crouch down", "to walk", "to jump") are consistently well recognized with a recognition rate of 100%. It would be desirable to improve these results with a second level of classification exclusively committed to the separation of the ambiguous classes (1/6, 2/7, etc.). This, may enhance clarity between the boundaries of these classes [9].

Table 2. Confusion matrix obtained with a features vector composed of 14 moments

	1	2	3	4	5	6	7	8
1	100	0	0	0	0	0	0	0
2	0	96	0	0	0	0	4	0
3	0	0	98.4	0	0	1.6	0	0
4	0	1.3	0	96.5	0	0	2.2	0
5	0	0	0	0	100	0	0	0
6	7.5	0	0	0	0	92.5	0	0
7	0	2.9	0	0	0	0	97.1	0
8	0	0	0	0	0	0	0	100

7 Segmentation of Real Sequences

In the preceding study, geometrical moments on binary volumes, where temporal boundaries were known, were determined. Indeed, these volumes represent only one action. This assumes that the duration and the limits in time of each action carried out during video acquisition are known in order to extract the corresponding binary volume. However, for monitoring applications, this data is not a priori known. Nonetheless, the proposed model is used in order to segment real sequences where a person carries out several actions in a random order. It is necessary that the system is capable of cutting the sequence in to distinct actions and that it recognizes the nature of these latter.

By examining the database it is observed that the duration of the actions is variable based on their nature and the person who executes them. Thus, for some cases the action includes only 7 images ("to crouch down" for example) while others comprise 60 successive images (for the action" to walk"). During the segmentation, it may therefore be necessary, at any given time, to consider several assumptions of temporal duration of actions. Input data of the system consists of a succession of binary images. At every instant t_m, all the binary volumes $V_{m,i}$ are considered. These binary volumes are assumed to be centered at t_m and composed of the binary images which belong to the temporal window of length *i,* where *i* varies between 5 to 50 images. This is presented in Figure 6. Volume $V_{m,i}$ is thus computed for images between $t_m - i/2$ and $t_m + i/2$.

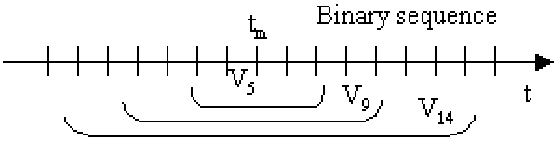


Fig. 5. Binary volumes centered about the instant of interest

The 14 geometrical 3D moments for all the space-time volumes $V_{m,i}$ are computed. These vectors that at any given time are 46, when the limits of the signal allow it, are then compared with those of the reference database composed of 1662 sequences (i.e., 1662 vectors). The distance $d_{m,i}$ between vectors is estimated with the Mahalanobis distance, as for the action recognition stage. Among all the distances $d_{m,i}$ computed, the smallest is selected. The corresponding action is assigned to times between $t_0 - i_0/2$ and $t_0 + i_0/2$. The process is reiterated (without labeling the time already assigned to a class) by searching the smallest distance among those which were not already selected, until all the points were assigned to classes.

Segmentation results are presented in Figure 6. Here the horizontal axis presents the temporal index and the vertical axis is the recognized actions by the method. The ground truth, realized by cutting the sequence manually is also presented to validate the results.

Most of the sequence was recognized similar to action 8, (i.e. "to jump".) Indeed, this action is characterized by its static character in space. This means that the majority of people do not jump "high" enough to mark a significant variation of the vertical component of the binary silhouettes. Therefore, when a person slows down or turns, it is rather recognized as jumping. Thus, a new action is added to the ground truth: action 9 which corresponds to "to half turn" and is present twice in the sequence.

The action "to half turn" of the ground truth was always recognized as "to jump" by the system, since this action is not present in the reference database. Nevertheless, one may conclude that for the final application which consists of studying the behavior of a person, it would be necessary to increase the number of actions to be recognized by the system.

The sequence was well cut and recognized for the actions "to walk", "to crouch down", "to stand up", "to sit down" and "to sit up". Thereafter, confusion between "to jump" and "to walk" is often made. An improvement could be obtained by better managing the temporal scales and by taking into account the probable duration of each action.

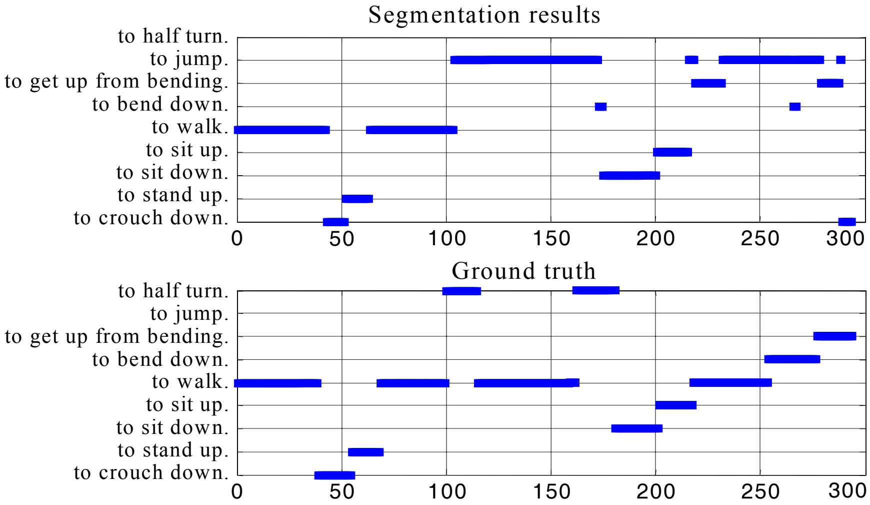


Fig. 6. New segmentation results

8 Summary and Conclusions

In this work, a method to recognize and to segment actions of everyday life is proposed. It can be used for example to monitor old people in their flat and raise the alarm when they are falling or presenting an unusual activity. Motion detection is obtained initially on each image due to a representation of pixels by mixtures of Gaussian and a particular treatment of shadows. The 3D volume constructed for each sequence from the binary images resulting from detection, is characterized by its 3D geometrical moments. Those are normalized in order to obtain invariance to the scale of actions and to the morphology of people executing them.

The proposed global characterization of sequences leads to an enhanced robustness and significantly improved results. A recognition rate of 97% on average was obtained from a database of 1662 sequences divided in to 8 actions and carried out by 7 people. The segmentation results could be improved by increasing the number of actions in order to incorporate all the transitional actions such as "to half turn", "to remain upright" or "to remain seated". An algorithmic enhancement could also be obtained by taking into account the probable duration of actions or by implementing more complex techniques for the segmentation (i.e., dynamic programming, relaxation, etc).

References

1. Bobick A.F. and Davis J.W., The recognition of human movement using temporal templates, IEEE transactions on pattern analysis and machine intelligence, vol.23, n3, mars 2001.
2. Chomat O. and Crowley J.L., Probabilistic recognition of activity using local appearance, Computer Vision and Pattern Recognition, Colorado, USA, 1999.
3. Gavrila D.M.,The visual analysis of human movement: a survey, Computer Vision and Image Understanding, vol. 73, n1, pp. 82-98, 1999.

4. Hongeng S., Bremond F., and Nevatia R., Bayesian framework for video surveillance application, International Conference on Computer Vision, Barcelona, Spain, 2000.
5. Hu M.K., Visual pattern recognition by moment invariants, IRE Trans. on Information Theory, IT-8: pp. 179-187, 1962.
6. Martin J. and Crowley J.L., An appearance based approach to gesture recognition, International Conference on Image Analysis and Processing, Florence, Italia, 1997.
7. Masoud O. and Papanikolopoulos N., Recognizing human activities, Conf. On advanced video and signal based surveillance 2003.
8. Porikli F. and Tuzel O., Human body tracking by adaptive background models and mean-shift analysis, IEEE International Workshop on Performance Evaluation of Tracking and Surveillance, Mars 2003.
9. Prevost L., Oudot L., Moises A., Michel-Sendis C. and Milgram M., Hybrid generative/discriminative classifier for unconstrained character recognition, Pattern Recognition Letters, to appear, 2005.
10. Rabiner L., A tutorial on hidden Markov models and selected applications in speech recognition, Proceedings of the IEEE, 77(2):257-285, February 1989.
11. Stauffer C. and Grimson W.E.L., Adaptive background mixture models for real-time tracking, Computer Vision and Pattern Recognition, vol.2, pp.246-252, June 1999.
12. Sun X. and Chen C. Manjunath B.S., Probabilistic motion parameter models for human activity recognition, International Conference on Pattern Recognition, pp. 443-446, 2002.
13. Wang J.J. and Singh S., Video Analysis of Human Dynamics - a survey, Real-time Imaging Journal, Vol. 9, Issue 5, Oct 2003, pp320-345.
14. Zelnik-Manor L., and Irani M., Event based analysis of video, Computer Vision and Pattern Recognition, pp. 123-130, 2001.
15. Yamato J., Ohya J. and Ishii K., Recognizing Human Action in Time-Sequential Images using Hidden Markov Models, Computer Vision and Pattern Recognition, Los Alamitos, IL, pp. 379-385, June 15-18, 1992.

3D Human Action Recognition Using Spatio-temporal Motion Templates*

Fengjun Lv, Ramakant Nevatia, and Mun Wai Lee

University of Southern California
Institute for Robotics and Intelligent Systems
Los Angeles, CA 90089-0273
{flv,nevatia,munlee}@usc.edu

Abstract. Our goal is automatic recognition of basic human actions, such as stand, sit and wave hands, to aid in natural communication between a human and a computer. Human actions are inferred from human body joint motions, but such data has high dimensionality and large spatial and temporal variations may occur in executing the same action. We present a learning-based approach for the representation and recognition of 3D human action. Each action is represented by a template consisting of a set of channels with weights. Each channel corresponds to the evolution of one 3D joint coordinate and its weight is learned according to the Neyman-Pearson criterion. We use the learned templates to recognize actions based on χ^2 error measurement. Results of recognizing 22 actions on a large set of motion capture sequences as well as several annotated and automatically tracked sequences show the effectiveness of the proposed algorithm.

1 Introduction and Related Work

Visual input is an important component of human computer interaction. In many applications, the objective is for human to *control* the responses of a computer without using keyboard and mouse [15]. In these examples, the actions of the human are deliberate and designed for easier understanding by the computer. Furthermore, the visual sensing is arranged to simplify the problems. The sensed images are of high resolution where hands and head motions are clearly visible and 2D analysis suffices [13] [4] [5]. In contrast are applications where the human is engaged in normal life activities and it is the computer's task to understand the human behavior and react according to the design goals of the system. Recognition of human actions, such as standing up, sitting down, pointing, waving arms etc. is essential to obtaining such capabilities and is the focus of this paper.

Human action recognition has been of interest to researchers in computer vision [2] [3] [9] [11] [13] [8] for many years. The problem can be defined as follows: given a motion sequence, the computer should identify the actions being

* This research was supported, in part, by the Advanced Research and Development Activity of the U.S. Government under contract No. MDA904-03-C1786

N. Sebe, M.S. Lew, and T.S. Huang (Eds.): HCI/ICCV 2005, LNCS 3766, pp. 120–130, 2005.

performed by the human subject. First difficulty is in estimating the positions of body parts; we do not address this problem but assume that such data is available, either from a Motion Capture (*MoCap*) system or from an automatic pose tracking system. Even if accurate 3D positions are available, action recognition is difficult due to the large dimensionality of pose space which not only increases computational complexity but may also hide key features of the action and due to significant spatial and temporal variations in an action for different, or even the same, subject.

Proposed methods can be divided into two categories with respect to the data types that they use: those based on 2D image sequences, e.g. [3] [11] [13] [8] and those based on 3D motion streams, e.g. [2] [9]. A 3D approach has many advantages over a 2D approach as the dependence on viewpoint and illumination has been removed. Nonetheless, most algorithms use a 2D approach because of the easy availability of video inputs and difficulty of recovering 3D information.

Our approach is based on 3D joint position trajectories. In this paper, our emphasis is on action recognition, given the 3D joint trajectories; we explore the effects of noise in automatically tracked data but robust tracking under complex conditions is a separate topic of research by itself. The actions we consider are *primitive* components that may be composed to form more complex actions. Examples of actions are: walk, sit, stand, wave hand, nod etc. One type of action is represented by one spatio-temporal motion template. Each template consists of a set of motion channels where each channel corresponds to the evolution of one joint coordinate. Channels with strong discriminative power are highlighted according to *Neyman-Pearson* criterion [14]. The recognition is based on the minimum weighted sum of χ^2 distance to each motion channel.

The proposed system achieves a recognition rate of 90.7% on 848 motion capture sequences consisting of 22 actions, showing the effectiveness of the proposed algorithms. Good results on annotated and automatically tracked videos show the potential of using real data.

2 Dataset and Pre-processing

We collected 848 MoCap sequences consisting of 22 Actions from Internet[1]. We also generated some sequences of 3D joint position from a 3D annotation software [7] and from an automatic 3D tracking software [7]. The generated data are much less accurate compared with MoCap. They are used in testing only to show that our algorithm can work on real data and that training on MoCap data transfers to video sequences; we do not claim to have solved the tracking problem as well.

Actions in these videos can be grouped into 3 categories according to the involved primary body parts: *leg+torso, arm, head*. The categorization is illustrated in Fig.1. Actions in the same group are mutually exclusive to each other, but actions from different groups can be recognized simultaneously. This allows us to execute logical queries such as *find the sequence in which the subject is walking while his head is nodding.*

[1] Part of data come from mocap.cs.cmu.edu, which was created with funding from NSF EIA-0196217

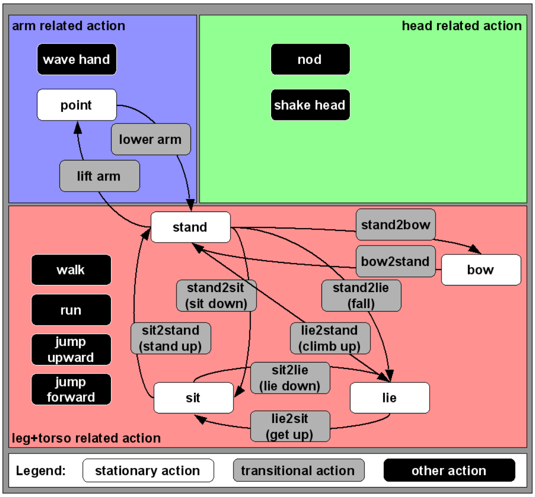


Fig. 1. Categorization of actions that need to be recognized

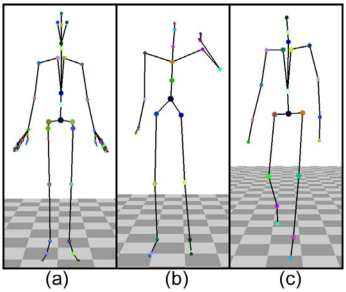


Fig. 2. Three different joint configurations. Notice the difference in the number of joints and joint hierarchy

Actions can also be classified based on the overall motion of human body. We view a *stationary* pose, *e.g. stand or sit* (*white blocks* in Fig.1) as a special type of action, with the constraint on the minimum duration. The *transitional* actions (*gray blocks*) transit from one stationary pose to another. The remainder (*black blocks*) consist of *periodic* actions (*e.g. walk, run*) and other actions.

MoCap data provides three rotation angles of each joint at each frame, however, we can not use them directly primarily because the data are heterogeneous in terms of different joint configurations (i.e. number of joints/bones, joint names and joint hierarchy). One example of this difference is shown in Fig.2. Using rotation representation, it is difficult to find correspondence between two sequences with different joint configurations. Instead, we compute 3D joint positions from rotation by kinematics. To find correspondence between two different joint configurations, we simply specify the corresponding joints the same name and ignore unwanted joints. We unify all joint configurations to the one shown in Fig.2(c) that consists of 23 joints. The joint positions are normalized so that the motion is invariant to the absolute body position, the initial body orientation and the body size.

3 Action Representation Using Spatio-temporal Motion Templates

As there are 23 joints and each has 3 coordinates (only y coordinate is used for hip), the whole body pose at each frame can be represented by a vector (called a *pose vector*) in a 67D space (called *pose space*). Consequently, any instance

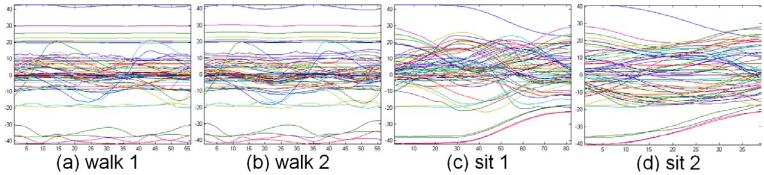


Fig. 3. Examples of action *walk* and *sit*. The horizontal axis is the frame number. The difference between the two *sit* sequences shown in (c) and (d) is significant. Note that the length of (c) and (d) is also different

of action can be viewed as a trajectory in the pose space. Fig.3 shows two instances of action *walk* and *sit*. Here for purpose of visualization, we overlay 67 1D trajectories of each single coordinate together. For example, the pose shown in Fig.2(c) corresponds to all points at the first frame of Fig.3(a).

An action recognition algorithm has to take dynamic information into consideration. Classification techniques based on static poses, such as Bayesian Network or *Support Vector Machine* (*SVM*), clearly won't work because different actions can share the same static poses. We tried a Bayesian network to classify static pose and found the classification accuracy to be less than 50%. Johansson's seminal study on *Moving Light Displays* (*MLDs*) [6] also confirms this result.

Hidden Markov Models (*HMM*) [10] is a powerful tool to capture dynamics of a time series. However, training a continuous *HMM* in such a high dimensional space requires large amount of training samples, which is not always available, especially for those infrequent actions. Other widely-used techniques include *Dynamic Time Warping* and template matching. Template matching is appealing to us because of its simplicity and less demanding requirement for training samples. Action recognition can be intuitively formulated as a template matching problem: If we are given one motion template (trajectory of pose vectors) of each action type, we can recognize an unknown sequence by comparing the trajectory of the unknown sequence with each template and find the best match using some distance measurement, e.g. *Sum of Squared Differences* (*SSD*).

However, this idea over-simplifies the problem because it does not consider noise and the spatial variation among the different instances of the same action type. For example, the idea may be able to recognize the *walk* sequence shown in Fig.3(b), given Fig.3(a) as the template, because these two sequences appear to be very similar. But there is apparent difference between the two *sit* sequences shown in Fig.3(c) and (d) and thus template matching based on simple distance measurement may misclassify this example.

Another problem is that trajectories have different discriminative power. For instance, the sine wave-like trajectories in the *walk* examples are salient and suitable for distinguishing *walk* from *sit*. The trajectories shown at the top and bottom of the *sit* examples look also unique. However, the difference between most of corresponding trajectories (those in the middle of each figure) in *walk* and *sit* can not be easily seen. Thus those trajectories are not good features for distinguishing *walk* and *sit*. Using all trajectories evenly for template matching is undesirable because the discriminative power of those good features is compromised when the distance between two whole pose vectors is computed.

We address these two problems in next two sub-sections and we describe the recognition algorithm in section 4.

3.1 Learning Motion Channels

We define a motion channel as the information that encodes the evolution of a single joint coordinate for a specific action class. If we are given several training samples, the simplest form of a motion channel is the mean vector of the samples and *SSD* can be applied as the matching function. But as stated above, this can not accommodate noise and the spatial variation among real sequences. So we also include the variance information in the motion channel and use the following χ^2 function to measure the closeness, or fitting error in other words, of an unknown trajectory X to the motion channel C. μ and σ^2 are mean and variance vector of C. f is the frame index.

$$err(X,C) = \chi^2(X,C) = \frac{\sum_{f=1}^{L} \left(\frac{X_f - \mu_f}{\sigma_f}\right)^2}{L} \tag{1}$$

Unlike *SSD*, the χ^2 function considers the prior distribution of the training samples and consequently allows large deviation from the mean when the variance is also large. Note that Eq.1 is normalized by L, the length of the channel, to eliminate the discrepancy caused by different channels lengths. Fig.4(a) shows some training samples of one channel and (b) shows the corresponding mean vector μ and error bar $(\mu_f - \sigma_f, \mu_f + \sigma_f)$ at each frame.

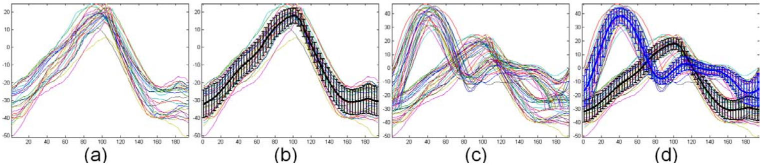


Fig. 4. (a) and (b) show some training samples of a channel with single cluster and the corresponding mean vector μ and error bar $(\mu_f - \sigma_f, \mu_f + \sigma_f)$ at each frame. (c) and (d) show some examples of a channel with multiple clusters and the clustering result using *k-means* algorithm

Eq.1 works fine when there is only one cluster. However, as shown in Fig.4(c), a channel may have multiple clusters (The training samples shown here contain two types of *sit*: *sit* on a high stool and *sit* on a step stool). So we use *k-means* algorithm to cluster the training data first and then find the closest match, as shown in Eq.2. μ_c and ${\sigma_c}^2$ are mean and variance of the c-th cluster of C.

$$err(X,C) = \min_{c} \frac{\sum_{f=1}^{L} \left(\frac{X_f - \mu_{c,f}}{\sigma_{c,f}}\right)^2}{L} \tag{2}$$

The number of clusters is set manually. Note that *k-means* algorithm takes vectors with length L as the input so the clustering result is not affected when two cluster centers are crossing at some frames, as shown in Fig.4(c) and (d).

We also tried other *soft* clustering algorithm such as Gaussian Mixture Model. However, in many cases, because we don't have enough training data (i.e. the number of training samples is less than 67), the covariance matrix of each Gaussian is singular and thus each training sample determinately belongs to one of clusters. Therefore the result is almost the same as *k-means*.

Temporal variation also exists, i.e. training samples contain different number of frames. In such cases, we first resample these trajectories using *Spline* interpolation so that they all have L frames. For *periodic* actions, we manually align training samples such that starting and ending phases are approximately the same.

3.2 Learning Motion Templates

We define a motion template as a set of N motion channels $\{C_j\}$ with weights $\{w_j\}$, *j=1,2,...,N*, which as a whole can capture the uniqueness of the motion of one specific action type. The error of fitting an unknown sequence Y (with length L) to template T is the weighted sum of fitting error of component Y_j to the corresponding channel C_j, as shown in Eq.3.

$$err(Y,T) = \sum_{j=1}^{N} w_j \cdot err(Y_j, C_j) \tag{3}$$

Weight w_j should reflect the discriminative power of C_j. We learn w_j according to the $Neyman$-$Pearson$ criterion [14]: We impose constraint on classification rate and try to minimize false alarm rate. Suppose we have P training samples of T and Q training samples of all other templates, which are considered as P positive and Q negative training samples of T. Suppose $(e^+_{j,1}, e^+_{j,2}, ..., e^+_{j,P}, e^-_{j,1}, e^-_{j,2}$, ... , $e^-_{j,Q})$ are errors of fitting P positive and Q negative samples to channel C_j. We sort these P+Q values in ascending order and the sorted list is $(e_{j,1}, e_{j,2}, ..., e_{j,P+Q})$. If the classification rate is γ, we have to accept γP positive samples. Suppose the shortest sub-list that contains γP positive samples is $(e_{j,1}, e_{j,2}, ..., e_{j,R_j})$. It also contains $R_j - \gamma P$ negative samples. $\frac{R_j-\gamma P}{Q}$ is thus the minimum possible *false alarm* rate. Let w_j be the *true rejection* rate and it is normalized as follows:

$$w_j = \left(1-\frac{R_j-\gamma P}{Q}\right)/\sum_{n=1}^{N}\left(1-\frac{R_n-\gamma P}{Q}\right) \tag{4}$$

Here e_{j,R_j} defines an error threshold for channel C_j in that samples with a larger error are rejected. We also define an error threshold for template T: $\varepsilon = max\{e_{j,R_j}\}$, j=$1, 2, ..., N$, which will be used in the recognition algorithm.

It is important to note that although this algorithm does not consider the joint distribution of channels, the correlation between channels is actually preserved in that all channels are in nature synchronized by time. However, a counterexample would be: Suppose we have two channels A and B, each contains two clusters $(A1, A2)$ and $(B1, B2)$. If the only possible combination of A and B is $(A1, B1)$ or $(A2, B2)$, an unknown sequence with combination $(A1, B2)$ or $(A2, B1)$ will generate a false alarm by the algorithm. Fortunately, in practice, such counterexample rarely exists (It never occurs in our dataset actually).

For *stationary* actions such as *stand*, the process is somewhat different in that there is no dynamic information. Therefore, the processing unit is a scalar value instead of a vector. This is actually a special case of the above algorithm: We can still use above equations except that L=1. Also when given training sequences, each frame is considered as one occurrence of the action. So P, the number of training samples, equals the total number of frames.

4 The Recognition Algorithm

Suppose we have trained a set of motion templates $\{T_i\}$ (with length L_i), the recognition becomes straightforward: Given an input sequence Y, at each frame t, we compare the fitting errors of Y within the sliding window with each of templates and the one with the minimum fitting error is considered to be the ongoing action, as shown in Eq.5.

$$action(Y) = \underset{i:err(Y,T_i)\leq\varepsilon_i}{\arg\min} (err(Y, T_i)) \tag{5}$$

Because there are three action groups *leg+torso, arm, head* and we only compare templates in the same group, there may be up to three actions recognized at one time. But if $err(Y, T_i) > \varepsilon_i$ for all T_i in one group, none of the actions in that group is chosen.

The correct recognition can not be done until the whole course of the action has been discovered. This means the previous L_i-1 frames may have already been recognized as something else. We keep track of the minimum error and the best action so far of each frame. At frame t, suppose T_i is the best match using the predictor in Eq.5 and the fitting error is $err_{T_{i\,t}}$. The program looks back and compares $err_{T_{i\,t}}$ with error stored in each frame of $[t - L_i + 1, t]$. If in one frame $err_{T_{i\,t}}$ is smaller, the minimum error in that frame is updated to $err_{T_{i\,t}}$ and the best action is changed to T_i. This method can correct wrong recognition of the previous frames but there is still L_i frames of delay.

As T_i has a fixed length L_i, if Y and T_i have different time scale (*e.g.* someone walks faster than others), the fitting error will be large even if Y and T_i represent the same action. So we resample T_i to form a series of templates $\{T_{i,L_{i_l}}\}_{l=1,2,\ldots,d_i}$. In our experiment, d_i=10.

For *stationary* actions, to make recognition results more stable, we impose a constraint such that the duration should be longer than some threshold L_δ. To clarify, we compute errors of fitting each frame in $[t - L_\delta + 1, t]$ to T_i using Eq.3. The overall error at current frame t is the mean of these L_δ fitting errors. We chose L_δ=60, which is equivalent to 2 seconds video in NTSC format.

5 Experimental Results

5.1 Results on MoCap Data

The 848 MoCap sequences in our dataset contain 159,564 frames in total. We manually segmented these sequences such that each segment contains a whole course of one action. In total we have 2343 action segments. The distribution of these segments in each action class is not uniform. *Walk* has 204 segments while *lie2stand* has only 19 segments. The average number is 107. The length of these segments are also different, ranging from 47 to 91 frames. The average is 68 frames.

In **Experiment 1**, we randomly selected half of segments of each action class for training and the remainder for classification. In **Experiment 2**, we reduced the amount of training data to 1/3. We repeated these experiments five times and the average classification rate of each class is shown in Fig.5(a). Here we chose γ=95% as the classification rate in training. The overall classification rate of *Exp.1* and 2 is 90.7% and 86.9%, respectively. As expected, the performance of the latter decreased, but not too much, which indicates the robustness in terms of the amount of available training data. Compared with *Exp.1*, most of the first 3 best channels (not shown due to limited space) for each action did not change (although the order may be different). This shows consistency of our algorithm in selecting good features.

Fig.5(b) shows the confusion matrix of the action group *leg+torso* in *Exp.1*. As indicated in those black grids on the diagonal, most of actions have been

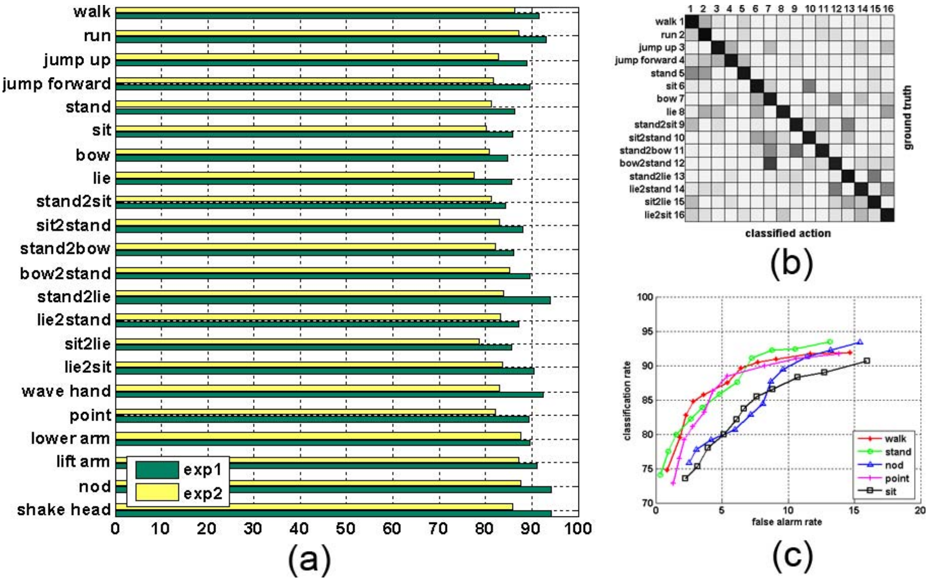


Fig. 5. (a)Classification result of *Exp.1* and 2; (b)Confusion matrix of the action group *leg+torso* in *Exp.1*. A dark grid indicates strong confusion. (c)ROC curves shows the influence of γ on the classification performance

correctly classified. Strong confusion (dark grids) only occurs between similar actions such as *bow* and *bow2stand*.

The classification rate γ plays an important role in our algorithm in that it affects not only the channel weights in training but also the error threshold ε_i in recognition. In **Experiment 3**, we used the same setup as in *Exp.1* except we chose different γ to see its influence. Although only in training phase can γ have a direct influence, it can end up affecting the classification rate and false alarm rate in testing. The ROC curves are shown in Fig.5(c). (Only some action types are shown here.) As expected, there is a tradeoff between a high classification rate and a low false alarm rate.

In **Experiment 4**, we tested our recognition algorithm on 73 unsegmented long sequences (from testing set) with average length of 914 frames. We use the templates learned in *Exp.1*. The algorithm achieves a recognition rate of 90.7% (in terms of frames).

Our algorithm is fast. Training about 75,000 frames takes about 28 minutes (computation time only, not counting the time of loading files) on a PC with a P4 2.4GHz CPU. Recognizing about 75,000 frames takes only 9.3 minutes.

5.2 Results on Annotated and Tracked Data

We tested our algorithm on two annotated (994 frames in total) and one automatically tracked (159 frames) sequence. Fig.6(a) shows some key frames of one annotated sequence. The rendered annotation results using POSER are displayed on the right. The text in top-right corner is the frame number with ground truth label and the text in bottom-right corner is the recognition result. The complete ground truth and the recognition result are shown in Fig.6(b).

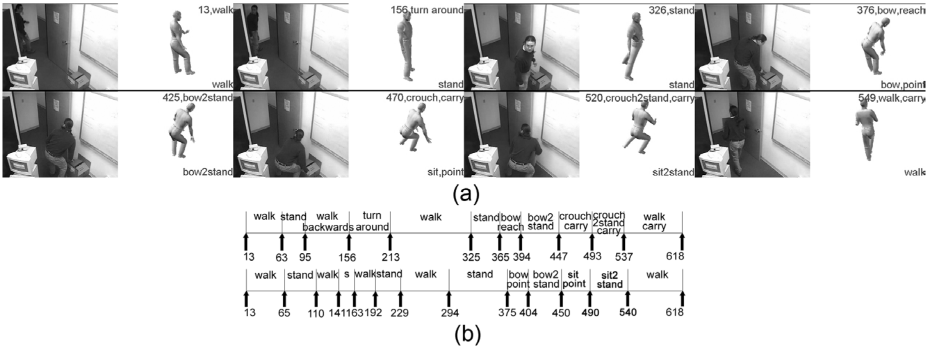


Fig. 6. (a)Key frames of one annotated video; (b)The ground truth (top) and recognition result (bottom)

Fig.6(b) shows that most of actions have been correctly recognized although the segmentation is not perfect. Errors occur when the subject turns around because we don't model such actions in our action set. "Carry" was not recognized for the same reason. “Reach” and “crouch”, however, were recognized as “point” and “sit”, which are reasonable substitutions for “reach” and “crouch”.

The recognition rate on the annotated and tracked data is 82.3% and 80.6%, respectively. This is satisfactory considering a substantial amount of jittery noises contained in the data (root mean square position error rates of about 10 pixels (∼10 cms) and joint angle errors of about 20°). The proposed algorithm in general is robust to this type of errors with short duration because the algorithm eliminates their effects on the overall fitting error by averaging all frames within the sliding window.

6 Conclusions

We have presented a learning-base algorithm to represent and recognize 3D human actions using spatio-temporal motion templates. Each template consists of a set of motion channels where each channel corresponds to the evolution of one 3D joint coordinate. Channels with strong discriminative power are highlighted according to *Neyman-Pearson* criterion. The recognition is based on the minimum weighted sum of χ^2 distance to each motion channel. This simple representation and recognition algorithm performs satisfactorily well in terms of fast speed and high recognition rate on a large set of human actions.

We separate action recognition from the motion recovery problem (though, the two may not be entirely independent) by using 3D data. This imposes one limitation on the proposed approach. However, good results on the noisy annotated and automatically tracked videos show the potential of using real data.

Our future work includes adding more arm-related actions and extending the algorithm to recognize actions at the semantical level.

References

1. A. Agarwal and B. Triggs. 3D Human Pose from Silhouettes by Relevance Vector Regression. In *Proc. of CVPR*, pp. 882-888, 2004.
2. L. Campbell and A. Bobick. Recognition of human body motion using phase space constraints. In *Proc. of ICCV*, pp. 624-630, 1995.
3. J. Davis and A. Bobick. The Representation and Recognition of Action Using Temporal Templates. In *Proc. of CVPR*, pp. 928-934, 1997.
4. K. Derpanis, R. Wildes and J. Tsotsos. Hand Gesture Recognition within a Linguistics-Based Framework. In *Proc. of ECCV*, pp. 282-296, 2004.
5. S. Gong, M. Walter and A. Psarrou. Recognition of temporal structures: Learning prior and propagating observation augmented densities via hidden Markov states. In *Proc. of ICCV*, pp. 157-162, 1999.
6. G. Johansson. Visual perception of biological motion and a model for its analysis. *Perception and Psychophysics* 14 (2) (1973) 201-211.
7. M.W. Lee and R. Nevatia. Dynamic Human Pose Estimation using Markov chain Monte Carlo Approach. In *Proc. of the IEEE Workshop on Motion and Video Computing* (WACV/MOTION’05), 2005.

8. A. Oikonomopoulos, I. Patras and M. Pantic. Spatiotemporal saliency for human action recognition. In *Proc. of IEEE Int'l Conf. on Multimedia and Expo* (ICME '05), 2005.
9. V. Parameswaran and R. Chellappa. View invariants for human action recognition. In *Proc. of CVPR*, pp. 613-619, 2003.
10. L. R. Rabiner. A tutorial on Hidden Markov Models and selected applications in speech recognition. In *Proc. of the IEEE*, 77(2):257-286, 1989.
11. C. Rao, A. Yilmaz and M. Shah. View-Invariant Representation and Recognition of Actions. In *Int'l Journal of Computer Vision* 50(2), Nov. 2002, pp. 203-226.
12. E. Shechtman and M. Irani. Space-Time Behavior Based Correlation. In *Proc. of CVPR*, pp. 405-412, 2005.
13. A. Shokoufandeh, S.J. Dickinson, C. Jonsson, L. Bretzner and T. Lindeberg. On the representation and matching of qualitative shape at multiple scales. In *Proc. of ECCV*, pp. 759-775, 2002.
14. V. Trees and H. L. Detection, Estimation and Modulation Theory, Part I. *John Wiley and Sons*, New York, 1968. ISBN 0-47109517-6.
15. Z. Zhang, Y. Wu, Y. Shan and S. Shafer. Visual panel: Virtual mouse keyboard and 3d controller with an ordinary piece of paper. In *Workshop on Perceptive User Interfaces.* ACM Digital Library, November 2001. ISBN 1-58113448-7.

Interactive Point-and-Click Segmentation for Object Removal in Digital Images

Frank Nielsen[1] and Richard Nock[2]

[1] Sony Computer Science Laboratories, Inc.
Frank.Nielsen@acm.org
[2] Université Antilles-Guyane
rnock@martinique.univ-ag.fr

Abstract. In this paper, we explore the problem of deleting objects in still pictures. We present an interactive system based on a novel intuitive user-friendly interface for removing undesirable objects in digital pictures. To erase an object in an image, a user indicates which object is to be removed by simply pinpointing it with the mouse cursor. As the mouse cursor rolls over the image, the current implicit selected object's border is highlighted, providing a visual feedback. In case the computer-segmented area does not match the users' perception of the object, users can further provide a few inside/outside object cues by clicking on a small number of object or nonobject pixels. Experimentally, a small number of such cues is generally enough to reach a correct matching, even for complex textured images. Afterwards, the user removes the object by clicking the left mouse button, and a hole-filling technique is initiated to generate a seamless background portion. Our image manipulation system consists of two components: (i) fully automatic or partially user-steered image segmentation based on an improved fast statistical region-growing segmentation, and (ii) texture synthesis or image inpainting of irregular shaped hole regions. Experiments on a variety of photographs display the ability of the system to handle complex scenes with highly textured objects.

1 Introduction

Removing and cutting objects from digital pictures are main operations of desktop publishing (DTP) for which many dedicated tools have been designed and refined over the years (e.g., the *magnetic lasso* or the *magic wand* of Adobe® Photoshop®). Image cutouts are typically pasted (composited) on a different background for photomontages. Removing objects is important in the movie industry to obtain clean plates (say, remove camera tripods or other calibration materials that have been used on stage to facilitate postprocessing computer graphics effects). Clean plates are usually obtained by synthesizing the *background mosaic* by tracking and registering a sequence of frames and removing undesirable objects by manually painting them for each frame, a time consuming process. With the advent of image inpainting [2] and texture synthesis technologies [19] for generating seamlessly image portions, removing objects in still

N. Sebe, M.S. Lew, and T.S. Huang (Eds.): HCI/ICCV 2005, LNCS 3766, pp. 131–140, 2005.

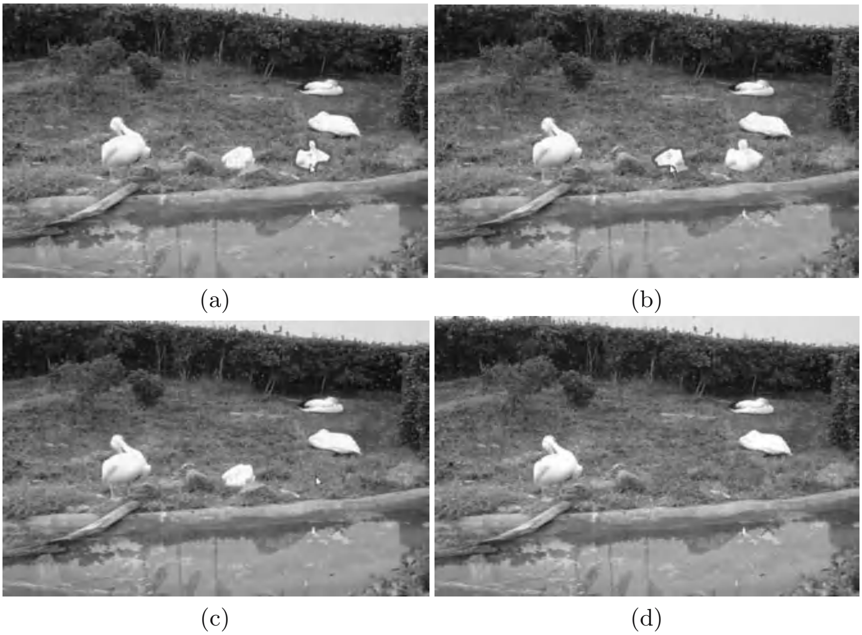

(a) (b) (c) (d)

Fig. 1. ClickRemoval is an interactive image object removal system. As the user's mouse cursor rolls over the image, the current implicitly selected object is highlighted, (a) and (b). By simple mouse clickings objects are removed and background portions are instantaneously synthesized, (c) and (d)

pictures becomes today an essential primitive of image retouching software. To remove objects in photographs (see Figure 1), we do not need a pixel accuracy that also requires to pull out the alpha matte but rather requires a *coarse* bounding region that separates the selected object from the remaining background of the scene. Because object cognition by computers is still very far from that of human abilities, the challenge consists in designing an intuitive *user interface* (UI) and corresponding effective *user-steered segmentation* algorithm for quickly selecting objects by putting the user's high-level cognition in the loop.

1.1 Related Work

Prior object selection work in images are classified into two categories:

Contour-based selection. A user marks the object boundaries by coarsely and piecewisely sketching with the mouse cursor the contours that are *on-the-fly* finely optimized to fit the object boundaries. The first contour-based methods were developed independently in 1995 using either dynamic programming [7] (*intelligent scissors*) or gradient descent optimization [4] (*image snapping*). Those methods are also better known today as *magnetic lassos*. Intelligent scissor techniques were later refined in [8]. Recently, a Monte-Carlo probabilistic system, called *jetstream* [11], has been designed to extract contours using particles. Contour-based selection works particularly well for in-focus objects out-of-focus background images, where the objects consist of a *single outer contour*. For more complex topological objects such as grid-like objects containing many holes, those methods are time consuming as they require users to trace *each* inner contour (see Figure 6).

Region-based selection. A user gives a few hints on which portion of the image is the object and which image pixels are the background, and an optimization algorithm then extracts the object based on these cues. The *magic wand* tool of Photoshop is such a typical system. Other approches are either based on segmentation and triangulation [1, 3, 16] or graph cuts [5, 14].

Note that even for image cutouts, a coarse extraction is also enough as it allows to initialize a trimap (labeling of images into background/object/undefined areas) to precisely extract objects with alpha mattes [6, 15] as a postprocessing operation. Matte extraction is required not only for copy-pasting translucent or furry objects that have potentially soft pixel memberships (pixels being a mixture of both object and background colors) but also necessary to merge seamlessly the object boundaries with the new background image.

1.2 System Overview

Our system, called ClickRemoval, consists of two basic modules: (1) user-steered segmentation, and (2) hole-filling. In the ideal scenario, the user just has to pinpoint objects s/he wants to remove and press the left mouse button to remove and synthesize *instantaneously* the background part (see Figure 1). Because automatic segmentation may not yield expected results, we provide a simple mechanism to input bias by pointing and clicking just a few object/background pixels (see Figure 2). The next section describes the ClickRemoval user interface. Section 3 present the novel statistical region-growing segmentation algorithm, and Section 4 concludes the paper.

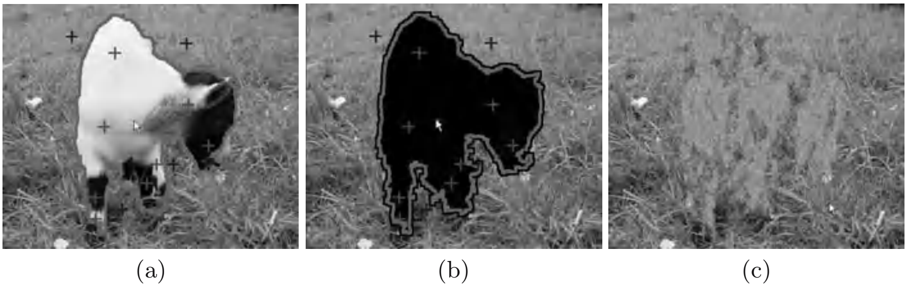


Fig. 2. Matching computer/human object definitions by putting the user in the loop and iteratively refining manually the segmentation. (a) depicts the segmentation obtained after the user manually pinpointed object (red) and background (blue) cues. (b) is the hole created by removing the slightly dilated object, and (c) is the result image after texture synthesis

2 User Interface Design

The user starts by loading a picture and moves the mouse cursor over the image (Figure 3.(1-3)a). Whenever the mouse moves, the segmented area implicitly defined by the cursor position is highlighted in real-time (Figure 3.(1-3)b). Computing segmentations is done in linear-time[1], as described in Section 3. Once an object is removed by the user, a fast hole-filling texture synthesis procedure is triggered [19] (Figure 3.1c, 3.2d and 3.3c).

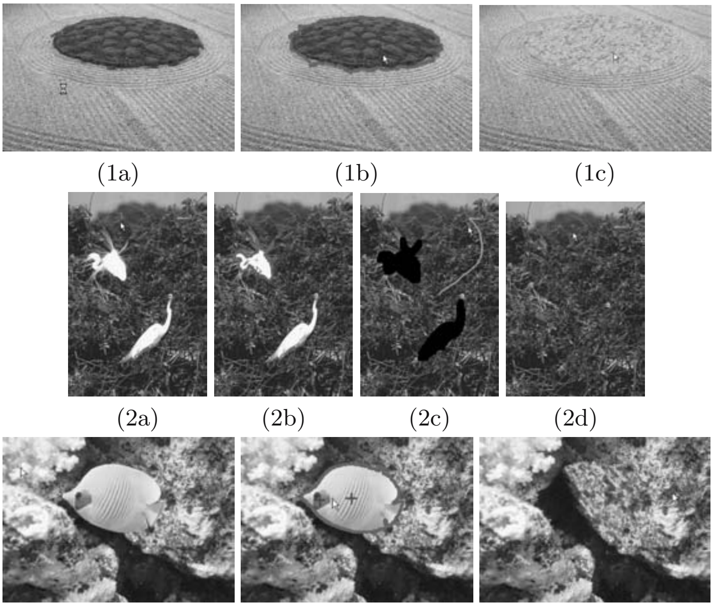
ClickRemoval has three operation modes:

1. To remove the implicitly selected area Fig 3.(1b), the user clicks the mouse left button. The system computes a bounding box around the removed area (expands it by some constant factor to be sure the object is contained in the selected area) and initialize texture synthesis using the remaining nonobject pixels inside the box (Figure 3.1.(1c)). We implemented the per-pixel texture synthesis of Wei and Levoy [19] because of its flexibility and real-time performance on current commodity PCs (see also [2]).
2. Sometimes, we prefer to *design* how to fill the hole using another part of the image. We then scribble the image by pushing the left mouse button and moving the mouse cursor over the portion of the image we are interested to initialize the texture synthesis (Figure 3.(2c)). (Depending on whether a hole has been created or not in the image, the UI interprets the left button click differently.) We define the selected pixels by taking either the bounding box of the stroke or choosing the image pixels falling within some prescribed distance from the stroke.
3. Automatic segmentation may fail to deliver appropriate object decompositions. In case of failure, the user presses the SHIFT key and the mouse left (object) or right (background) mouse button to indicate prior cues (Figure 2 and Figure 3.(3b)). ClickRemoval then instantly refines the segmentation to satisfy the constraints of the user's cues (segmentation priors). Texture synthesis is either initialized automatically using the remaining pixels of an enlarged bounding box, or user-steered (as described in 2.).

3 Statistical Image Segmentation

Image segmentation is computed using a fast iterative statistical region-growing process. The region-growing segmentation framework dates back to the late 60s. Since then, it has been a very popular method of image processing/computer vision [13]. A region growing segmentation starts by initializing for each pixel a corresponding single-pixel region (say, pixel region $\mathbf{R}_l = \{(x, y)\}$ has initially region ID number $l = x + yw$, where w denotes the image width). At each iteration of the region growing algorithm, we consider the region adjacency graph (RAG), and based on a *merging predicate*, decide to merge or not the pair of adjacent regions that is under consideration. Usually, the RAG is dynamically updated.

[1] More precisely, almost linear-time in theory



(1a) (1b) (1c)

(2a) (2b) (2c) (2d)

(3a) (3b) (3c)

Fig. 3. The three different modes of the ClickRemoval interface: fully automatic, manual texture selection, and both user-steered segmentation/texture selection

The segmentation result strongly depends on: (1) the merging predicate, and (2) the order in which pairs of adjacent regions are inspected. Let $\mathbf{R}_i$ and $\mathbf{R}_j$ denote a pair of adjacent regions. A typical merging predicate $P(\mathbf{R}_i, \mathbf{R}_j)$ is to test whether $|\overline{\mathbf{R}_i} - \overline{\mathbf{R}_j}| \leq \max\{\sigma_i, \sigma_j\}k$, where $\overline{\mathbf{R}_i}$ and σ_i (resp. $\overline{\mathbf{R}_j}$ and σ_j) are the color mean and variance of region $\mathbf{R}_i$ (resp. $\mathbf{R}_j$), and k is a prescribed constant. Let $|\mathbf{R}|$ denote the number of pixels defining region $\mathbf{R}$. Let $\mathbf{R}(p)$ denote the region containing pixel p (bag of pixels). In our recent segmentation work [10], we developed a fast linear-time algorithm with provably guaranteed segmentation bound based on a statistical image generation model. We presented: (1) a concentration inequality based on statistical aggregation phenomena[2] and showed that in practice it is enough to consider (2) a *static order* of region pairs (that is, do not update the RAG, a time consuming procedure). Because regions are *disjoint sets* of pixels ($\mathbf{R}_i \cap \mathbf{R}_j = \emptyset$ for $i /\!\!= j$), we can merge regions and retrieve their IDs ($\mathbf{R}(p)$ provided a pixel handle p) using the optimal union-find datastructure of Tarjan [9, 17]. The algorithm is further shown robust to noise and handle occlusions [10] (that is, the method can segment as a *single object* several connected areas of the image that belong to the same object). We summarize the static order region-growing segmentation algorithm:

[2] For example, the sum of independent uniform random variables yields a Gaussian random variable (central limit theorem). More generally, statistical aggregation phenomena have been recently found for random variables satisfying loose distribution assumptions [10]

```
RegionMerging($\mathbf{I}$)
◁ 4-connectivity of pixels $C_4$ (implicit RAG) ▷
$\mathbf{P} \leftarrow \{(p_l, q_l, C_l = |\mathbf{I}(q_l) - \mathbf{I}(p_l)|) \mid \text{with } q_l \in C_4(p_l)\}$
◁ Sort $\mathbf{P}$ in increasing order of their $C_l$ keys ▷
Sort($\mathbf{P}$)
for $i \leftarrow 1$ to $|\mathbf{P}|$
    do
        if FindRegionID($p_l$) $\neq$ FindRegionID($q_l$)
          then if MergePredicate($\mathbf{R}(p_l), \mathbf{R}(q_l)$)
                  then MergeRegions($\mathbf{R}(p_l), (q_l)$)
```

The union-find data-structure [17] is implemented as follows:

```
InitializeRegionID($x$)
parent($x$) $\leftarrow x$
rank($x$) $\leftarrow 0$
```

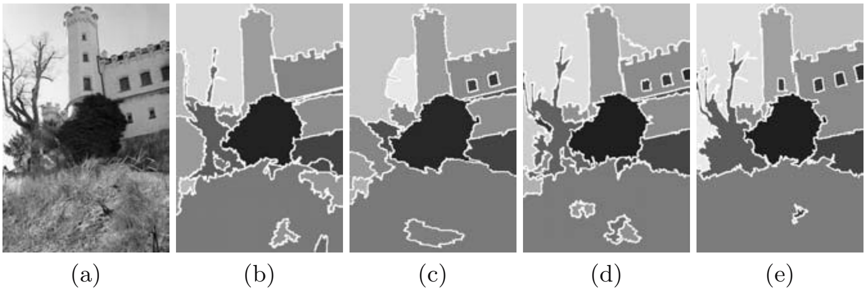
```
FindRegionID($x$)
◁ Walk from $x$ to the leader pixel element ▷
while $x \neq$ parent($x$)
do $x \leftarrow$ parent($x$)
return $x$
```

```
MergeRegions($x, y$)
◁ Union by rank and path compression ▷
$x_r \leftarrow$ FindRegionID($x$)
$y_r \leftarrow$ FindRegionID($y$)
if rank($x_r$) > rank($y_r$)
   then parent($x_r$) $\leftarrow y_r$
   else parent($y_r$) $\leftarrow x_r$
           if rank($x_r$) = rank($y_r$)
              then rank($x_r$) $\leftarrow$ rank($x_r$) + 1
```

Figure 4 presents the segmentation results obtained by using different statistical predicates. ClickRemoval uses the following merging predicate based on concentration inequalities [10]:

$$P(\mathbf{R}, \mathbf{R}') = \begin{cases} \texttt{true} & \text{iff } |\overline{\mathbf{R}'} - \overline{\mathbf{R}}| \leq \Delta(\mathbf{R}) + \Delta(\mathbf{R}') \\ \texttt{false} & \text{otherwise} \end{cases}, \tag{1}$$

with $\Delta(\mathbf{R}) = \frac{256^2}{64|\mathbf{R}|}(\min(256, |\mathbf{R}|)\log(1 + |\mathbf{R}|) + 2\log 6wh)$, for a 8-bit gray image $\mathbf{I}$ of dimension $w \times h$ pixels. For color images, we consider the RGB channels independently of each other and define the merging predicate as the Boolean And of elementary predicates: $P_{RGB}(\mathbf{R}, \mathbf{R}') = P_R(\mathbf{R}, \mathbf{R}') \wedge \ P_G(\mathbf{R}, \mathbf{R}') \wedge P_B(\mathbf{R}, \mathbf{R}')$. Region-growing segmentation tends to yield imprecise boundaries compared to graph-cut edge-based segmentation methods [5]. Loosely speaking, as regions become bigger the mean/variance statistics reflect statistically better the region attributes but does not tell much on whether merging two incident regions will provide a seamless merge at the region borders. Thus, to obtain more precise object boundaries from the region-merging paradigm, we rather consider the statistics of the *region crusts* (and not the full regions as in [10]). Figure 5 illustrates this region/crust concept. The crust of region $\mathbf{R}$ is defined as the set of pixels *within* distance c to the region's border $\partial\mathbf{R}$. (Thus for large enough c, there



(b) $P_1 : |\overline{\mathbf{R}_i} - \overline{\mathbf{R}_j}| \leq t$

(c) $P_2 : |\sigma(\mathbf{R}_i) - \sigma(\mathbf{R}_j)| \leq s \max(\sigma(\mathbf{R}_i), \sigma(\mathbf{R}_j))$

(d) $P_3 : |\overline{\mathbf{R}_i} - \overline{\mathbf{R}_j}| \leq \sqrt{b^2(R_i) + b^2(R_j)}$, with $b(\mathbf{R}) = 512\sqrt{\frac{1}{\min(g,|\mathbf{R}|)} \log(2 + \frac{|\mathbf{R}|}{wh})}$

(e) $P :$ $|\overline{\mathbf{R}_i} - \overline{\mathbf{R}_j}| \leq \Delta(\mathbf{R}_i) + \Delta(\mathbf{R}_j)$, with

$$\Delta(\mathbf{R}) = \frac{256^2}{64|\mathbf{R}|}(\min(256, |\mathbf{R}|)\log(1 + |\mathbf{R}|) + 2\log 6wh)$$

Fig. 4. Region growing segmentation: (a) original image, (b) predicate P_1 (mean threshold with $t = 30$), (c) predicate P_2 (standard deviation threshold with $s = 1.5$), (d) predicate P_3 (simple concentration inequality) and (e) ClickRemoval's concentration inequality predicate P

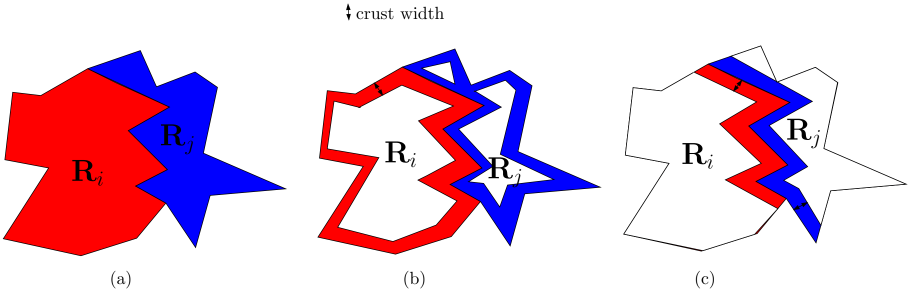


Fig. 5. Regions (a), circular region crusts (b), and crusts (c). The white areas are not taken into account for computing region statistics

is no difference between regions and their crusts, and the algorithm is identical to [10].) Furthermore, we only need to update the crust's mean/variance statistics when we merge regions . Updating the crust statistics is done by retrieving the pixels of the merged region belonging to its crust using a slightly modified flood-filling algorithm[3].

Since fully automatic segmentation may give results that differ from human perception, we provide a simple user-steered mechanism to control interactively the segmentation. A user may input object (foreground)/background cues by pointing and clicking at a few pixel positions. Each time a user input some bias, the segmentation is recomputed in real-time (0.03s for VGA images on Intel®

[3] Flood-filling is used in painting systems to fill objects from a seed pixel position given a specified foreground color (see [9], Chapter 2). Flood-filling-type segmentations are also called watershedding (eg., Photoshop's magic wand)

Pentium® IV 3.6 GHz), as we do not need to reinitialize or sort the region pair order. We handle bias in our crust merging algorithm similarly to [10]. First, let us rename regions into metaregions, and define *pure metaregions* as the metaregions that do not contain any bias information. The other metaregions are said *biased* and contain at least one object/background pixel pinpointed by the user. When inspecting the adjacent metaregion pairs, we *never* merge metaregion pairs that contain both bias. Pairs with both pure metaregions are handled as in the nonbias case. To decide whether to merge a biased metaregion with a pure metaregion, we choose among *all* metaregions having the same bias ID (say, 1=object and 0=background), the one that statistically best matches the pure metaregion and test using the merging predicate P whether they should merge or not. At the last stage, we merge all biased object metaregions together, and all biased background metaregions altogether. Note that ClickRemoval is different from watershedding-like Photoshop's magic wand since (1) it performs global segmentation, and (2) accept both inside/outside object cues.

4 Experiments and Conclusions

We implemented the ClickRemoval system in C++ using OpenGL®. The full code is a mere 1000 lines, including both the novel crust merging segmentation algorithm and the per-pixel texture synthesis procedure [19]. Figure 4 and Figure 5 displays a few examples obtained using ClickRemoval. The accompanying video provides a sense of the UI and the system responsiveness. Although our crust-based region-growing segmentation algorithm does not provide as accurate object boundaries as edge-based graph cut methods [5], it is much faster, allows to handle *light* user-supplied bias information, and is anyway enough for our object removal application. Bias is input as a few inside/outside points (the smallest "extra" information unit) and not by strokes as in [5]. This UI is particularly advantageous for objects with many contours, since "intelligent scissors" assume in their UI that objects have only a single outer contour (Figure 6).

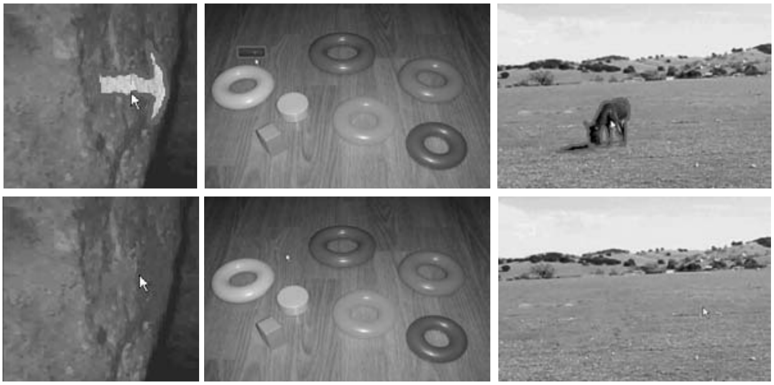


Fig. 6. ClickRemoval: 1-Click results (fully automatic segmentation)

We are currently investigating further extensions to our system: pulling out object mattes from trimaps obtained by our segmentation (e.g., see [6, 15]), trading between texture synthesis and image inpainting [2], testing various texture synthesis methods (per-pixel, per-patch, per-tile, etc) while keeping the system responsiveness.

We envision such an interactive system fed by live video camera images with numerous applications in computational photography [1] and augmented/modified reality.

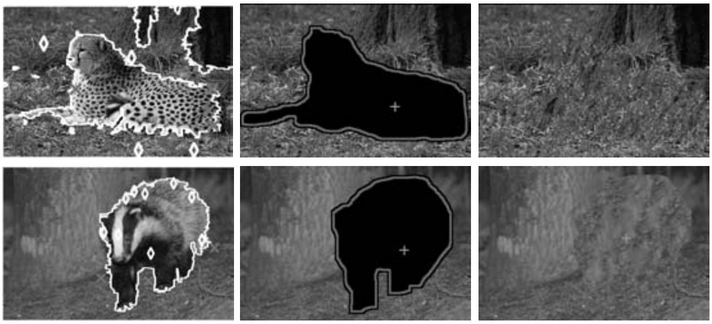


Fig. 7. Result with user-input bias: a few clicks are enough to remove complex objects

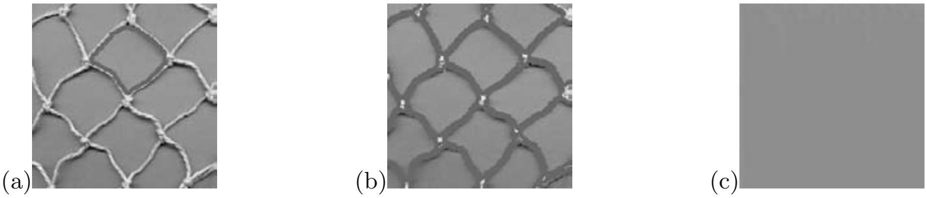


Fig. 8. ClickRemoval allows users to either (a) catch a hole, or (b) remove the whole net using a single click. (c) is the result image after texture synthesis

Acknowledgments

The Berkeley Segmentation Dataset and Benchmark (`http://www.cs.berkeley.edu/projects/vision/grouping/segbench/`). The Approximate Nearest Neighbor libary (`http://www.cs.umd.edu/~mount/ANN/`). The Visual C++ AFMM inpainting source code [18] (`http://www.acm.org/jgt/papers/Telea04/`). We thank Dr. Shigeru Owada for implementing a light version of the Poisson matting [15] and Poisson image editing [12].

Code availability. The user-steered statistical region growing segmentation code is available from the authors' Web pages (Windows®/Linux®).

References

1. W. A. Barrett and A. S. Cheney. Object-based image editing. In *ACM SIGGRAPH*, pp. 777–784, 2002.
2. A. Criminisi, P. Perez, and K. Toyama. Region filling and object removal by exemplar-based inpainting. *IEEE Transactions on Image Processing*, 13(9):1200–1212, 2004.
3. A. X. Falcão, R. A. Lotufo, and G. Araújo. The image foresting transformation. Technical Report #IC-00-12, 2000.
4. M. Gleicher. Image snapping. In *ACM SIGGRAPH*, pp. 183–190, 1995.
5. Y. Li, J. Sun, C.-K. Tang, and H.-Y. Shum. Lazy snapping. *ACM Trans. Graph.*, 23(3):303–308, 2004.
6. S. Lin, Q. Zhang, and J. Shi. Alpha estimation in perceptual color space. In *Proc. IEEE ICASSP*, 2005.
7. E. N. Mortensen and W. A. Barrett. Intelligent scissors for image composition. In *ACM SIGGRAPH*, pp. 191–198, 1995.
8. E. N. Mortensen and W. A. Barrett. Toboggan-based intelligent scissors with a four-parameter edge model. In *IEEE CVPR*, pp. 2452–2458, 1999.
9. F. Nielsen. Visual Computing: Geometry, Graphics, and Vision. Charles River Media, ISBN: 1-58450-427-7, 2005.
10. R. Nock and F. Nielsen. Semi-supervised statistical region refinement for color image segmentation. *Pattern Recognition*, 38(6):835–846, 2005.
11. P. Pérez, A. Blake, and M. Gangnet. Jetstream: Probabilistic contour extraction with particles. In *IEEE ICCV*, pp. 524–531, 2001.
12. P. Pérez, M. Gangnet, and A. Blake. Poisson image editing. *ACM Trans. Graph.*, 22(3):313–318, 2003.
13. A. Rosenfeld. *Picture processing by computer.* Academic Press, 1969.
14. C. Rother, V. Kolmogorov, and A. Blake. Grabcut: Interactive foreground extraction using iterated graph cuts. *ACM Trans. Graph.*, 23(3):309–314, 2004.
15. J. Sun, J. Jia, C.-K. Tang, and H.-Y. Shum. Poisson matting. *ACM Trans. Graph.*, 23(3):315–321, 2004.
16. K.-H. Tan and N. Ahuja. Selecting objects with freehand sketches. In *IEEE ICCV*, pp. 337–344, 2001.
17. R. E. Tarjan. Efficiency of a good but not linear set union algorithm. In *J. ACM*, 22:2, pp. 215-225, 1975.
18. A. Telea. An image inpainting technique based on the fast marching method. *Journal of Graphics Tools*, 9(1):23–34, 2004.
19. L.-Y. Wei and M. Levoy. Fast texture synthesis using tree-structured vector quantization. In *ACM SIGGRAPH*, pp. 479–488, 2000.

Information Layout and Interaction Techniques on an Augmented Round Table

Shintaro Kajiwara[1], Hideki Koike[1], Kentaro Fukuchi[1],
Kenji Oka[2], and Yoichi Sato[2]

[1] Graduate School of Information systems, University of Electro-Communications,
1-5-1, Chofugaoka, Chofu, Tokyo 182-8585, Japan
kaji@vogue.is.uec.ac.jp, koike@acm.org, fukuchi@megaui.net
http://www.vogue.is.uec.ac.jp/

[2] Institute of Industrial Science, The University of Tokyo, 4-6-1 Komaba, Meguro-ku,
Tokyo 153- 8505, Japan
{oka,ysato}@iis.u-tokyo.ac.jp
http://www.hci.iis.u-tokyo.ac.jp/

Abstract. Round tabletop display systems are currently being promoted, but the optimal ways to use these systems to display a large amount of information or how to interact with them have not been considered. This paper describes information presentation and interaction technique for a large number of files on a round tabletop display system. Three layouts are explored on our augmented table system: sequential layout, classification layout, and spiral layout. Users can search and find files by virtually rotating the circular display using a "hands-on" technique.

1 Introduction

In small group meetings, people often bring their laptop PCs to see digital files, to write memos, or to access the Internet. However, if we want to share information displayed on the screen, we often face some difficulties. For example, the laptop's display is too small to share and it is hard to be seen by people other than its owner. We often use a projector and a vertical screen. However, only the presenter, the owner of the PC, can manipulate the information using input devices of the PC.

To address such issues, horizontal circular display systems, which use front or rear projection to display information onto the tabletop, have been developed [2, 6, 7]. On these round tables, people around the table can share the displayed information. They can change the orientation of the display by rotating the information to see it from the right perspective. Moreover, since the displayed information is close to each user, users can directly point and manipulate the information by using their fingers.

Although previously proposed Systems [2, 6, 7] demonstrated these advantages of the round tabletop display systems, they have not considered how to

N. Sebe, M.S. Lew, and T.S. Huang (Eds.): HCI/ICCV 2005, LNCS 3766, pp. 141–149, 2005.

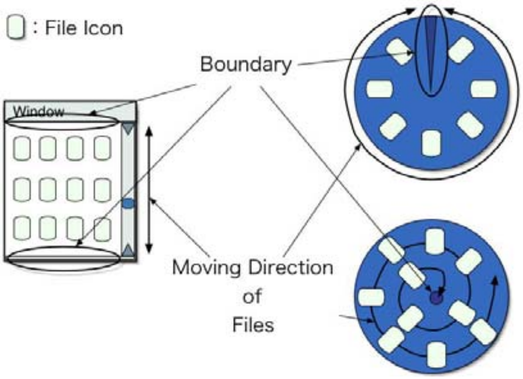


Fig. 1. Scrolling using a window system (left) and using a rotation table (right)

display a large amount of information and how to find a particular file in this amount of information.

This paper describes a method of displaying and interacting with a large amount of information on a round tabletop display system, and also describes the application of this method for displaying and searching image files.

2 File Layout

In traditional GUI environments, standard methods are used to display a large amount of information. For example, it is usually displayed in a large virtual area and different parts of the information are viewed by using a window with scroll bars.

This technique could be simply applied to the round table. However, it has to be considered that the table must be usable from any direction. That is, the orientation of each file should be maintained so that it always faces the users correctly.

It should be also considered how files appear or disappear on a round table. In window systems, there are natural boundaries at the top, bottom, left, and right of the window where files appear or disappear. On the other hand, the round table does not have such natural boundaries.

3 Implementation

Figure 2 shows an overview of our system. The system is composed of a table, an LCD projector, two CCD cameras, two PCs (Pentium 4 2.8GHz, 512MB memory, Linux) for image recognition, and one PC (Pentium 4 2.8GHz, 512MB memory, Windows) for image generation. The CCD cameras capture the images on the table, and these images are processed by the two image-processing PCs. To recognize users' hands and fingers, we used a real-time finger tracking method we previously developed [3]. As image-processing software, we used Intel's OpenCV

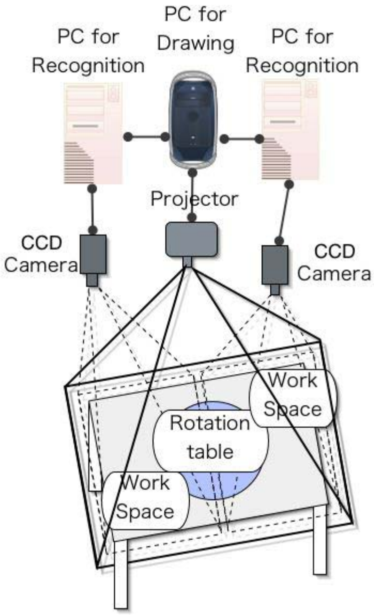


Fig. 2. System architecture

library. Each PC can recognize up to two hands in about 10 frames/sec. Then the computer-generated images are projected on the table. The size of the projection image is 67cm x 84cm.

4 Presentation

This section describes how a large amount of information is displayed on a circular table that is physically limited in size.

The files are sorted based on the feature that the user selected. The basic features are the filename, the creation date, and the file size. In addition, we used the mean value (0 to 255) of the hue, saturation, and brightness of the image file.

After the files are sorted, they are laid out on the table by using one of three layouts: sequential layout, classification layout, or spiral layout.

4.1 Sequential Layout

In sequential layout, the sorted files are laid out sequentially on the table as shown in Fig. 3. There is a boundary in the circle. When the user rotates the table, files appear or disappear at the boundary. Currently 23 files that are the same size are simultaneously displayed.

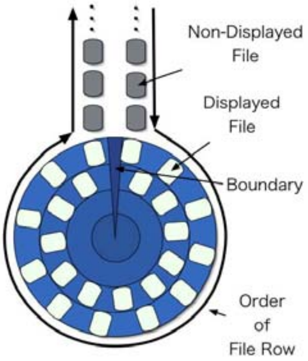


Fig. 3. Sequential layout

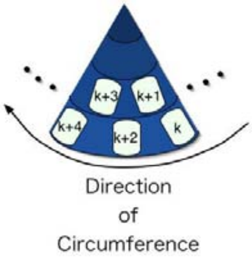


Fig. 4. Relation between arrangement and order

4.2 Classification Layout

In classification layout, files are also laid out sequentially, but they are classified based on a feature that users selected (Fig. 5). A maximum of 6 groups and 36 files may be displayed.

The user can select the method of classification, such as the filename, the creation date, the file size, or some image features. In the case of classifying by filename, the user can select the number of characters to be used to sort files between 1, 3, 5, and 7. In the case of classification by creation date, the user can select sorting options involving the year, month, day, and hour. In the case of classification by file size, the user can select values between such as 1MB, 100KB, 10KB, and 1KB. In the case of classification by an image feature such as color, the user can classify files using hue, saturation, or brightness.

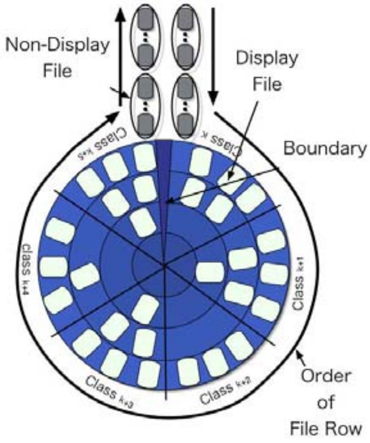


Fig. 5. Classification layout

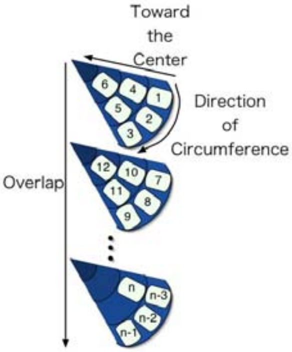


Fig. 6. Relation between arrangement and order

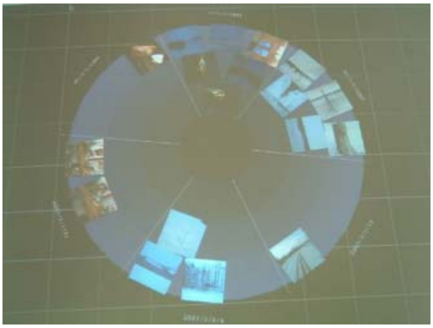


Fig. 7. Screenshot of classification layout

4.3 Spiral Layout

In spiral layout, after the files are sorted, they are laid out to make a spiral from the center to the edge of the circle. The size of the files increases as they go to the edge. The relation between the size of the icons and the angle of deviation is shown in Fig.9.

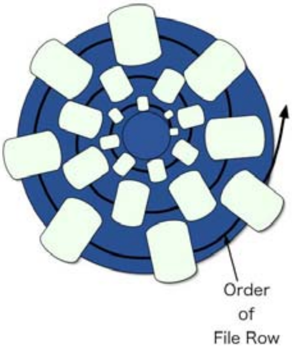


Fig. 8. Spiral layout

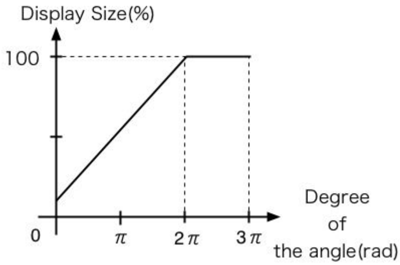


Fig. 9. Relation between the files' angle and display size

5 Interaction

This section describes interaction capabilities of our rotational table. Basically, the users are able to do multiple manipulate at a time.

Moving and copying: When the user pinches a file with a thumb and pointing finger, the file is in selection mode and the user can move it by moving his/her hand with two fingers closed. If the user moves the file from the inside of the circular area to the outside, or from the outside to the inside, the file is copied.

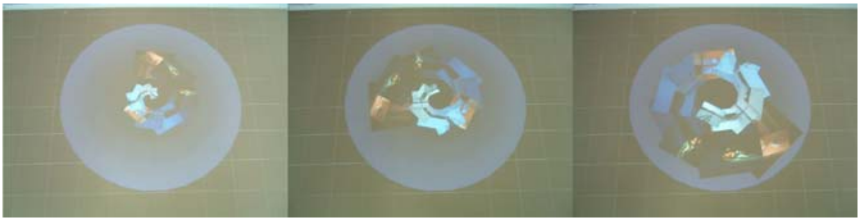


Fig. 10. Screenshot of spiral layout

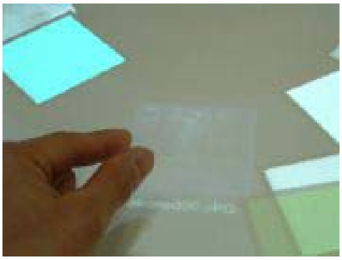


Fig. 11. Move

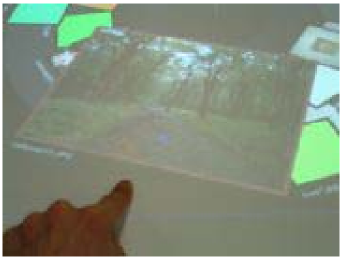


Fig. 12. Zoom

Zooming: If the user points the file in sequential layout or in classification layout, the file is automatically magnified as shown in Fig. 12.

Rotating: When the user shows five fingers and moves his/her hand inside the circular area, all the images are virtually rotated.

If multiple users try to rotate, the rotation table responds to the hand that reached it first.

Menu: When the user points at the background in the circular area with his/her left hand, the structured menu appears (Fig. 15). The user can select each menu item with the right hand.

The user first selects the feature used to sort files. Then, the second menu appears where the user selects the layout method.

In the whole of the system, only one menu appears at a time. If multiple users call up the menu at a time, it appears in front of the user who called it up first.

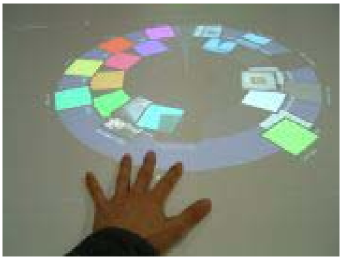


Fig. 13. Rotation

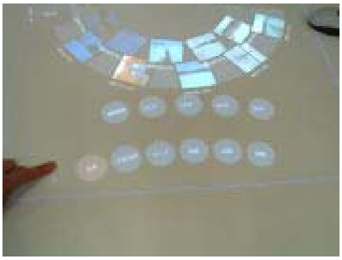


Fig. 14. Menu

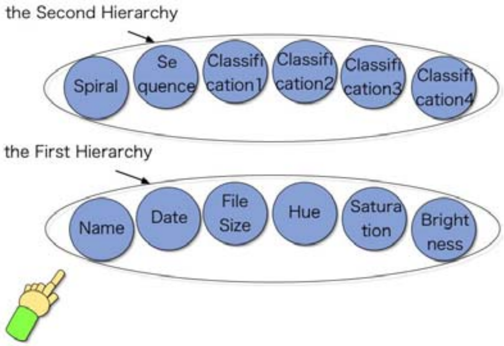


Fig. 15. Menus

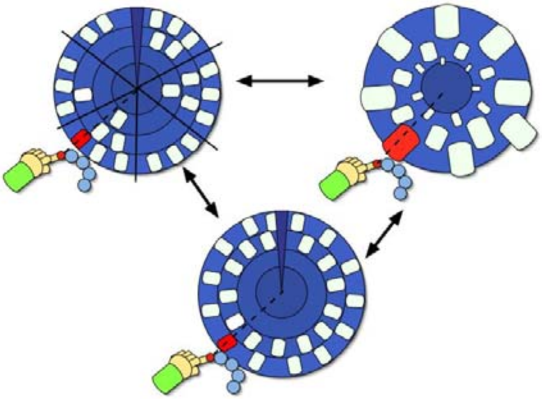


Fig. 16. Relation between the position of the menu and the position of a file

6 Discussion

The sequential layout is a basic layout method in which the scroll metaphor of a window environment is directly applied to a circular display. It is useful when the user searches files sequentially because the files are sorted sequentially and the user can see as many files as possible. However, the displayed size of the file is fixed, and therefore the user needs to magnify the focused file one by one.

On the other hand, the classification layout sorts the files based on the feature the user selected. It is useful particularly when the user searches for files for which he/she already knows the creation date, the file name, and so on. However, because the files in each class are laid out as they are stacked, it is sometimes not possible to browse all the files in the class at a time.

In general, there is a trade-off between the displayed size of the files and the number of displayed files. Therefore, it is important to consider the balance between the visibility of each file and the browsability of many files. The sequential layout and classification layout are examples of this trade-off because the number of displayed files and the displayed size of the files are fixed.

On the other hand, spiral layout uses a focus+context technique in which the size of files is gradually increasing. The user can see the detail of some files while seeing as many files as possible. Moreover, the user can interactively move his/her focus just by rotating the table.

Considering the visibility and the browsability of the three layouts, it seems apparent that the sequential layout and the classification layout are better for browsing because the number of displayed files is greater than that in the spiral layout. However, the spiral layout provides more visibility because the user can zoom files interactively just by rotating the table.

In the sequential and classification layouts, there exists a boundary where the files appear or disappear, and this is not good for the user near the boundary.

On the other hand, in the spiral layout, the files appear from the center of the table and all the people can see the files equally.

7 Related Work

Digital Desk [4] is the system that merges the physical desktop and the electronic desktop into one. Capturing the user's hands on the desk by the camera above it, and projecing the digital information on it, the system allows the user to manipulate the digital information as it is displayed on the physical desktop.

Augmented Surfaces [5] is the system that integrates the laptop's desktop, the table surface where the laptop is laid, and the wall. The user is able to move the digital information in the laptop to the table and wall by dragging it out of the laptop's workspace.

Media Table [6] is the table where people can communicate by seeing and manipulating objects projected on the table. The objects on the table move freely. The user can gather the objects near him/her by touching the table. However, Media Table did not discuss issues involving the visualization of a large amount of information. The number of users who can interact with the table is limited to one because the table uses a normal touch panel.

PDH (Personal Digital Historian) [7] is a tabletop system for collaborative work. The information is laid out according to the annotation of time, place, and so on. Participants visualize the folder hierarchy by using Hyperbolic Tree [8].

ARTHUR [10] is the tool for structuring a 3D model of the round table. The users have a high-resolution see-through head mounted display, and manipulate by using tangible interfaces and hand gestures. Because the users can see the same virtual objects and other participants at the same time, this system is efficient for collaboration.

Sunburst [9] is a radial, space-filing visualization for the file/directory hierarchy. Keeping the context, this visualization can focus the peripheral parts of the hierarchy that are generally difficult to examine in hierarchy structure visualization. While this system shows the meta-information of file size, file type, and so on, our system displays the content of the files.

One of advantages of our system is that it allows multiple interactions because of using computer vision technology. Another advantages of our system is that it provides three layout methods in order to visualize and search a large number of files.

8 Conclusion

This paper described layout and interaction methods for a large number of files on a round tabletop display system. We proposed three different layout methods: sequential, classification, and spiral layout.

References

1. Intel Open Computer Vision Library http://www.intel.com/research/mrl/research/opencv/
2. H. Koike, S. Nagashima, Y. Nakanishi, Y. Sato, EnhancedTable: Supporting a Small Meeting in Ubiquitous and Augmented Environment, Proc. of 5th Pacific Rim Conference on Multimedia(PCM2004), Part I, pp. 97-104, 2004.
3. K. Oka, Y. Sato and H. Koike, Real-time tracking of multiple fingertips and gesture recognition for augmented desk interface systems, Proc. IEEE Int'l Conf. Automatic Face and Gesture Recognition (FG 2002), pp.429-434, May 2002.
4. P. Wellner, The DigitalDesk Calculator: Tangible Manipulation on a Desk Top Display, Proceedings of UIST'97, ACM Press, pp. 27-33.
5. J. Rekimoto, M. Saitoh. Augmented Surfaces: A Spatially Continuous Work Space for Hybrid Computing Environments, ACM Conf. on Human Factors in Computing Ssytem (CHI'99), pp.378-385, 1999.
6. J. Misawa, K. Tsuchiya, K. Yoshikawa, MediaTable: A Computer Screen System with Nonspecific Direction in Presenting Information, Proc. of The 9th Workshop on Interactive Systems and Software(WISS2001), pp. 173-178, 2001.
7. C. Shen, N. Lesh, F. Vernier, Personal Digital Historian: Story Sharing Around the Table, ACM Interactions, Vol. 10, Issue 2, pp. 15-22, March/April, 2003.
8. J. Lamping, R. Rao, Laying out and visualizing large trees using a hyperbolic space, Proceeding of the 7th annual ACM symposium on User interface software and technology, November, 1994.
9. J. Stasko, E. Zhang, Focus+Context Display and Navigation Techniques for Enhancing Radial, Space-Filling Hierarchy Visualizations, Proceedings of the IEEE Symposium on Information Visualization 2000(InfoVis'2000), pp. 57-65.
10. ARTHUR http://www.fit.fraunhofer.de/projekte/arthur/

On-Line Novel View Synthesis Capable of Handling Multiple Moving Objects

Indra Geys[1] and Luc Van Gool[1,2]

[1] ESAT/PSI-VISICS,
Katholieke Universiteit Leuven
Kasteelpark Arenberg 10, 3001 Leuven, Belgium
{igeys,Luc.Vangool}@esat.kuleuven.be
http://www.esat.kuleuven.ac.be/psi/visics/
[2] D-ITET/BIWI,
Swiss Federal Institute of Technology (ETH)
Gloriastrasse 35, 8092 Zürich, Switzerland
vangool@vision.ee.ethz.ch
http://www.vision.ee.ethz.ch

Abstract. This paper presents a new interactive teleconferencing system. It adds a 'virtual' camera to the scene which can move freely in between multiple real cameras. The viewpoint can automatically be selected using basic cinematographic rules, based on the position and the actions of the instructor. This produces a clearer and more engaging view for the remote audience, without the need for a human editor.
For the creation of the novel views generated by such a 'virtual' camera, segmentation and depth calculations are required. The system is semi-automatic, in that the user is asked to indicate a few corresponding points or edges for generating an initial rough background model. Next to the static background and moving foreground also multiple independently moving objects are catered for. The initial foreground contour is tracked over time, using a new active contour. If a second object appears, the contour prediction allows to recognize this situation and to take appropriate measures. The 3D models are continuously validated based on a Birchfield dissimilarity measure. The foreground model is updated every frame, the background is refined if necessary. The current implementation can reach approx 4 fps on a single desktop.

1 Introduction

We are witnessing a quickly growing interest in systems for teleconferencing and tele-teaching (see e.g. [1]). Most approaches however involve a major (hardware) investment and use dedicated conferencing rooms, permanently staffed with a video crew. Moreover they still rely on hand editing of the video streams. What we envision is kind of the opposite. We target an intuitive and low-cost teleconferencing system which can operate in a semi or even fully automatic mode. The setup only needs a desktop computer and a few low-end static cameras placed around an instructor, in a regular conferencing room. New interpolated views of

N. Sebe, M.S. Lew, and T.S. Huang (Eds.): HCI/ICCV 2005, LNCS 3766, pp. 150–159, 2005.

the scene are synthesized between the real camera viewpoints. As such a ‘virtual’ camera is created, that can move freely between the real cameras in the scene. The system can automatically select a good viewpoint depending on the position and the actions of the instructor, following cinematographic rules. The basic assumption for our system to work is that there is one major foreground object, e.g. the main instructor. The background is assumed quasi static. As otherwise this would pose too stringent constraints to its use, the system also can deal with ‘secondary’ independently moving objects, such as a person passing by or an object of the background being moved.

This said we’re not the first nor the only one focusing on low-cost conferencing systems. However most research in this area covers one-to-one conferencing, instead of one-to-many tele-teaching. Mostly two persons are sitting behind their desks and communicate to each other over a network. Billinghurst *et al.* [2] developed such a system which presents to a user an animated face of her/his conversation partner. This system is implemented as augmented reality and requires a head mounted display. As such, the user is free to move, but still is seriously hindered by the wearable display. Later (see the work in [3]) this system was modified towards the use of a desktop interface. More recently, Criminisi *et al.* [4] developed an algorithm for gaze correction in such a setup. A camera is mounted at both sides of a computer screen, and as such a stereo pair is formed. The algorithm used is an accelerated version of scanline based dynamic programming. The stereo maps are used to generate an interpolation of the person’s upper body. More recently they extended this system by taking opacity effects at the object’s border into account [5]. The major differences with the system we propose, is that we target ‘tele-presenting’. The user is offered the possibility to give a remote presentation in a regular conference room and is not locked in front of his desktop. The recorded video, is automatically edited on-the-fly and transmitted to a remote audience.

The generation of depth maps at high speeds is a crucial sub-step of our algorithm. Only recently, the computation of dense depth maps in real time has become feasible without dedicated hardware (see e.g. [6, 7]). In [6] a plane sweeping algorithm is described, while Ansar *et al.* [7] use bilateral filtering. Assembly level optimizations using special extensions of the CPU instruction set (such as MMX) are also used extensively [8]. In contrary to these algorithms, we rather use a coarse to fine strategy to calculate the depth. First a bounding volume is determined. A more accurate depth measurement is obtained by a global approach. Small artifacts are removed and opacity effects at the borders of the foreground object are taken into account. A balanced use of both GPU and CPU underlies the implementation.

The outline of the rest of the paper is as follows. Section 2 gives a general overview of the system. The interactive part of our pipeline is covered in 3, while section 4 focuses on the on-line interpolation. Experimental results are presented in 5, and section 6 concludes the paper.

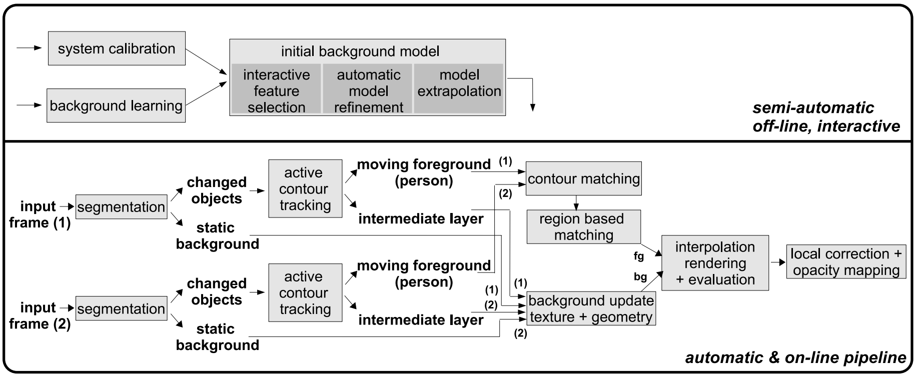

system calibration
background learning
initial background model
interactive feature selection
automatic model refinement
model extrapolation
semi-automatic off-line, interactive
input frame (1)
segmentation
changed objects
static background
active contour tracking
moving foreground (person)
intermediate layer
input frame (2)
contour matching
region based matching
background update texture + geometry
fg
bg
interpolation rendering + evaluation
local correction + opacity mapping
(1)
(2)
automatic & on-line pipeline

Fig. 1. Overview of the system, working with two input cameras. Processing comprises an interactive and an on-line part. The on-line processing pipeline is executed for every frame

2 System Overview

Our setup comprises an interactive and an on-line part, which is executed automatically for every frame. (See also the overview in fig. 1) The input to our system are the video-streams coming from two or more static cameras. They are calibrated off-line and on beforehand. This is done using auto calibration (see e.g. [9, 10]). The search for robust multi frame correspondences, necessary for the calibration, is facilitated by illuminating the scene with a laser pointer. Before starting the on-line mode it is up to the user to provide 'clean' images of the background only. During a short 'learning phase', the algorithm for background segmentation collects a robust initial version of the reference background. These reference images will be updated during the on-line phase with the parts of the background which remained static. Finally the user is asked to select a limited number of corresponding points or lines based on an intuitive user interface, in order to build up an initial background model. This step is not strictly necessary, but yields better results for complicated scenes.

During the on-line phase, the output of the segmentation algorithm points us to places where the images did change. In contrast to most other approaches, this change is no longer attributed *a priori* to the moving foreground object. By use of tracking of the foreground contour, changes in the background or so-called intermediate layers are set apart from the real moving foreground (e.g. the person). This will be explained in more detail in the subsequent sections. As soon as all 'layers' –background, intermediate and foreground– are identified, the foreground contours are matched between the different cameras based on their outlines only. Finally pixel based matching based on this initialization, and a validation step using the Birchfield dissimilarity refine the updated model, and add opacity effects where required. The information from the intermediate layers is taken into account by updating both the texture and the 3D structure of the background model. The newly computed models are rendered as the resulting interpolation.

3 Interactive Background Modeling and Refinement

An initial background model is generated interactively based on the reference background images. Alternatively it can be constructed automatically, with a plane sweep algorithm. However this technique is error prone if the background scene contains strong depth discontinuities, or only limited texture. We prefer to ask the user to bootstrap the algorithm, and visually validate the outcome. As such we can overcome this otherwise hard to solve problem in a robust way.

Besides modeling the overlapping parts of the input frames, the background model is extrapolated. This allows to show a background rendering for all intermediate camera positions, and not only for those looking at the common part of the scene in the input cameras.

3.1 Initial Background Model

At startup the user launches the system in *background learning* mode. After a few seconds acquisition stops, and s/he is presented an interface with all different views of the cameras lined up. Fig. 2 illustrates this interface for two cameras. Both straight edges and salient corner points are detected and indicated on all images. If the user selects a feature – corner point – by use of the mouse, the corresponding epipolar line is drawn in all other views. If no feature is present in a 10 × 10 region around the selection, the point itself is used.

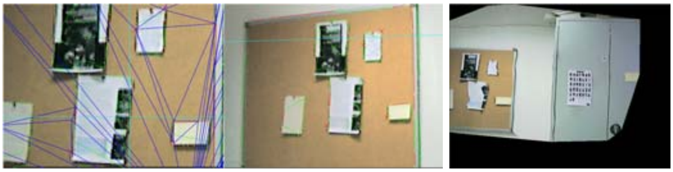


Fig. 2. Left: The Delaunay triangulation for the currently selected point pairs. Quickly zooming in allows for faster and more accurate operation. Middle: Corners and edges visualized on the reference backgrounds. Right: Partially completed rough 3D model, seen from an extrapolated viewpoint at the left side of the inputs

Now the user has to select roughly the same point in one of the other views. This approximative correspondence is projected orthogonally to the corresponding epipolar line, and a local search for the best epipolar match is started. By back projecting the 3D point corresponding to the initial selection, we obtain a possible correspondence in all views. This allows for matching over all views simultaneously. To this end we maximize the sum of the normalized cross correlations (see eq. 1) for a window around the image point selected and those in all other views.

$$NCC = \frac{\sum I_1(x,y) * I_2(x,y)}{\sqrt{\sum I_1^2(x,y) * \sum I_2^2(x,y)}} \tag{1}$$

Modeling based on corresponding points only, would require a rather large amount of selections to obtain an acceptable model. As a consequence this would be too time consuming. Therefore the user can also select corresponding edges. Since they typically originate from salient textures or from depth discontinuities, they are especially apt to be used as constraints for the model. The correspondences for the points on two edges are confined to those which obey the epipolar geometry. In other words, only the 'common part' of the edges between two views are used. All matched points or edges are indicated on the input images to avoid multiple selections.

During selection, after each user interaction a constrained Delaunay triangulation is computed. The point selections serve as corner points for the triangles, the lines are constraints for the triangulation which are not allowed to be divided. As such we obtain a 3D mesh of the background, which is incrementally improved by selecting more points or edges. The model update and selection is done at interactive rates, there are no noticeable delays to the user. The 3D locations of all features are calculated by triangulation, based on the camera matrices known from the calibration. This incrementally improved model can be manipulated and is covered with the input textures. It provides visual feedback to the user, who decides when to stop modeling or where to add or remove points. In the next stage this model will automatically be refined where necessary.

3.2 Refinement Using the Birchfield Dissimilarity

Visualization is not the only purpose of projecting the images on the model. At the time of rendering the model, also the Birchfield distance [11] is computed for the blended textures. This distance $\mathrm{D}(x_L, x_R)$ is a pixel-wise dissimilarity measure that is insensitive to image sampling. It will be used as a robust measure for the accuracy of the model. Where the model is correct, the textures map perfectly and the dissimilarity will be small. Where the dissimilarity is large, the model still has to be improved.

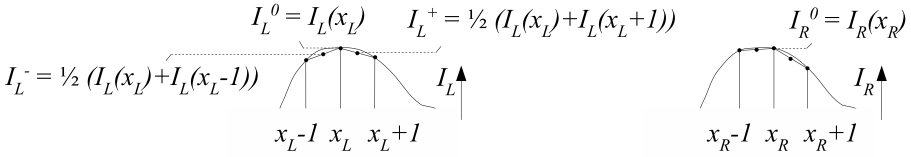


Fig. 3. The Birchfield dissimilarity measure between two intensity images I_L and I_R

The pixel-wise calculation of this dissimilarity is implemented as a fragment program, which is executed on the graphical board (GPU) during rendering. Consider two corresponding pixels x_L and x_R, resulting from the left and a right camera image. First the linearly interpolated intensities ${I_L}^-$ and ${I_L}^+$ halfway between x_L and its neighbors are determined, as illustrated in fig. 3. Let $I_{Lmin} = \min({I_L}^-, {I_L}^+, {I_L}^0)$ and $I_{Lmax} = \max({I_L}^-, {I_L}^+, {I_L}^0)$, with I^0 the intensity of the pixel under consideration, then $\mathrm{D}(x_L, x_R)$ is defined as follows:

$$D_1 = \max(0, {I_L}^0 - I_{Rmax}, I_{Rmin} - {I_L}^0)$$

$$D_2 = \max(0, {I_R}^0 - I_{Lmax}, I_{Lmin} - {I_R}^0) \tag{2}$$

$$D(x_L, x_R) = \min(D_1, D_2)$$

We now investigate (CPU based) the mean Birchfield dissimilarity for every triangle of the model, computed on the GPU. If the average for a triangle exceeds a threshold (empirically chosen), this triangle is automatically 'subdivided'. Corner points within the triangle are searched, and are matched. Again the matching is restricted by the epipolar geometry, moreover it is acceptable to confine the search range to a depth window around the range spanned in space by the triangle. The resulting matched corners are all added to the Delaunay triangulation. The matching itself is based on a normalized cross correlation (NCC) computed over 3x3 windows (see eq. 1).

This subdividing is iterated, and each time the Birchfield dissimilarity is recomputed and checked. After the first iteration the subdivision is extended. Next to adding matched corner points, a new vertex is created on the middle of each edge. This means we split every triangle into four smaller triangles, if good correspondences can be found. Since these new vertices are located on joined edges of two triangles, both triangles are subdivided. This gives the mesh the possibility to represent finer scene structures where they are needed. This is done without risking to diverge towards an incorrect solution as the rough outline of the geometry largely constrains its large scale behavior.

3.3 Model Extrapolation

The model built with the help of the user interface, is determined by the point- and edge-correspondences provided by the user. As such this model will be limited to the area which is visible in more than one camera image. However, we want to obtain a realistic background model that is as complete as possible. Therefore it is necessary to extrapolate the model to allow for background rendering at all camera positions.

To this end a least squares regression is used to obtain the planar continuation of the 3D mesh. A necessary set of points of the resulting plane are added to the Delaunay triangulation. As a result the model is continuously extended for the regions visible in only a single input camera.

During visualization the part of the model visible in the 'virtual camera' is determined. For the regions on which more than one input texture project, the textures are blended. The blending factor is gradually adapted for the position of the virtual camera to compensate for intensity differences between the inputs. In the extrapolated regions for which only one input texture is available, this input is projected on the model and the brightness is adjusted locally during the projection. This to integrate smoothly with the other parts of the model. A visualization of such an extrapolated model is shown in fig. 4.

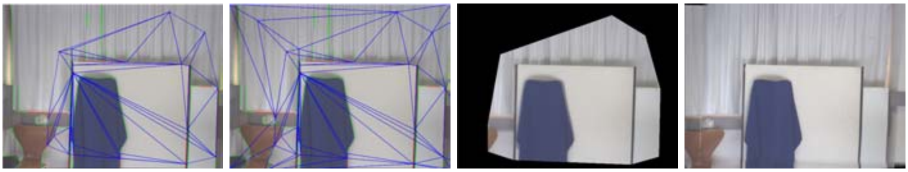


Fig. 4. (a) The Delaunay triangulation obtained by selection within the area visible in two cameras; (b) after the model extrapolation. (c) Partial model seen from a virtual middle camera after the initial selection; (d) extrapolated model

4 On-Line Interpolation and Model Update

As stated before, a segmentation algorithm based on background differencing is used to extract moving objects from the scene. In earlier work [12] it is demonstrated how the contour information of one segmented object can be matched on a shape basis only. This delimits a bounding volume in 3D which is considerably smaller than what is retrieved by the visual hull, for the same segmentation. Hence further pixel based matching will be more robust. More details on the matching algorithms and how to deal with opacity effects at the object's borders can be found in [12].

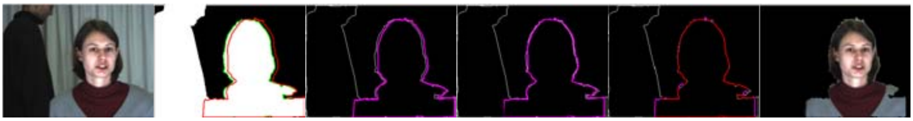


Fig. 5. (a) Input image (b) Segmentation result shown in white. Foreground contour from previous frame visualized on top. (c-d) Initial contour is moved towards new position. (e) Contour is fit to the contour of the segmentation. (f) Final foreground segmentation

However, the situation becomes more complicated if there are multiple moving objects and/or persons. Suppose a person walks behind another person. Logically this action will be segmented out (see fig. 5 (a,b)). However, in one camera view the segmented areas will be 'connected' sooner than in the other due to the depth discontinuity. At that moment the contours will have different outlines in the different cameras (for one camera still separate objects are found, while for the other a single undivided foreground is detected), and shape based matching of the contours will fail. Such a failure can also happen in case of a partial occlusion of one object by another in more than one view. It is more effective to treat both objects separately even if they occlude each other. The secondary moving objects will be referred to as 'intermediate layers'. Iterative contour matching over time and a new active contour allow to separate the foreground from this layer in a robust way.

4.1 Foreground Contour Extraction

As we focus on processing streaming video, the movement of the person between subsequent frames is assumed small. As such the outline of the foreground contour will be very similar over time, until a partial occlusion occurs. To this end the Hausdorff distance between each contour and its predecessor in the previous frame is computed. If this distance suddenly peaks, we discard the current contour, as it will not be the outline of the current foreground object. Additional measures should be taken.

The foreground contour – represented as list of pixels – extracted in the previous frame, denoted by $C^{t-1} = \{p_i, i = 1...K\}$, will be used as an initialization of the contour in the current frame, $C^t = \{q_j, j = 1...L\}$. However this old contour should first be aligned to its correct location in the current frame. To this end we propose an algorithm inspired by ICP:

- build a set of closest point pairs (p_i, q_j) between the two contours. The search is confined to a 10x10 neighborhood around the p_i.
- compute the translation T, that minimizes the sum of squared distances $\sum dist(p_i - T, q_j)^2$. The error measurement *dist*, is the Euclidean distance.
- apply T to C^{t-1}

Repeat the above steps until convergence. As at the onset of an occlusion the contours will still greatly have the same outline (only where foreground and intermediate layer join a mismatch occurs), this algorithm moves the old contour to its new location in a few iterations. The consecutive transformations applied are shown in fig. 5 (b-d).

If the translated contour nearly coincides with the boundary of the new segmentation (max 3 pixel is used as a threshold in our tests) the latter is used, otherwise the translated historic contour is retained. (see fig. 5 (e)).

Finally the shape of the contour is refined starting from this initial rough location. This mainly allows us to deal with non rigid transformations of the foreground object, which typically occur in case of a person. To this end an active contour (or snake) is applied which jointly minimizes bending and curvature and pulls the outline towards the edges. The result is that we get an exact contour for the foreground, and effectively separate it from the intermediate layer. Something which could not be done using the mere segmentation result. Now, after contour matching, classical pixel based matching across all views can determine the geometry of the foreground object.

4.2 Background Update

The remaining blobs, segmented out by the background differencing point us to the intermediate layer. All of these are taken care of by a local update of the background, in order to represent the changing scene. Locally the matching process is repeated as before (see eq. 1), but based on the latest input images, containing the moving objects. Further subdivision of triangles is done where needed, as explained in subsection 3.2. All parts of the background which did not change are used to update the background reference images.

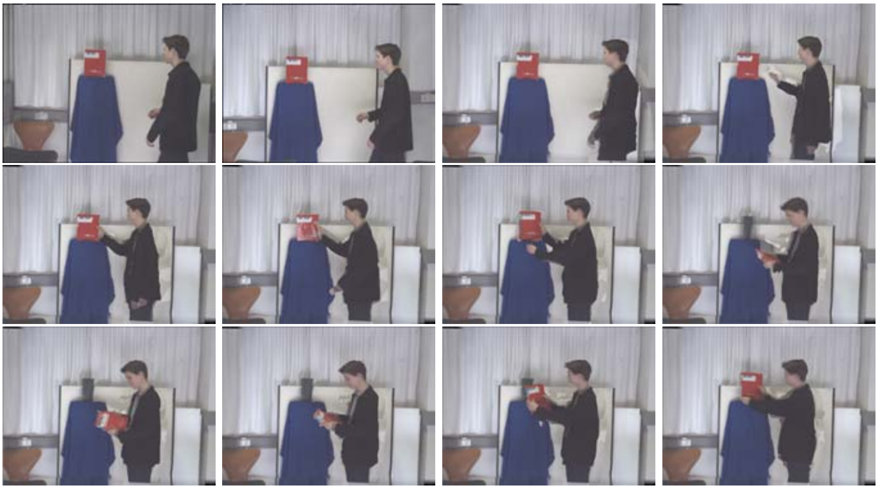


Fig. 6. The first two images are the inputs for the first frame. The other images show several frames of an interpolation result. The virtual camera is located in the middle of the baseline between the two input cameras

5 Results

Fig. 6 shows a set of shots from a sequence of interpolations obtained with only two input video streams. The position of the virtual camera is taken exactly in the middle of the baseline, as this is the position which is most challenging. The more we move towards a real camera, the better the viewpoint and the texture mask the errors. The baseline for this test was 1 m, while the person is at approx. 2-3 m. The person is the main foreground object. He grabs a part of the background (the book), manipulates it and places it back. During this series of operations the book and the changing background become part of the intermediate layer. The geometry is continuously updated. Note that the object behind the book is correctly visualized after its removal. Without the automatic background texture and geometry update this would not be possible. Some ghosting around the foreground is caused by the flicker introduced by the lamps.

The background 3D model generated only by the user selection of features, still contains some artifacts. These faults are detected by calculating the Birch-

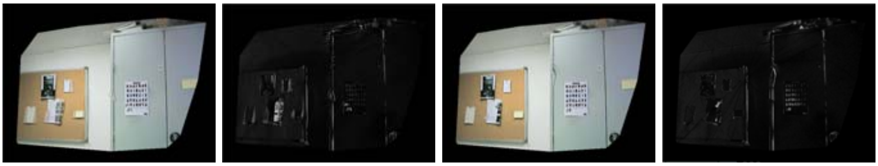


Fig. 7. (a) Model built from features selected by the user. (b) Birchfield dissimilarity per pixel from this model. White areas have a high dissimilarity value. (c) Refined model. (e) Birchfield dissimilarity after automatic model refinement

field dissimilarity between the textures projected on the model, as can be seen in fig. 7 [left]. Fig. 7 [right] illustrates the first automatic refinement of this model and the Birchfield dissimilarity recalculated after this refinement.

6 Conclusion

We propose an algorithm for novel view synthesis, which is amenable to real time processing. The system strives to offer a high level of automation, in combination with feedback and interaction to the user where necessary. This comprises the user's validation of the (initial) background model, and his/her help to bootstrap the system. As we only need a desktop computer and a set of low end consumer grade cameras the system is very low cost, without losing the added value offered by video stream editing.

In this work we mainly focus on an integrated system. We also propose a possible solution to the segmentation and tracking problem in case multiple independently moving/changing objects are present within the scene. A more engaging and compelling tele-presence can result, without an extended cost.

Acknowledgments. The authors gratefully acknowledge support from the K.U.Leuven GOA project 'MARVEL', and the ETH project 'BLUE-C-II'.

References

1. Arapis, C., Konstantas, D.: Design and implementation of a teleteaching environment. In: Ninth DELOS Workshop. (1999) 79–83
2. Billinghurst, M., Bowskill, J., Jessop, M., Morphett, J.: Wearable spatial conferencing space. In: ISWC'98. (1998) 76–83
3. Billinghurst, M., Kato, H.: Real world teleconferencing. In: CHI'99. (1999) 194–195
4. Criminisi, A., Shotton, J., Blake, A., Torr, P.: Gaze manipulation for one-to-one teleconferencing. In: ICCV'03. (2003) 191– 198
5. Criminisi, A., Blake, A.: The SPS Algorithm: Patching Figural Continuity and Transparency by Split-Patch Search. In: CVPR'04 (1). (2004) 342–349
6. Yang, R., Welch, G., Bishop, G.: Real-time consensus-based scene reconstruction using commodity graphics hardware. In: Pacific Graphics'02. (2002) 225–235
7. Ansar, A., Castano, A., Matthies, L.: Enhanced Real-time Stereo Using Bilateral Filtering. In: 3DPVT'04. (2004) 455–462
8. Hirschmüller, H., Innocent, P.R., Garibaldi, J.: Real-time correlation-based stereo vision with reduced border errors. International Journal of Computer Vision **47** (2002) 229–246
9. Triggs, B.: Autocalibration and the absolute quadric. In: CVPR'97. (1997) 609
10. Pollefeys, M., Koch, R., Van Gool, L.: Self-calibration and metric reconstruction inspite of varying and unknown intrinsic camera parameters. International Journal of Computer Vision **32** (1999) 7–25
11. Birchfield, S., Tomasi, C.: A pixel dissimilarity measure that is insensitive to image sampling. IEEE Trans. Pattern Anal. Mach. Intell. **20** (1998) 401–406
12. Geys, I., Van Gool, L.: Hierarchical coarse to fine Depth Estimation for Realistic View Interpolation. In: 3DIM'05. (2005) 237–244

Resolving Hand over Face Occlusion

Paul Smith, Niels da Vitoria Lobo, and Mubarak Shah

Computer Vision Lab, School of Computer Science University of Central Florida
{rsmith,niels,shah}@cs.ucf.edu

Abstract. This paper presents a method to segment the hand over complex backgrounds, such as the face. The similar colors and texture of the hand and face make the problem particularly challenging. Our method is based on the concept of an image force field. In this representation each individual image location consists of a vector value which is a nonlinear combination of the remaining pixels in the image. We introduce and develop a novel physics based feature that is able to measure regional structure in the image thus avoiding the problem of local pixel based analysis, which break down under our conditions. The regional image structure changes in the occluded region during occlusion. Elsewhere the regional structure remains relatively constant. We model the regional image structure at all image locations over time using a Mixture of Gaussians (MoG) to detect the occluded region in the image. We have tested the method on a number of sequences demonstrating the versatility of the proposed approach.

1 Introduction

The task of segmenting the hand over complex backgrounds such as the face is a challenging problem. The difficulty lies in the fact that the hand and head are similarly colored/textured regions. A necessary step for many HCI applications such as gesture recognition, pointing interfaces, hand pose recognition, and event detection is a reliable hand segmentation. Sign language recognition methods also need to first segment the hand over complex boundaries, such as the face. Some events like coughing, eating, and taking medication could be more easily recognized by segmenting the hand from the face. In short there are many applications that could benefit from having a robust segmentation of the hand over complex backgrounds.

We propose two main contributions in segmenting the hand over complex backgrounds such as the face. First we develop a new feature based on the force field image [10]. The force field image uses concepts from force field transformations used in physics. Basically, each image location is represented by a vector value which is a nonlinear combination of all other pixels in image. Their approach focused on a possible feature space for recognition of faces and uses single frames. The feature we develop is the distance traveled by test pixels placed in the force field. Our novel feature is able to model regional structural changes in the image over time. Local methods (pixel based) cannot resolve the occlusion because there is little change in local color when similarly colored objects occlude each other. Regional structure in the image does change when the hand occludes the face, although local pixel colors in the occluding region remain largely the same before and during the occlusion. By quantifying the regional structural change in an image over time we can resolve this kind of occlusion.

N. Sebe, M.S. Lew, and T.S. Huang (Eds.): HCI/ICCV 2005, LNCS 3766, pp. 160–169, 2005.

The second contribution is in presenting a method that is able to model our newly developed feature response over time and capture where and when occlusion is happening using a Mixture of Gaussians (MoG) modeling paradigm. We also clarify several concepts from [10] and give more details in using this image representation. An extension of the force field computation to video data is also given.

Section 1.1 gives previous work. Section 2 provides details on the image representation. Section 3 shows how formulating the problem using MoG can aid the task of segmenting the hand/face. Results are presented in Section 4, and then we conclude.

1.1 Previous Work

Much of the work in finding the hand in a complex background relies on colored markers [7] on the hands or requires the hand to be the only skin object in view [5]. Contour based approaches [1] [9] [11] and other edge based methods [14] rely on good edges separating the hand and head, which are often not present in such difficult occlusion. Active contour approaches [1] [11] require the hand shape change to be small. Our method has no such constraint. In [9], hand shape is estimated over a complex background by using a shape transition network with the attributes of contour, position, and velocity. They use a simple template based approach and skin color segmentation to find the hand during hand face occlusion. Their approach is sensitive to small changes in lighting, different skin colors, and requires small differences in the 2D hand shapes. Other color based approaches [3] [12] [14] would have difficulties in segmenting the hand over face. In [12] body parts are tracked using Bayesian Networks but the conditional probabilities are specified manually. Further, skin color is used to find the body parts. In [8] examples are given handling a few frames of occlusion using shape and color in a Bayesian framework, but it is unclear if it can withstand occlusion involving hundreds of frames (as our approach does). In [18] hand tracking is performed using eigen dynamics analysis, but the hand tracking system uses pretrained hand models. It is unclear how person-independent these models are.

In [2], a method is presented which uses multiscale features to find the hand. Color priors are used, requiring retraining for new people. This method will not work when the face is present because of the stronger blobs and ridges on the face. [19] performs well on segmenting hands, but the method requires that the hand cover a large portion of the image. Our image sequences frequently have only part of the hand in the image.

An Elastic Graph Matching approach is given in [15], which uses color models to find skin regions. It has problems when the illumination changes, as the skin color model fails. Each training image requires manual labeling of at least 15 node points. The approach has problems handling geometric distortions of the hand as does [16]. Our approach is not hand model based, so we do not have this limitation.

In [6] an approach is given that segments the hand from a complex background. They localize the hand using motion information and map this region to a fovea vector. The method does not extend to other people. There is significant change in hand size which our method can cope with. Most model based approaches presented above fail in the case where the hand is only partially visible in the image or for gestures not in the database.

Because of the similar colors of the hand and face, segmentation algorithms such as [4] will generally either under or over segment the hand/face occlusion. In principle, one can do tracking but then the question becomes how to initialize the tracking. Further, tracking methods generally fail when tracking across similarly colored regions.

Background subtraction [13] will not work in segmenting the hand over the face because even a slight movement of the head will trigger a large change of foreground pixels. Further, supposing the head was relatively fixed, the underlying problem with the RGB (and other color spaces) input domain is in the similarity of the head and hands. These methods cannot distinguish between the head and hand colors. Most background subtraction methodologies operate on RGB or some other color space (i.e. the input space is color information). When similarly colored objects, occlude each other the individual pixel values in the region of occlusion give little information considered individually because the objects are similarly colored. This presents difficulties for individual pixel based methods.

2 Potential and Force Images

We can define the smoothed potential at a given position, $\mathbf{r}_j$, with respect to position $\mathbf{r}_i$, in image I as

$$E_i(\mathbf{r}_j) = \frac{I(\mathbf{r}_i) + I'(\mathbf{r}_i)}{2 \cdot |\mathbf{r}_i - \mathbf{r}_j|} \tag{1}$$

where $\mathbf{r}_j$ is the image location in question and $I(\mathbf{r}_i)$ is the image intensity at position $\mathbf{r}_i$. I' represents the image intensity at the previous time instant. Because we are dealing with video data, we introduce temporal smoothing into the force field representation to account for spurious noise.

Equation 2 gives the potential energy for a particular image location. This computation is then performed for every location in the image. This gives the potential energy image. The total potential energy at location $\mathbf{r}_j$ is given by:

$$E(\mathbf{r}_j) = \sum_{\mathbf{r}_i \not= \mathbf{r}_j} E_i(\mathbf{r}_j) = \sum_{\mathbf{r}_i \not= \mathbf{r}_j} \frac{I(\mathbf{r}_i) + I'(\mathbf{r}_i)}{2 \cdot |\mathbf{r}_i - \mathbf{r}_j|} \tag{2}$$

2.1 Force Fields

To find the force exerted by all pixels at a particular image location $\mathbf{r}_j$ simply compute

$$F(\mathbf{r}_j) = \sum_{\mathbf{r}_i \not= \mathbf{r}_j} E_i(\mathbf{r}_j) \frac{\mathbf{r}_i - \mathbf{r}_j}{|\mathbf{r}_i - \mathbf{r}_j|^2} = \sum_{\mathbf{r}_i \not= \mathbf{r}_j} \frac{I(\mathbf{r}_i) + I'(\mathbf{r}_i)}{2} \frac{\mathbf{r}_i - \mathbf{r}_j}{|\mathbf{r}_i - \mathbf{r}_j|^3} \tag{3}$$

We can see that the force is a vector as it has magnitude and direction. These vector fields will be very important in the image representation. The units of pixel intensity, direction, and force are arbitrary as is the origin of the coordinate system. $\overline{F}(\mathbf{r}_j)$ is the normalized vector at $\mathbf{r}_j$. Examples of the potential and force fields are shown in Figure 1. Since the force fields are two dimensional the magnitude and direction are shown separately. The direction was quantized (for display purposes only) into 10 regions.

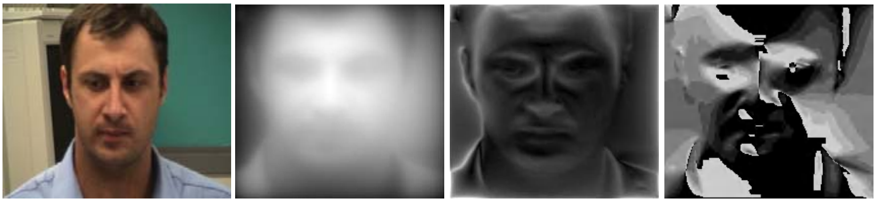


Fig. 1. Potential and Force Vector Fields for various input frames. Input images are shown on the top left. Potential image is next. Next is magnitude of the force field and the last contains the direction (quantized) of the force field

2.2 Finding Potential Wells

Once the potential and force field images have been computed the well points(local extrema) are computed. This is done in an iterative fashion. Unit test pixels are placed uniformly (resulting in a rectangular grid of test pixels) throughout the image. They can be placed at every pixel, every other pixel etc. They are placed in the field and serve to capture the flow of the field. Suppose there are m test pixels $t_1, \ldots, t_m$. Since the position of each test pixel will change as it traverses the force field, we denote the initial location of t_i as $t_{i,0}$. To find any $t_{i,j}$ apply the recursive equations:

$$t_{i,0} = (x_i, y_i)$$

$$t_{i,j} = t_{i,j-1} + \overline{F}(t_{i,j-1}) \tag{4}$$

Where $\overline{F}(x)$ is the normalized vector at x, which is computed as $\overline{F}(x) = \frac{F(x)}{|F(x)|}$. Given a unit test pixel starting point, $t_{i,0}$, it goes through the force field until it stabilizes at a well point, denoted as $t_{i,N}$. Unit test pixels eventually reach stable points. In our examples N=500. Convergence was always reached well before N=500, but we could test for convergence to allow more than 500 iterations. The computation could be ended earlier if convergence is reached. Iterations needed for convergence depends on image size and the number of wells. Larger images or ones with fewer wells will need more iterations, but 500 iterations was sufficient for the 1000's of image we tested. Not all t_i end up at the same wells. The path that a test pixel takes is called a channel. It is easy to see that once two test pixels reach a common point, they both travel the same path from them on. Before deriving the distance traveled feature we would like to give some intuition as to what information in the image the force field is capturing and why it is useful in our problem domain. Equation 3 shows that the force field captures global structure, technically. However since the effect on the field is proportional to $\frac{1}{d^2}$ the net effect is that the force field captures regional image structure.

The potential image is a scalar at each pixel and it is a measure of the brightness of that region. The force field is a vector at each pixel location. It measures properties related to regional edge strength. It is not an edge detector, but it is related. The force field measures regional edge like structure in the image. The potential wells are those points in the force field where the net force is zero. Intuitively these are the points that seek to position themselves in between the regional edge structures of the image. A well equalizes the force (regional structure) around itself. See Figure 2 for an example of a synthetic image demonstrating these ideas.

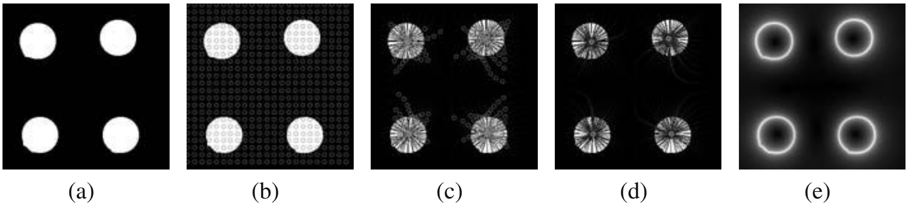


Fig. 2. This is a synthetic image used to give some intuition of the force field representation. (a) is the original image. (b) is the original image with the initial configuration of test pixels overlaid on it. (c)-(d) show the movement of the test pixels through the force field after 50 and 200 iterations. (e) shows the magnitude of the force field

The force field captures regional structure and we can model this regional structure over time to detect structural changes in the image. Though the hand and head have similar color and texture, by analyzing regional image structure we are able to capture structural changes that are introduced when the hand enters the scene. We can see that other methods monitoring pixel wise information are not enough because of the similar texture of the hand and head. When the hand enters, the local structure would not change (i.e. the pixel values remain largely the same), but there is useful regional structure variation (we will show examples of this change in subsequent sections). We now detail how we model this changing force field over time.

3 Developing New Image Feature

The structure of these field lines for a particular image sequence are relatively constant until the hand (or anything else) enters the image. Once the hand enters a clear disturbance in the channels occurs in the region of occlusion. This hypothesis has been borne out in experiments on thousands of video frames. It is consistent with the fact that the force field is a measure of regional image structure. Figure 3 shows an example of this phenomenon. It can be seen that most of the channels are stable before and during the occlusion. We could show more examples, but due to space limitations, we will not. We next demonstrate how to measure and quantify this changing force field.

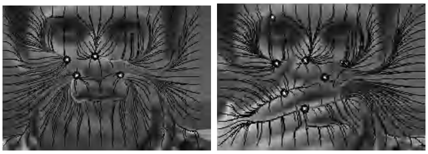


Fig. 3. Channels before and during occlusion. Notice that a disturbance in the channels can be seen in the lower left corner of the image, whereas the rest of the channels in the image are relatively stable

If test pixels are placed uniformly in each image we can measure the variation a certain test pixel exhibits in the distance it travels to a potential well. Since these distances remain relatively constant when there is no disturbance in the image (i.e. no hand/face occlusion), the distance that each test pixel travels can be modeled as a random variable with Gaussian distribution. When the hand enters, the wells and the distances that the test pixels travel will vary significantly. These will be the foreground channels, and they are somewhat analogous to foreground pixels in background subtraction.

The reason this occurs is that when another object is introduced, it has its own set of channels and wells. When the two objects merge, the channels and wells of both objects interact with one another. Although the hand and face are similar in color, the potential and force structure present in the image changes when another object enters the scene. Using the MoG modeling technique we are able to measure and localize this change, which allows us to find the boundary between the face and the hand. The distance from a test pixel start location to its final well position can be measured by computing

$$d = |t_{j,0} - t_{j,N}| \tag{5}$$

This is the new distance traveled in a force field feature. Other distance measures such as the arc length could be used. In any case, the distances the test pixels travel are relatively constant until the hand enters the facial region. We model the face before occlusion in terms of the distance traveled at each test pixel start location using a MoG.

Let us assume that in the first video frame for a particular test pixel t_j: $|t_{j,0} - t_{j,N}| = X_0$. In the next video frame for the same test pixel location we can compute $|t_{j,0} - t_{j,N}| = X_1$. Given the distance traveled history of a particular test pixel at location t_j: $X_0, X_1, \ldots, X_\tau$, we want to model this density as a mixture of K Gaussians. The current distance traveled by t_j, X_τ, at time τ, has probability

$$P(X_\tau) = \sum_{i=1}^{K} w_{i,\tau} \frac{1}{\sqrt{2\pi}\sigma_{i,\tau}} e^{\frac{-(X_\tau - \mu_{i,\tau})}{2\sigma_{i,\tau}^2}} \tag{6}$$

of belonging to the current model. $w_{i,\tau}$ is the weight of the i^{th} Gaussian, and $\mu_{i,\tau}$ and $\sigma_{i,\tau}$ are the mean and variance of the distribution all at time τ. If none of the Gaussian distributions match for this particular location t_j, the least likely distribution is replaced by the new distance. The distribution's mean is the distance traveled by t_j, $|t_{j,0} - t_{j,N}|$, with the weight of this distribution set low. At each time instant the weights of the K distributions are updated as

$$w_{i,\tau} = (1 - \alpha) w_{i,\tau-1} + \alpha(M_{i,\tau}) \tag{7}$$

with α set to a constant (learning rate) and $M_{i,\tau}$ being an indicator function which is 1 for the distribution that matched and 0 otherwise. The distribution i that matched the current distance observation has its mean and variance updated as

$$\mu_{i,\tau} = (1 - \rho)\mu_{i,\tau-1} + \rho X_\tau \tag{8}$$

$$\sigma_{i,\tau} = (1 - \rho)\sigma_{i,\tau-1}^2 + \rho(X_\tau - \mu_{i,\tau})^2 \tag{9}$$

In our case ρ is set to a constant. For notational convenience we denote t_{j_u} as the mean of the distribution that matched for test pixel t_j. Using this approach we are able to model the distances traveled by each test pixel in a coherent manner. The next task is to use these models to segment the hand from the face.

3.1 Extracting the Hand

There are two steps needed to extract the hand. We must first identify whether or not the frame has a hand in it. A good measure is when the maximally changing test pixel's distance from its distribution is much larger than its change in the previous frame. This indicates a large change in the image. Concretely, we say the hand has entered when

$$t_{l_\mu} - X_\tau > 3 \cdot (t_{l_{\mu-1}} - X_{\tau-1}), \tag{10}$$

$$\text{where } l = \underset{l}{\text{argmax}}\; t_{l_\mu} - X_\tau, l = x \tag{11}$$

t_{l_μ} is the mean of the distribution for t_l, and X_τ is the current distance traveled observation (computed as $|t_{l,0} - t_{l,N}|$ for t_l. $t_{l_{\mu-1}}$ and $X_{\tau-1}$ are the mean of the distribution and observation for t_l at the previous input frame.

The goal is to find the set $\mathbf{H}$ which is all the hand pixels. Initially set $\mathbf{H} \leftarrow t_{l,x}, \forall x \ni 1 \leq x \leq N$. This only gives one t_i and corresponding channel. To get the full hand, any test pixel which ended up at the same well is also assumed to be part of the hand. Further, any test pixel whose well is within β pixels is assumed to be part of the hand. Concretely, set

$$\mathbf{H} \leftarrow \mathbf{H} + t_{a,x}, \forall a, x \ni |t_{l,N} - t_{a,N}| \leq \beta, 1 \leq x \leq N \tag{12}$$

These test pixels and corresponding channels taken together segment the hand region. Once the hand enters the head region, the distances test pixels travel will vary greatly. This variation should not be learned, so the models are not updated after the hand enters the head region. Figure 4 shows three frames of the found channel lines. The final segmentation is achieved by finding the convex hull of this point set $\mathbf{H}$ and drawing the hull. Other methods could be used to improve the resulting contour. The full algorithm is given in Table 1. Detailed results are presented in Section 4.

Table 1. Overall Algorithm

For every frame

1. Compute force at every pixel using Equation 3
2. Place test pixels t_i uniformly and $\forall t_i$ compute Equations 4 and 5
3. $\forall t_i$Use Equations 6 - 9 to update online MoG models
4. Check for hand using Equations 10 and 11
5. If hand present, segment using Equation 12, find convex hull and display result

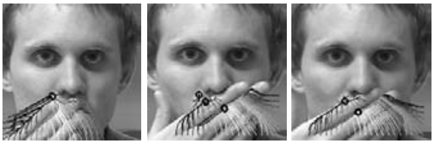


Fig. 4. Channels superimposed on hand region. These channels varied most from the previous model. A convex hull algorithm could be used to fill in this hand region

4 Results

Our method was tested on 14 sequences involving hand/face occlusion for a total of 1800 frames. Not all of these frames contained hand over face occlusion. Of course the non-occlusion frames were needed in order to build the online distance models. Out of the 1800 frames, roughly one half contained hand over face occlusion. The method was successful under a variety of lighting conditions. We assume that the hand is initially not present (which allows us to build the model). In order to allow translational invariance and to have faster processing,we find the head region using [17] and only process these regions. We model every 5^{th} pixel in both directions for faster computation. More samples would increase segmentation rates and the contour accuracy. Figures 5 and 6 show results of the hand segmentation on different input sequences. We should note that in Figure 5 the head starts out frontal and then rotates to a half-profile position. Our method is able to cope with this type of rotation, after which the model starts to break down. Again to obtain the results we run a convex hull algorithm on the set **H**, described in Section 3.1, and show the hull. The algorithm was always able to determine when the hand entered the image using the steps in Section 3.1. Figure 7 shows a comparison between our proposed method, background subtraction [13], and mean shift segmentation [4] respectively. Our method and [13] give pixel wise segmentation, so comparison was straightforward and unambiguous. We felt it would be interesting to compare against general methods because our approach does not use hand color/shape to improve its decision, meaning it could possibly be applied in other contexts. Neither of these two other methods were successful in segmenting the hand from the face.

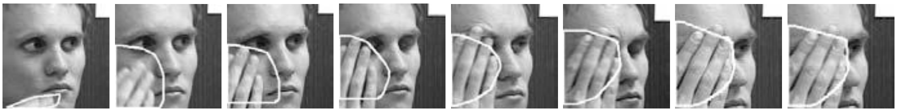


Fig. 5. Hand Segmentation. This was a difficult sequence due to the large face rotation

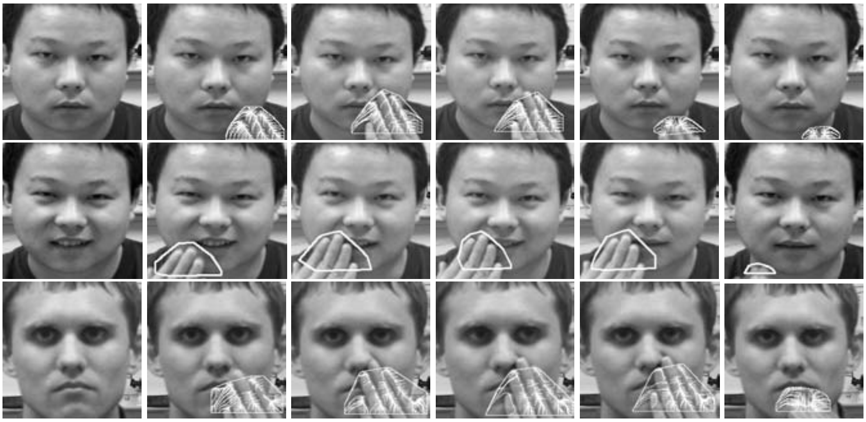


Fig. 6. Hand segmentation results for three sequences. Row 1 shows channel lines superimposed. Row 2 shows the convex hull. The sequence in Row 3 involves occlusion for over 300 frames

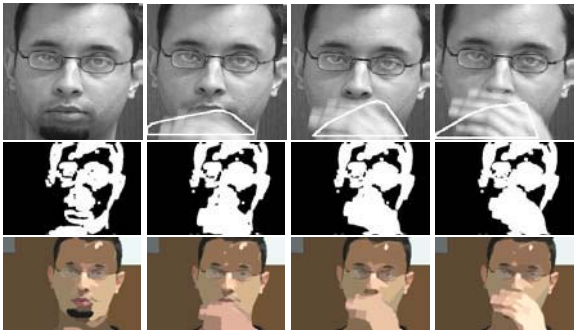


Fig. 7. Segmentation results for our method (row 1), [13] (row 2) and [4] (row 3)

To quantify how well the algorithm performed we manually generated ground truth segmentations for two sequences. Comparisons of our method to ground truth and background subtraction [13] are shown in Table 2. Comparison was made pixel wise. For our method each pixel in the convex hull was counted as hand and each pixel outside was counted as non-hand. The true positive percentages for every frame were added and divided by the total number of frames. A similar method was used for the true negative rate. Our method outperformed [13] in all cases. While [13] segmented part of the hand, it found much of the head region as hand, indicated by the low true negative rate.

Table 2. True positive (TP) and true negative (TN) segmentation % for the specified sequences

Seq #	# Frames	Our Method TP %	Method in [13] TP %	Our Method TN %	Method in [13] TN %
1	44	80.04	72.00	97.11	74.12
9	150	79.53	73.15	96.58	72.19

5 Conclusion and Future Directions

We have developed a method that is successfully able to segment the hand from the face. From a high level the method succeeds because we developed an image feature which is based on regional information. During occlusion of head and hand, local pixel regions remain similar, but the regional image structure changes during the occlusion. Our method detects this change and is able to recover the occluding region. Our main contributions are in development of a novel feature: the distance traveled of a test pixel in the image force field. And in modeling the distance traveled using a MoG, which allowed us to capture occlusion information that is difficult to extract. We want to explore more the force field representation, and determine its limits in resolving occlusion. Better methods of segmentation using the MoG model could be explored. It would be useful to test how well the method resolves occlusion with other types of objects.

References

1. T. Ahmad, C. Taylor, A. Lanitis, and T. Cootes. Tracking and recognising hand gestures, using statistical shape models. *Image and Vision Computing*, 1997.
2. L. Bretzner, I. Laptev, and T. Lindeberg. Hand gesture recognition using multi-scale colour features, hierarchical models and particle filtering. *Automatic Face and Gesture Recognition*, 2002.
3. L. Brèthes, P. Menezes, F. Lerasle, and J. Hayet. Face tracking and hand gesture recognition for human-robot interaction. *International Conference on Robotics and Automation*, 2004.
4. D. Comaniciu and P. Meer. Mean shift: A robust approach toward feature space analysis. *TPAMI*, 2002.
5. Y. Cui and J. Weng. Learning-based hand sign recognition. *Automatic Face and Gesture Recognition*, 1995.
6. Y. Cui and J. Weng. Hand sign recognition from intensity image sequences with complex backgrounds. *Automatic Face and Gesture Recognition*, 1996.
7. J. Davis and M. Shah. Recognizing hand gestures. *ECCV*, 1994.
8. H. Fei and I. Reid. Joint bayes filter: A hybrid tracker for non-rigid hand motion recognition. *ECCV*, 2004.
9. Y. Hamada, N. Shimada, and Y. Shirai. Hand shape estimation under complex backgrounds for sign language recognition. *Automatic Face and Gesture Recognition*, 2004.
10. D. J. Hurley, M. S. Nixon, and J. N. Carter. Force field energy functionals for image feature extraction. In *IVC*, 2002.
11. M. H. Jeong, Y. Kuno, N. Shimada, and Y. Shirai. Recognition of shape-changing hand gestures. *IEICE Transactions Division D*, E85-D No. 10:1678–1687, 2002.
12. J. Sherrah and S. Gong. Resolving visual uncertainty and occlusion through probabilistic reasoning. *BMVC*, 2000.
13. C. Stauffer and E. Grimson. Learning patterns of activity using real-time tracking. *PAMI*, 2000.
14. B. Stenger, A. Thayananthan, P. Torr, , and R. Cipolla. Hand pose estimation using hierarchical detection. *Intl. Workshop on Human-Computer Interaction*, 2004.
15. J. Triesch and C. von der Malsburg. A system for person-independent hand posture recognition against complex backgrounds. *TPAMI*, 2001.
16. J. Triesch and C. von der Malsburg. Classification of hand postures against complex backgrounds using elastic graph matching. *Image and Vision Computing*, 2002.
17. P. Viola and M. Jones. Rapid object detection using a boosted cascade of simple features. *CVPR*, 2001.
18. H. Zhou and T. S. Huang. Tracking articulated hand motion with eigen dynamics analysis. *ICCV*, 2003.
19. X. Zhu, J. Yang, and A. Waibel. Segmenting hands of arbitrary color. *Automatic Face and Gesture Recognition*, 2000.

Real-Time Adaptive Hand Motion Recognition Using a Sparse Bayesian Classifier

Shu-Fai Wong and Roberto Cipolla

Department of Engineering,
The University of Cambridge
{sfw26,cipolla}@cam.ac.uk

Abstract. An approach to increase adaptability of a recognition system, which can recognise 10 elementary gestures and be extended to sign language recognition, is proposed. In this work, recognition is done by firstly extracting a motion gradient orientation image from a raw video input and then classifying a feature vector generated from this image to one of the 10 gestures by a sparse Bayesian classifier. The classifier is designed in a way that it supports online incremental learning and it can be thus re-trained to increase its adaptability to an input captured under a new condition. Experiments show that the accuracy of the classifier can be boosted from less than 40% to over 80% by re-training it using 5 newly captured samples from each gesture class. Apart from having a better adaptability, the system can work reliably in real-time and give a probabilistic output that is useful in complex motion analysis.

1 Introduction

One of the challenges in building a system for sign recognition (and also gesture recognition) is that inter- and intra- personal *variation* may lead to a poor performance. In most situations, different signers may sign in different ways and even the same signer may not sign in the same way all the time (see Figure 4). A classifier that is capable to give a good classification result on one dataset may not be able to give a good result on another set. This idea can be illustrated by Figure 3 that shows an original decision boundary (the line in a lighter intensity) can separate original positive and negative samples ('×' and '•' in a lighter intensity) properly but not for new samples (shown in a darker intensity).

Adaptability is therefore an essential property of a sign recognition system applied on a wide range of users. Instead of training a recognition system using all possible samples, it is sensible to train the system using a limited amount of samples and then re-train it using online samples. As illustrated in Figure 3, an updated decision boundary (the line in a darker intensity) could be estimated based on some of the new samples in order to achieve a good classification result on both new and original samples.

In the past decade, sign recognition was done by exploiting *Hidden Markov Models* (HMMs). In [1], HMMs were directly applied to solve the problem and their extensions such as parallel HMMs [2] and self-organizing HMMs [3] were

N. Sebe, M.S. Lew, and T.S. Huang (Eds.): HCI/ICCV 2005, LNCS 3766, pp. 170–179, 2005.

also proposed to improve the performance. In addition, HMMs have also been extended to perform *adaptive* gesture recognition [4].

Recently, the use of HMMs has been criticised. One major criticism is that using HMMs to recognise gesture requires *large training sets* (e.g. [5]) and this will inhibit the *growth of the vocabulary.* In addition, recent HMMs design analyses *each sign as a whole* without breaking it down into corresponding components (e.g. [6]), making the model more complicated and reducing its *extensibility.* Although recent works (e.g. [5, 6]) provide some alternative solutions to the sign recognition problem and achieve an acceptable accuracy, they have not considered how to improve the adaptability of their recognition systems.

In this paper, an adaptive approach to recognise 10 *primitive movements*, which can be considered as the building blocks in any sign language system, is proposed. Motion recognition is done by exploiting *motion gradient orientation* (MGO) images to form motion features and using a *sparse Bayesian classifier* to map the features into their corresponding classes. The Bayesian classifier is designed in a way so that it can be *re-trained* using *online samples.* The present research has three main contributions. Firstly, by allowing online learning, the *adaptability* and the *accuracy* of the recognition system are raised. Secondly, due to the use of the Bayesian classifier, the final outcome is a *probabilistic* value, which is useful in high-level inference processes that must maintain multiple hypotheses. Thirdly, the classifier maintains a *sparse model*, which facilitates an efficient use of computational resources and leads to a real-time performance.

2 Approach

As mentioned in the previous section, HMMs are not the only choice for performing sign recognition and there are alternative solutions such as [5, 6]. This paper extends the work of Derpanis et al. [6] to allow online training and classifying inputs captured under a wider range of conditions. The basic framework and theory used will be described in the following sub-sections.

2.1 Framework

In [6], Derpanis et al. introduced the idea of *breaking down* signs into constituent *primitive movements* with the aid of linguistic information (e.g. [7]). Sign language recognition can then be considered as recognising the primitive movements and the corresponding sequence. Derpanis et al. used simple and manually defined mapping functions to map the motion data in time-series format into their corresponding movement classes. Their work is thus difficult to be extended. In this paper, we adopt their divide-and-conquer strategy but also exploit a general recognition procedure to increase the extensibility.

This paper focuses on the *hand motion classification* problem (i.e. classifying a given video sequence of hand motion into one of the movement primitives). Motivated by [6], we have 10 *primitive movements* to be classified: (1) upward, (2) downward, (3) rightward, (4) leftward, (5) toward signer, (6) away signer,

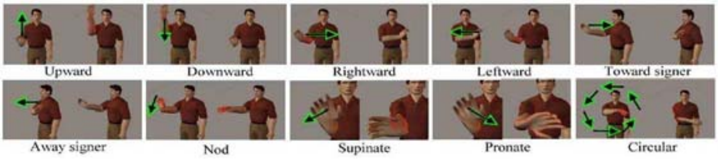


Fig. 1. This figure shows the 10 primitive movements recognised by the proposed system

(7) nod, (8) supinate, (9) pronate, and (10) circular. Figure 1 illustrates these 10 primitives. The classification scheme used will be presented next in sub-section.

2.2 Theory

Unlike recent works such as [5, 6] that are exploiting simple classification schemes, this work is going to handle inter- and intra-personal variation through the use of a powerful classifier. Among several state-of-the-art classifiers, a *sparse Bayesian classifier* or *Relevance Vector Machine* (RVM) is used in the proposed system.

Compared with other state-of-the-art classifiers such as Support Vector Machine (SVM), the RVM classifier performs equally well in term of *accuracy*. In addition, the final outcome of the RVM classifier is a *probabilistic* value instead of a simple true-or-false answer. Furthermore, the *sparsity* of the model stored by the RVM classifier ensures fast and efficient classification process, and this implies the classifier can be implemented in computing devices with limited memory storage such as Pocket PC or Smartphone.

RVM classifier is a simple *binary classifier*. Consider a training set that consists of N motion feature vectors, $\{\mathbf{x_n}, t_n\}_{n=1}^{N}$. The problem of learning a binary classifier can be expressed as that of learning a function f so that the input feature $\mathbf{x_n}$ will map onto their correct classification label t_n and the probability of $\mathbf{x_n}$ is classified as the target class (where $t_n = 1$) equals to $\sigma(y_n) = 1/(1 + e^{-y_n})$where $y_n = f(\mathbf{x_n})$.

The function f can be written as a sparse model where $(M \ll N)$ [8]:

$$f(\mathbf{x_n}) = \sum_{m=1}^{M} \omega_m \phi_m(\mathbf{x_n}) + \omega_0 \tag{1}$$

where $\omega = (\omega_0, ...\omega_M)^T$ are the weights and $\phi_m(\mathbf{x_n}) = K(\mathbf{x_n}, \mathbf{x_m})$ with $K(\cdot, \cdot)$ a positive definite kernel function (where Gaussian Kernel with width 1 is used in the proposed system) and $\mathbf{x_m}$ an example (or a relevance vector) from the training set. Under the RVM framework where hyperparameters $\alpha = \{\alpha_0, ..., \alpha_M\}$ are introduced, *learning* f from the training data means *inferring* ω from the data $\mathbf{t} = \{t_1, ..., t_N\}$ such that the posterior probability over the weights, $p(\omega \mid \mathbf{t}, \alpha)$, is maximised. Given $\mathbf{A} = \mathrm{diag}(\alpha_0, \alpha_1, ..., \alpha_N)$, $\mathbf{B_{nn}} = \sigma\{y_n\}[1 - \sigma\{y_n\}]$ and $\mathbf{\Phi}$ is the $N \times (N + 1)$ design matrix, the optimal values of the weights can be estimated by using an *iterative procedure* [8], where the inverse of a Hessian matrix at 'most probable' weight (ω_{MP}), $\nabla\nabla \log p(\mathbf{t}, \omega \mid \alpha)|_{w_{MP}} = -(\mathbf{\Phi^T B \Phi} + \mathbf{A})$ have to be computed for a current, fixed values of α at each loop. The values of α can be inferred from the training data such that the marginal likelihood $p(\mathbf{t} \mid \alpha)$ is maximised. The iterative procedure for estimating ω and α is repeated until some suitable convergence criteria are satisfied.

In a *batch-learning* approach, all training examples will be considered as relevance vectors at the initial stage and the irrelevance vectors will be 'pruned' after re-evaluation of α in each iteration. In other words, every α_i has a finite value at the beginning and the Hessian matrix to be computed in each estimation loop has a size of $(N+1) \times (N+1)$ initially, where N is the number of training samples. Since inversion of Hessian matrices is involved in the learning algorithm, the overall *training complexity* is $O(N^3)$. This implies that if the initial sample size is huge, the learning algorithm may take a long time to converge.

According to [9], we can also start with an *initially small model* and sequentially 'add' relevance vectors to increase the marginal likelihood. Considering the marginal likelihood, or equivalently, its logarithm $\mathcal{L}(\alpha)$:

$$\mathcal{L}(\alpha) = \log p(\mathbf{t} \mid \alpha) = -\frac{1}{2}[N \log 2\pi + \log \mid \mathbf{C} \mid + \mathbf{t}^T \mathbf{C}^{-1} \mathbf{t}] \tag{2}$$

with $\mathbf{C} = \mathbf{B}^{-1} + \mathbf{\Phi} \mathbf{A}^{-1} \mathbf{\Phi}^T$. From the analysis given in [9], $\mathbf{C}$ can be rewritten in this way: $\mathbf{C} = \mathbf{B}^{-1} + \sum_{m \not= i} \alpha_m^{-1} \phi_m \phi_m^T + \alpha_i^{-1} \phi_i \phi_i^T = \mathbf{C}_{-i} + \alpha_i^{-1} \phi_i \phi_i^T$, where $\mathbf{C_{-i}}$ is $\mathbf{C}$ without basis vector i. $\mathcal{L}(\alpha)$ can be therefore rewritten as:

$$\mathcal{L}(\alpha) = \mathcal{L}(\alpha_{-i}) + \frac{1}{2}[\log \alpha_i - \log(\alpha_i + s_i) + \frac{q_i^2}{\alpha_i + s_i}] \tag{3}$$

where $s_i = \phi_i^T \mathbf{C}_{-i}^{-1} \phi_i$ and $q_i = \phi_i^T \mathbf{C}_{-i}^{-1} \mathbf{t}$. From [9], estimation of α, which gives maximum value of marginal likelihood, can be computed directly from:

$$\begin{array}{ll} \alpha_i = \frac{s_i^2}{q_i^2 - s_i}, & \text{if } q_i^2 > s_i, \\ \alpha_i = \infty, & \text{if } q_i^2 \leq s_i, \end{array} \tag{4}$$

The implication of this evaluation method for α is that we can make discrete changes to the model while we are guaranteed to increase the marginal likelihood. This means we can start from an *initially small model* and *test* the 'relevance' of each new input vector i *sequentially.* When vector i is in the model (i.e. $\alpha_i < \infty$) but $q_i^2 \leq s_i$, then vector i should be removed (i.e. α_i set to ∞); When vector i is not in the model (i.e. $\alpha_i = \infty$) and $q_i^2 > s_i$, vector i should be added (i.e. α_i set to a certain optimal value). Classification model is thus built incrementally.

By adopting this *incremental training* approach, computational complexity is $O(M^3)$ where M is the number of relevance vectors and $M \ll N$. In other words, the *training time* can be *reduced* dramatically. In addition, this learning approach allows any new input to be *evaluated on the fly* and to be added to the model if certain criteria are fulfilled. This property can be used to develop an *online adaptive recognition system* where online training is needed.

3 Implementation Details

The hand motion recognition problem addressed in this paper can be divided into 2 tasks, namely feature extraction and classification.

3.1 Feature Extraction

Extraction of MGO: In this work, a motion gradient orientation (MGO) image is extracted directly from a raw video input and transformed to a motion feature vector that contains necessary spatial-temporal information. MGO image was proposed by Bradski and Davis [10] to explicitly encode image changes introduced by motion events. The MGO is computed from a motion history image (MHI) and a motion energy image (MEI) [11]. MHI is an image that shows moving edges where the recency of a motion is represented by intensity level; MEI is a binary image that indicates where the current moving edges (moving regions) are; Pixels in MGO encode the change in orientation between nearest moving edges shown on the MHI and the region of interest is defined as the largest rectangle covering all bright pixels in MEI. The MGO therefore contains information about *where* and *how* a motion occurred. The MGO obtained within the region of interest will be rescaled to a standard size, which is 200 × 200 in the proposed system. Typical MGO images corresponding to the 10 primitive movements used are illustrated in Figure 2.

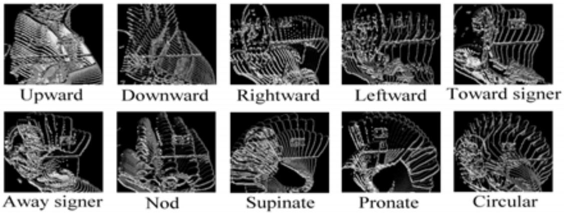


Fig. 2. This figure illustrates the MGO images corresponding to the 10 primitive movements classified by the proposed system

Dimension Reduction: In order to reduce the necessary *number of training samples* (which is proportional to the dimension size), dimension reduction is done on the training MGO images that can be potentially very large in image size (i.e. the dimension size). Principal Component Analysis (PCA) is used in the proposed system to reduce the dimension of the MGO images. By performing PCA on all training MGO images, the eigenvalues indicate that the first 12 components provide an adequate summary of all the images, which account for 95% of the variation. Thus, the first 12 eigenvectors are chosen as the new basis functions for converting any new incoming MGO image into a new feature vector. Finally, normalization is done to give a final feature vector ($\mathbf{x}$) with zero mean and standard deviation of one in all dimensions.

3.2 Learning and Classification

Binary Classification: As explained in the previous section, given a training set $\{\mathbf{x_n}, t_n\}_{n=1}^N$, a RVM classifier can be trained to separate positive and negative samples in an iterative manner through maximising the marginal likelihood. Under the incremental learning scheme, each sample in the training set will be *tested sequentially* to determine whether it will be included in or excluded from the classification model. Decision boundary will be updated once sample vector is added to or removed from the model.

Multi-class Classification: The RVM classifier can be *extended* to a multi-class classifier by using the "one-versus-others" method. Since we have a total of 10 classes of motion, 10 independent RVM classifiers are constructed and each of them is trained to separate one class of data from all others. After all the classifiers are trained, the system can be tested by feeding a sample to all the classifiers. In practice, suppose this sample belongs to class i, the classifier which is trained to separate class i data from the others will give the largest output.

Re-training Procedure: If a new sample of class i is used to re-train the RVM classifiers, this sample will become a positive sample for $i-$th RVM classifier while become a negative sample for the other classifiers. All RVM classifiers will be trained separately by evaluating the relevance of this new sample. This sample will be either *added* to or *ignored* by the *classification model* of each RVM classifier. The *decision boundary* of each classifier will also be re-evaluated accordingly. In other words, if the new sample is quite different from the previously trained samples due to inter- and intra-personal variation, the classification model and the associated decision boundary will be adjusted to account for its influence. This implies adaptability can be achieved through the integration of a new training sample to the previously trained model.

4 Experiments

The proposed method was implemented using unoptimised C++ code and the OpenCV library. All the experiments described were executed on a P4 2.4GHz computer with 1G memory.

4.1 Experiments on Synthetic Data

We first utilise a set of synthetic data to illustrate how incremental training scheme improves the adaptability. An initial training set consisted of 2 classes, where class I (denoted by '×' in a lighter intensity) was sampled from a mixture of 2 Gaussians while class II (denoted by '•' in a lighter intensity) was sampled from a mixture of 3 Gaussians. Similarly, a training set for re-training the classifier consisted of 2 classes. Class I was still sampled from the same distribution as those used to generate the initial training set. Class II, however, was sampled from a mixture of 3 Gaussians whose means are *shifted upward* compare to the

Gaussian mixture that generated the initial training set (see Figure 3). The sample points for re-training are shown in a darker intensity.

Firstly, the initial training set was used to train a RVM classifier using the incremental learning scheme. Afterwards, the re-training set was used to re-train the RVM classifier. The training results are illustrated in Figure 3. The plot shows that the decision boundary *adjusts (bends upwards) automatically* to separate the 're-training' samples from different classes. Another RVM classifier was trained by these training sets using batch learning. Similar decision boundary was achieved but the training time was longer (7344 ms vs. 766 ms).

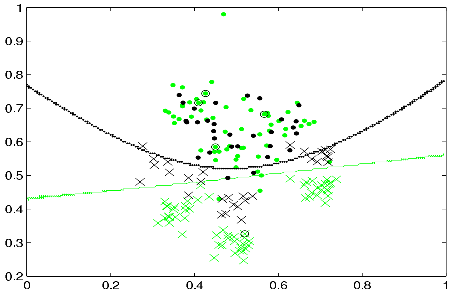


Fig. 3. This figure illustrates the training result on the initial training set (denoted by lighter '×' and '•') by a RVM classifier using incremental learning and also the re-training result on the re-training set (denoted by darker markers) by the same RVM classifier. The decision boundary obtained from training using the initial training set is shown as a lighter line while the decision boundary obtained from training using the re-training set is shown as a darker line. Relevance vectors are shown circled

4.2 Experiments on Real Data

In this part, we will use video data to evaluate the performance of the RVM classifier using incremental learning. Both training and testing data were video captured under arbitrary room conditions (with various backgrounds and lighting). The video was captured by a webcam with a resolution of 320 × 240 pixels at 15 frames per second. In each video clip, the signer signs one of the ten primitive movements as described in Section 2. On average, each movement, which is manually segmented, lasts between 2 and 5 seconds.

We have five pairs of training set and testing set. Different pairs are captured under different conditions. Each dataset has a size of 300 (where each class of movement contributes to 30 samples). The first pair captured the motion of a signer (subject I) who signs the primitive movements using *hand shape 'B5'* (see Figure 4). The second, third and fourth pairs captured the motion of the same signer who signs using *hand shape 'B'*, to sign in a *faster speed*, and to sign with a slight *deviation in direction* respectively. The fifth pair captured the motion of *another signer* (subject II) who signs in the same way as subject I did in the first pair. The difference in capturing conditions between these datasets and their corresponding MGO images generated are illustrated in Figure 4.

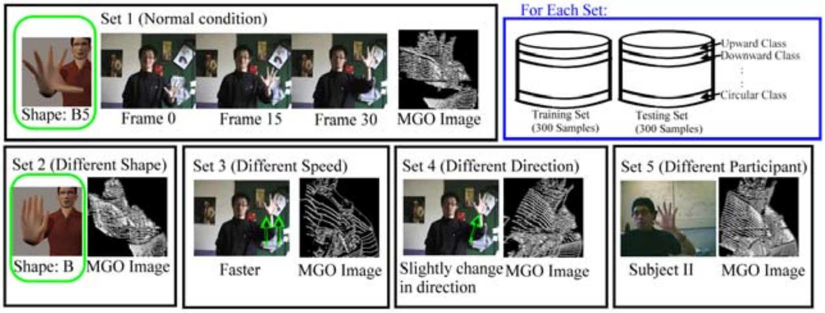


Fig. 4. This figure illustrates the difference in capturing conditions between the datasets used and the corresponding MGO images produced. In all cases, the signers sign the 'Upward' gesture

The training set from the first pair was exploited to train a RVM classifier and a SVM classifier (both have the same kernel configuration). The testing sets of all five pairs of dataset were used to evaluate the performance of these classifiers. The training and testing results are summarised in Table 1.

Table 1. This table shows the training and testing results on all testing sets by a SVM classifier and a RVM classifier

	SVM	incremental RVM
Training Time (ms)	2156	885906
Average No. of Vectors retained	140	3
Classification Time (ms)	15.8	4.2
Accuracy on	Accuracy% (Unrecognised%)	
Set 1 (normal)	0.80 (0.18)	0.93 (0.03)
Set 2 (hand shape)	0.04 (0.96)	0.21 (0.68)
Set 3 (hand speed)	0.05 (0.95)	0.29 (0.57)
Set 4 (direction)	0.06 (0.93)	0.32 (0.53)
Set 5 (subject)	0.12 (0.87)	0.39 (0.59)

The training sets of the remaining four pairs (i.e. Set 2, 3, 4, and 5) were exploited to re-train the RVM classifier. We used *different amount of training data* per each pair to re-train the classifier (sample sizes used are 10, 20, 50, 100, 300). The testing sets of all five pairs were used to test the system. The training and testing results are shown in Table 2.

4.3 Discussion

The experimental results illustrate three main advantages of using incremental RVM for motion recognition. Firstly, a RVM classifier maintains a *sparse model* and can thus perform classification with a *minimum amount* of *online* computational resources. Experiments show that the RVM classifier maintains a sparser

Table 2. This table shows the training and testing results of using RVM classifiers that are re-trained by a different amount of training samples

No. of Re-training Samples per Training Set	10	20	50	100	300
Training Time (ms)	18922	27796	81344	140281	1085406
Average No. of RVs	4.8	6.0	7.4	9.2	12.5
Accuracy on	Accuracy% (Unrecognised%)				
Set 1 (normal)	0.90 (0.03)	0.92 (0.02)	0.90 (0.03)	0.90 (0.04)	0.91 (0.03)
Set 2 (hand shape)	0.35 (0.3)	0.72 (0.11)	0.88 (0.05)	0.93 (0.01)	0.94 (0.01)
Set 3 (hand speed)	0.54 (0.18)	0.66 (0.10)	0.82 (0.04)	0.91 (0.02)	0.93 (0.01)
Set 4 (direction)	0.47 (0.22)	0.63 (0.09)	0.78 (0.04)	0.86 (0.04)	0.89 (0.02)
Set 5 (subject)	0.54 (0.25)	0.82 (0.12)	0.91 (0.04)	0.92 (0.02)	0.92 (0.02)

model than the SVM classifier (3 RVs vs. 140 SVs). The time taken for performing RVM classification on a feature vector is shorter (4.2 ms by RVM vs. 15.8 ms by SVM). Since the time taken for extracting the motion features is 34.3 ms, the total time for performing RVM classification on video data is 38.5 ms (i.e. 26 frames per second). That is to say, the system can run in *real-time*.

Secondly, a RVM classifier returns a *probabilistic result*, which can provide information about uncertainty and facilitate high-level inference. Experiments demonstrate that the number of *unrecognisable samples* is higher if a SVM classifier is used. This is mainly because the RVM classifier has a better *generalisability* than the SVM classifier. Apart from this, the relatively poor classification result of SVM may also be due to the non-probabilistic nature of its output. Under the "one-versus-others" scheme, if all SVM classifiers give '0' response, the system will conclude that the input is not recognisable. In contrast, RVM classifiers give probabilistic values as output, the final decision will be made based on these values and will seldom give unrecognisable results with the exception that all probabilistic values are too low.

Thirdly, a RVM classifier allows incremental learning and thus has a higher *adaptability* to new samples captured under different environment. Experiments indicate that re-training a RVM classifier using a small amount of new samples is sufficient to achieve a fairly *high accuracy* when the classifier is applied on unseen data, which is captured under a different condition. In other words, re-training of a RVM classifier enables a better adaptability of the classifier towards variations, such as changes in hand shape, moving speed and moving direction, and even different person.

Experiments also reflect the main problem of using RVM classifiers is its relatively *long training time* compare with SVM classifiers. Incremental learning, however, does shorten the learning time (7344 ms by batch learning vs. 766 ms by incremental learning). In addition, it is worth spending less than 2 minutes on re-training a RVM classifier with a small amount of new samples to achieve a better classification result.

5 Conclusion

A new method is proposed to increase the adaptability of a gesture recognition system that can be extended to sign language recognition. The proposed method performs better than recently used methods in three ways. Firstly, through the use of an incremental learning approach, newly available samples can be exploited to re-train the classifier that has been trained by an initial training set. Such an online re-training scheme can increase the *adaptability* of the classifier to input captured under a new condition. Secondly, by using a sparse Bayesian classifier that has relatively better *generalisability* and *sparsity*, the final classification result is comparable to other motion recognition methods and the result can be obtained with a minimum amount of online computational resources. Finally, the *probabilistic* nature of the Bayesian classifier implies that the proposed method can be applied in *complex motion* analysis that must maintain multiple hypotheses. A further investigation of how to extend this work to analyse complex motion and how to further reduce the *training time* is under progress.

Acknowledgements

SW is funded by the Croucher Foundation Scholarships.

References

1. Starner, T., Weaver, J., Pentland, A.: Real-time american sign language recognition using desk and wearable computer based video. PAMI **20** (1998) 1371–1375
2. Vogler, C., Metaxas, D.: A framework for recognizing the simultaneous aspects of american sign language. CVIU **81** (2001) 358–384
3. Bauer, B., Kraiss, K.F.: Video-based sign recognition using self-organizing subunits. In: Proc. ICPR. (2002) 282–296
4. Wilson, A., Bobick, A.: Realtime online adaptive gesture recognition. In: Proc. ICPR. (2000) Vol I: 270–275
5. Bowden, R., Windridge, D., Kadir, T., Zisserman, A., Brady, M.: A linguistic feature vector for the visual interpretation of sign language. In: Proc. ECCV. (2004) Vol I: 390–401
6. Derpanis, K.G., Wildes, R.P., Tsotsos, J.K.: Hand gesture recognition within a linguistics-based framework. In: Proc. ECCV. (2004) 282–296
7. Stokoe, W.C., Casterline, D., Croneberg, C.: A Dictionary of American Sign Language. Linstok Press, Washington, DC (1965)
8. Tipping, M.E.: Sparse bayesian learning and the relevance vector machine. The Journal of Machine Learning Research **1** (2001) 211–244
9. Tipping, M.E., Faul, A.C.: Fast marginal likelihood maximization for sparse bayesian models. In: Proceedings of the Ninth International Workshop on Artificial Intelligence and Statistics. (2003)
10. Bradski, G.R., Davis, J.W.: Motion segmentation and pose recognition with motion history gradients. Machine Vision and Applications **13** (2002) 174–184
11. Bobick, A.F., Davis, J.W.: The recognition of human movement using temporal templates. PAMI **23** (2001) 257–267

Topographic Feature Mapping for Head Pose Estimation with Application to Facial Gesture Interfaces

Bisser Raytchev, Ikushi Yoda, and Katsuhiko Sakaue

Intelligent Systems Institute
National Institute of Advanced Industrial Science and Technology (AIST)
Umezono 1-1-1, Tsukuba, Ibaraki, Japan
b.raytchev@aist.go.jp, yoda@ieee.org, sakaue@computer.org

Abstract. We propose a new general approach to the problem of head pose estimation, based on semi-supervised low-dimensional topographic feature mapping. We show how several recently proposed nonlinear manifold learning methods can be applied in this general framework, and additionally, we present a new algorithm, IsoScale, which combines the best aspects of some of the other methods. The efficacy of the proposed approach is illustrated both on a view- and illumination-varied face database, and in a real-world human-computer interface application, as head pose based facial-gesture interface for automatic wheelchair navigation.

1 Introduction

Head pose estimation, the automatic estimation of the orientation of the head relative to a camera-centered coordinate system, is a problem of both theoretical and practical importance. Especially, automatic head pose estimation is expected to be a major feature of future human-computer interfaces. The number of degrees of freedom for head pose variation is limited to three (three angles of rotation: pan, tilt and roll) and can be relatively easily characterized by simple 3D Euclidean geometry. In other words, the set of all facial images generated by varying the orientation of a face is intrinsically a three-dimensional manifold (ignoring or compensating for other types of image variation like changes in scale, illumination etc.), which however is embedded in image space of a much higher dimensionality. Efficient classical techniques for dimensionality reduction like principal component analysis (PCA) or multidimensional scaling (MDS) are available and can be used for learning a parameterization along the underlying degrees of freedom of the manifold when it is embedded linearly in the observation space. Recently, a number of new techniques like Isomap [1], Laplacian eigenmaps [2], etc. have been proposed, which seem to be able to recover the intrinsic geometric structure of nonlinearly embedded data manifolds. Although these have been used mainly for *visualization*, and to a lesser extent for *classification*, it seems reasonable to expect that the low-dimensional topographic (i.e. geometric structure-preserving) feature maps they generate could be useful also for *pose estimation*. Thus, the *general* approach we propose here is to formulate pose estimation as a two-step process: (1) “unfold” the curved high-dimensional data manifold into a *linear* and *low-dimensional* topographic map*;* (2) assuming step (1) has successfully removed the non-linearity from the data, now it would be easy to learn a *linear pose-parameter map* using only a few training samples (provided the resulting embedding

N. Sebe, M.S. Lew, and T.S. Huang (Eds.): HCI/ICCV 2005, LNCS 3766, pp. 180–188, 2005.

is low-dimensional, of course). This would make possible a new *semi-supervised* approach to pose estimation, which would be important if supervised data (in the form of view-labeled training samples) is not readily available or costly/inconvenient to obtain. On the other hand, learning a linear map from the discovered low-dimensional embedding space to pose-parameter space, and testing it on new unseen samples, would provide valuable feedback about the quality of the found embedding (i.e. how successful has been the method in unfolding the nonlinear manifold), in a more objective way than the one usually done for visualization.

Previous work on head pose estimation from 2D images (which we limit our attention to) can be broadly categorized into two groups: (1) shape-based geometric analysis (e.g. [3]) where head pose is deduced from geometric information like configurations of facial landmarks; and (2) template matching (e.g. [4]-[5]) based on nearest neighbor classification with texture templates. Our work is most strongly related to the approach in [6], however, as outlined above, we extend this approach utilizing more powerful nonlinear methods for topographic mapping, which make possible the semi-supervised approach, advocated here. Additionally, as the proposed method does not rely on tracking geometric features like eyes, etc., which can be difficult to detect for example for near-profile poses, it is able to achieve reliable pose estimation for much larger view range. In section 3, we illustrate the efficacy of the proposed approach on a view- and illumination-varied face data and in real-world human-computer interface application, as pose estimation-based facial-gesture interface for automatic wheelchair navigation.

2 Methods

In this section first we briefly review several alternative methods for low-dimensional topographic mapping, which can be used in the general framework outlined in the Introduction for semi-supervised head pose estimation. In section 2-4 we propose a new method, IsoScale, which seems to be more suitable for our task (this will be discussed and experimentally demonstrated below and in section 3), but we will need the other methods both for the sake of comparison later, and to gradually introduce some relevant ideas and notation.

2.1 The Linear Subspace Method

First we briefly describe the PCA-based linear subspace model for 3D view representation, which is similar to [6]. The training data necessary to build the head pose model is represented by the centered data matrix $\mathbf{X} = (\vec{\mathbf{x}}_1 - \vec{\mathbf{x}}_{av}, \ldots, \vec{\mathbf{x}}_N - \vec{\mathbf{x}}_{av})$, where $\vec{\mathbf{x}}_i$ is the ith training sample and $\vec{\mathbf{x}}_{av} = N^{-1} \sum_{i=1}^{N} \vec{\mathbf{x}}_i$. We assume that N training face samples are available, including samples from all available views from several different subjects. A linear subspace model $\mathbf{Y} = (\vec{\mathbf{y}}_1, \vec{\mathbf{y}}_2, \ldots, \vec{\mathbf{y}}_P)^t$ is constructed using the top P principal components (PCs) obtained by PCA by solving the eigenvalue problem $\mathbf{X}\mathbf{X}^t\mathbf{Y} = \mathbf{Y}\mathbf{\Lambda}$. Thus, a parameterization of a certain view sample $\vec{\mathbf{x}}_i$ by $\mathbf{Y}$ (i.e. by linear projection) is given by $\vec{\mathbf{q}}_i = \mathbf{Y}(\vec{\mathbf{x}}_i - \vec{\mathbf{x}}_{av})$, or more generally by matrix

$\mathbf{Q} = \mathbf{YX}$ for the training samples and $\mathbf{Q}_T = \mathbf{YX}_T$ for the test samples, where $\mathbf{X}_T$ contains the test samples (e.g. a multi-view face sequence for a subject whose samples have not been used in the process of building the model). Next, a linear pose parameter map **F,** relating sample $\vec{\mathbf{x}}_i$ to view angle $\vec{\boldsymbol{\theta}}_i$ can be learned from $\boldsymbol{\Theta} = \mathbf{FQ}$, where $\boldsymbol{\Theta} = (\vec{\boldsymbol{\theta}}_1, ..., \vec{\boldsymbol{\theta}}_N)$ is a matrix of the view angles of the training samples. **F** can be computed by singular value decomposition (SVD) to determine the inverse of $\mathbf{Q}^t$:

$$\mathbf{F}^t = \mathbf{V}_Q \mathbf{W}_Q^{-1} \mathbf{U}_Q^t \boldsymbol{\Theta}^t \tag{1}$$

where $\mathbf{V}_Q \mathbf{W}_Q \mathbf{U}_Q^t$ is the SVD of $\mathbf{Q}^t$. Then, the unknown view angles $\boldsymbol{\Theta}_T$ of the test samples in $\mathbf{X}_T$ are given by

$$\boldsymbol{\Theta}_T = \mathbf{FQ}_T = \mathbf{FYX}_T \tag{2}$$

The vector space spanned by the subset of the PCs in **Y** corresponding to the *P* largest eigenvalues provides an optimal parameterization (in the sense of optimal L2 reconstruction error of the training samples) of the multi-view face representations. However, a limitation of this approach is that PCA can only recover the true structure of *linear* manifolds, while the ambient geometry of view-varying face (or other complex 3D objects) manifolds can be highly folded or curved in the high-dimensional input space. This in turn would put limitations on the precision of the resulting model, especially if a large view range is considered (as we are interested here). Thus we expect that the more sophisticated methods described in the following 3 subsections might be able to better recover the intrinsic geometric structure of the nonlinearly embedded data manifolds, i.e. to "unfold" the curved high-dimensional data into a linear and low-dimensional topographic map from where linear pose-parameter maps can be learnt similar to (1)-(2) above.

2.2 Topographic Mapping with LPP

Locality Preserving Projection (LPP) has been proposed recently [7] as a linear approximation of the Laplacian Eigenmap (LE) method [2], a local approach (as it attempts to preserve the local geometry of the data) to the nonlinear dimensionality reduction problem. LE has a number of desirable properties for low-dimensional embedding (see [2] for details), however, as pointed out in [7] it does not produce a transformation function, i.e. there is no way how to map data points which are not in the training data set into the dimensionality-reduced space. LPP overcomes this problem, by finding an optimal linear approximation to the eigenfunctions of the Laplace-Beltrami operator on the manifold. The subspace model **W** is constructed as a matrix with column vectors the eigenvectors corresponding to the smallest *l* eigenvalues of the generalized eigenvector problem $\mathbf{XLX}^t\mathbf{W} = \mathbf{XDX}^t\mathbf{W}\boldsymbol{\Lambda}$, where **D** is a diagonal matrix of the column sums of the similarity matrix $\mathbf{S}_{ij} = \exp(-\|\vec{\mathbf{x}}_i - \vec{\mathbf{x}}_j\|^2 t)$, *t* is a free parameter and **L=D-S** is the Laplacian matrix. The optimal *l*-dimensional embedding is thus given by $\mathbf{Q} = \mathbf{W}^t\mathbf{X}$ for the training samples and $\mathbf{Q}_T = \mathbf{W}^t\mathbf{X}_T$ for the test samples. The pose parameter maps can be computed analogously to (1)-(2).

2.3 Topographic Mapping with Isomap

Isomap (from *iso*metric feature *map*ping) [1], is essentially a Multidimensional Scaling (MDS) [10] operating on the pairwise *geodesic* distance matrix $\mathbf{D}_G$ built from the input data samples $\vec{\mathbf{x}}_i$. A neighborhood graph $\boldsymbol{G}$ is determined, such that $\boldsymbol{G}$ would contain edge $x_i x_j$ *iff* x_j is one of the k nearest neighbors of x_i. Then the shortest paths in $\boldsymbol{G}$ are computed for all pairs of data points, e.g.

$$d_G(\mathbf{x}_i,\mathbf{x}_j)=\min\{d_G(\mathbf{x}_i,\mathbf{x}_j),d_G(\mathbf{x}_i,\mathbf{x}_k)+d_G(\mathbf{x}_k,\mathbf{x}_j)\} \tag{3}$$

and the shortest paths between any two samples are represented in matrix $\mathbf{D}_G=\{d_G(\mathbf{x}_i,\mathbf{x}_j)\}$ .Then classical MDS is applied to $\mathbf{D}_G$ to find a lower-dimensional embedding of the data that best preserves the manifold's intrinsic geometry. First, the inner-product matrix $\mathbf{B}=\mathbf{HAH}$ is calculated, where $\mathbf{A}=\{-0.5d_G^2(\mathbf{x}_i,\mathbf{x}_j)\}$, and $\mathbf{H}=\mathbf{I}-N^{-1}\mathbf{11}^t$ is the centering matrix. Next, the eigenvectors $\vec{\mathbf{v}}_i$, corresponding to the top positive l eigenvalues λ_i of $\mathbf{B}$ are found and the required l-dimensional embedding is given by the matrix $\mathbf{R}=(\sqrt{\lambda_1}\cdot\vec{\mathbf{v}}_1,...,\sqrt{\lambda_l}\cdot\vec{\mathbf{v}}_l)^t$, i.e. the ith column of $\mathbf{R}$ gives the coordinates of sample $\vec{\mathbf{x}}_i$ in the newly found l-dimensional subspace.

We can define a pose parameter map $\mathbf{F}$ relating view angles $\boldsymbol{\Theta}$ to their corresponding embedded training samples $\mathbf{R}$ in a similar way as in (1), $\mathbf{F}^t=\mathbf{V}_R\mathbf{W}_R^{-1}\mathbf{U}_R^t\boldsymbol{\Theta}^t$, where $\mathbf{V}_R\mathbf{W}_R\mathbf{U}_R^t$ is the SVD of $\mathbf{R}^t$. However, in order to find an equation similar to (2) for the test samples, first we must find a way how to embed the test samples in the subspace created by the model, i.e. to calculate $\mathbf{R}_T$ from $\mathbf{X}_T$. Such a technique exists for MDS (see [8] for a proof) and can be applied to Isomap in a similar way:

$$\mathbf{R}_T=\frac{1}{2}(\mathbf{R}^{\#})^t\mathbf{D}_T=\frac{1}{2}((\mathbf{R}^t\mathbf{R})^{-1}\mathbf{R}^t)^t\mathbf{D}_T \tag{4}$$

where $\mathbf{D}_T=(\mathbf{d}_1,...,\mathbf{d}_M)$ is given by

$$\mathbf{d}_i=diag(\mathbf{R}^t\mathbf{R})-\mathbf{d}_{i,N} \qquad \mathbf{d}_{i,N}=(d_G^2(\vec{\mathbf{x}}_{T(i)},\vec{\mathbf{x}}_1),...,d_G^2(\vec{\mathbf{x}}_{T(i)},\vec{\mathbf{x}}_N))^t. \tag{5}$$

In (5), $\mathbf{d}_{i,N}$ is a column vector of the squared geodesic distances between the ith test sample (i: 1…M) and each of the training samples. Now the unknown view angles $\boldsymbol{\Theta}_T$ of the test samples in $\mathbf{X}_T$ can be calculated as $\boldsymbol{\Theta}_T=\mathbf{FR}_T$.

2.4 Topographic Mapping with Neuro/Isoscale

An alternative well-known topographic method is Sammon mapping [9], which minimizes a 'stress' measure which explicitly embodies the topographic constraint:

$$E=\sum_{i}^{N}\sum_{j>i}^{N}(d_{ij}^{*}-d_{ij})^2 \tag{6}$$

where $d_{ij}^{*} = \left\| \mathbf{x}_i - \mathbf{x}_j \right\|$ are the inter-point distances in the original data space, and $d_{ij} = \left\| \mathbf{y}_i - \mathbf{y}_j \right\|$ the inter-point distances in the lower-dimensional map. One important disadvantage of this mapping is that test data cannot be added to the map without an expensive re-optimization of the stress on the whole dataset. To alleviate this problem, Neuroscale has been proposed in [10], which defines a nonlinear map between the data space and the output map using a radial basis functions (RBF) net. An efficient algorithm called "shadow targets" has been proposed [10] to train the model. Hypothetical target values t_i

$$t_i = y_i + 2\sum_{j \neq i} \left(\frac{d_{ij} - d_{ij}^{*}}{d_{ij}} \right) (\mathbf{y}_i - \mathbf{y}_j) \tag{7}$$

are defined, where the second term is the partial derivative $-\partial E / \partial \mathbf{y}_i$ of the stress (6). For a fixed set of targets **T**, the weights of the RBF net can be calculated as $\mathbf{W} = \mathbf{\Phi}^{\#}\mathbf{T}$, where $\mathbf{\Phi}^{\#}$ is the pseudo-inverse of the basis functions activations matrix $\mathbf{\Phi}_{ij} = \phi_j(\mathbf{x}_i)$. Since the estimated targets are not fixed, a model trust region based iterative algorithm is used to re-estimate the targets and update the weights at each step. The mapping obtained in this way is still unsupervised in the sense that there is no specific target information for the map coordinates, only a *relative* measure of target separation between each pair of points. However, if available, supervised information can be used by this method to influence the topology induced by the training data by replacing d_{ij}^{*} in (6) with $\delta_{ij} = (1-\alpha) d_{ij}^{*} + \alpha s_{ij}$, where s_{ij} is the *absolute* target separation, and α ($0 \leq \alpha \leq 1$) controls the degree of supervisory information in the mapping. Neuroscale has also been used mainly for visualization, with Euclidean metric to measure the distances between data points.

However, for the task at hand we propose to modify Neuroscale to use the *geodesic* distance, expecting to achieve an effect similar to that of Isomap extending nonlinearly MDS (which in Euclidean metric is equivalent to PCA). In this case, the geodesic distances in the data space can be calculated as in (3) above. To discriminate between the cases when the Euclidean, and when the geodesic distance is used, we will call the latter method *IsoScale*. Thus, *IsoScale* can be considered a hybrid between *Iso*map and Neuro*scale*, combining those aspects of each, which we expect to be useful for our task, namely: (a) the geodesic distance would be better suited to recover the intrinsic geometric structure of the nonlinearly embedded head pose manifolds than the Euclidean, resulting in a lower-dimensional embedding (we need a very low-dimensional embedding if we are to be successful with our semi-supervised approach, which has to use as few training samples as possible); (b) at the same time IsoScale has the following important advantages over Isomap: (1) since the RBF net additionally learns a nonlinear map between the image data space and the output map, there is no need to compute distances to *all* training samples at test time as in Isomap, which would be prohibitive for real-time implementations; (2) IsoScale permits efficient sequential implementation, so practically there is no limit on the training set size used to find (in unsupervised way!) the low-dimensional embedding, while Isomap has to store the whole distance matrix in memory, i.e. cannot handle more than a few thousand face images.

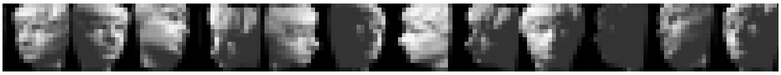


Fig. 1. Examples of faces with different pan, tilt, and illumination values from the dataset used in the first experiment

3 Experiments

Here we report some experimental results obtained by implementing the semi-supervised approach described in the previous two sections. First we start with a (relatively difficult) toy problem. We test the plausibility of the proposed approach on the same dataset which has been used in [1] for visualization (note that *visualization* is a different problem, and a much easier one than pose *estimation*), which is downloadable from http://isomap.stanford.edu/datasets.html. This dataset contains about 700 face images (see Fig.1 for an example) with known pan, tilt and illumination direction values: pan changes in the range $(-75^{\circ},...,75^{\circ})$, tilt in the range $(-10^{\circ},...,10^{\circ})$, and illumination direction in the range (105 ,..., 255). We adopt the following experimental procedure: we use each of the methods in section 2 in turn to find in an *unsupervised* way a low-dimensional embedding using 90% of all data (randomly selected), and then learn a linear pose-parameter map using respectively only 10 or 60 randomly selected samples. Then we test the accuracy of the pose estimation on the remaining 10% of the data which has been set aside as test samples. This procedure is repeated several times and the average of the obtained results are reported in Fig. 2 for the case of 3-dimensional embedding.

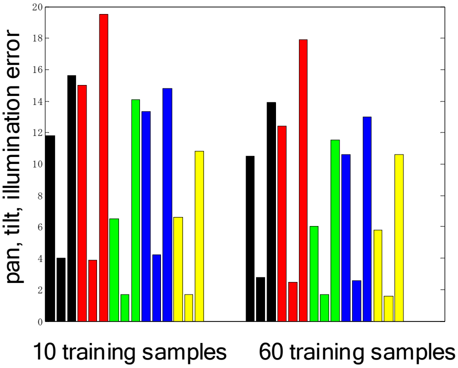


Fig. 2. Average absolute errors obtained by each of the following methods: PCA (black), LPP (red), Isomap (green), Neuroscale (blue) and Isoscale (yellow). For each set of bars with the same color, the first represents pan error, the second tilt error and the third illumination direction error. The left part is obtained for semi-supervised learning using only 10 training samples, and the right one with 60 training samples

As can be seen from Fig. 2, the geodesic distance-based methods, Isomap and IsoScale achieve the best performance, with only 10 training samples being sufficient to obtain satisfactory error rates. Note that when only 10 training samples are used, IsoScale has error rate which is about 2 times smaller than Neuroscale.

Next we describe one typical example of a real-world application, where the proposed semi-supervised approach is particularly advantageous. We have developed a facial gesture interface, using the pose estimation method proposed in this paper, for automatic wheelchair navigation for people with paralyzed limbs suffering from cerebral palsy. Such people are unable to use their hands to control their wheelchairs, but can use some simple head movement based gestures instead. A stereo camera is mounted on the wheelchair, which monitors the head movements of the subjects. In this way, it is easy to obtain a lot of (even hours of) raw image data from the camera, but obtaining view-labeled supervised data is difficult, as magnetic sensors providing the supervised information have to be attached to the head of the subjects, and this can be done only for a limited period of time (ideally only once). In Figs. 3-6 we illustrate how IsoScale can be used in a semi-supervised mode to learn a low-dimensional embedding from about 50000 faces video stream taken in different environments (indoors and outdoors, under different illumination conditions, etc.), followed by learning a linear map to pose-parameter space using only a small view-labeled subset of the data. Different combinations of pan and tilt angles are used to encode a facial gesture interface able to reliably detect signals for controlling the motion of the wheelchair, e.g. “left”, “right”, “start”, “stop”, “back”, left/right rotations, menu for choosing speed, etc. By careful observation of the range and type of head motions available to people with cerebral palsy, we designed a state transition graph diagram which is used to generate a control decision based on the continuously estimated head pose information. When the wheelchair is in a certain state, a transition to a different state is possible only if a suitable head pose is detected. The set and range of valid head poses which can be accepted in a certain state are chosen to minimize the possibility of wrong interpretation.

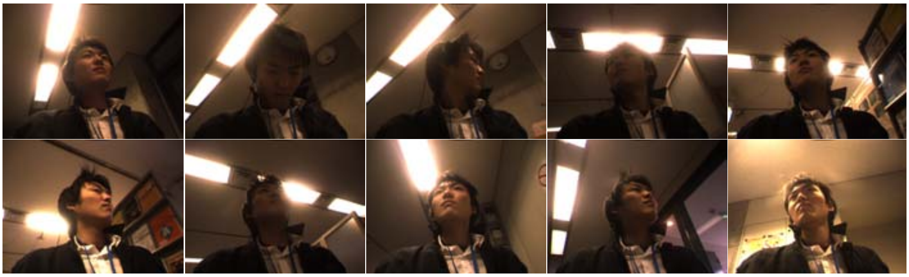


Fig. 3. Several frames from a video sequence obtained from the camera mounted on the wheelchair (*ordinary office environment*)

4 Conclusion

In this paper we have proposed a new general approach to the problem of head pose estimation, based on semi-supervised low-dimensional topographic feature mapping, which is expected to find application in situations in which the larger part of the available data is not accompanied by supervised information, or even in extreme cases when we have only a few labeled samples. As the proposed method relies on finding a topography-preserving low-dimensional embedding of the highly-dimensional input data in an unsupervised way, we have shown how several recently

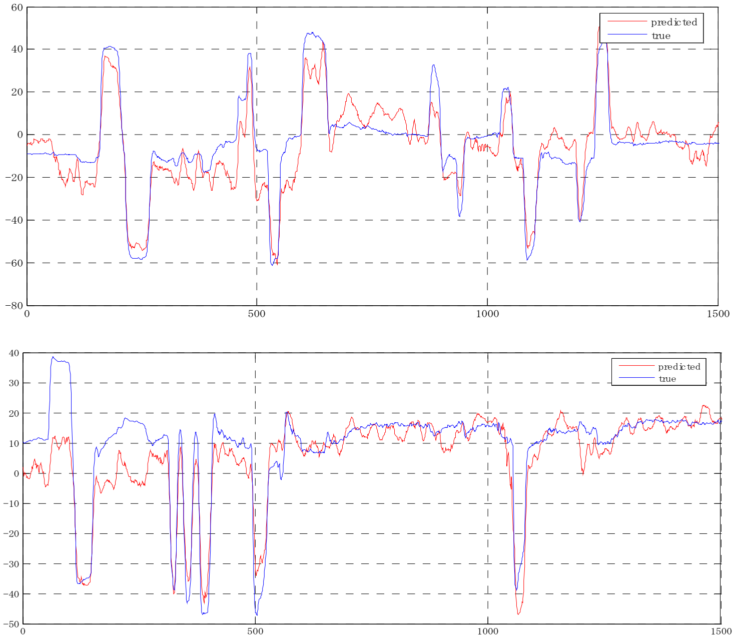


Fig. 4. Trajectories of true (red) and predicted (blue) views obtained by IsoScale for the facial gesture test sequence in Fig. 3, with pan (top) and tilt (bottom) measured in degrees

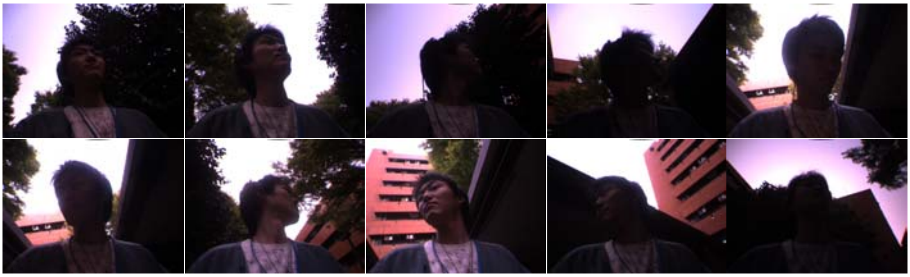


Fig. 5. Several frames from a video sequence obtained from the camera mounted on the wheelchair (*outdoor environment*)

proposed nonlinear manifold learning methods can be applied in this context, while additionally we have presented a new algorithm, IsoScale, which is especially suitable for the task at hand as discussed in detail in section 2-4 and experimentally demonstrated in section 3. A further advantage of the proposed method compared e.g. to feature tracking based pose estimation methods is that this method does not rely on detecting/tracking geometric features like eyes, nose, etc., which can be difficult to detect/track for example for near-profile poses or fast gesture movements, therefore, the proposed method is able to achieve reliable pose estimation for much larger view range. Furthermore, as no domain knowledge is used, the method would be useful for any 3D object pose estimation, not limited to faces.

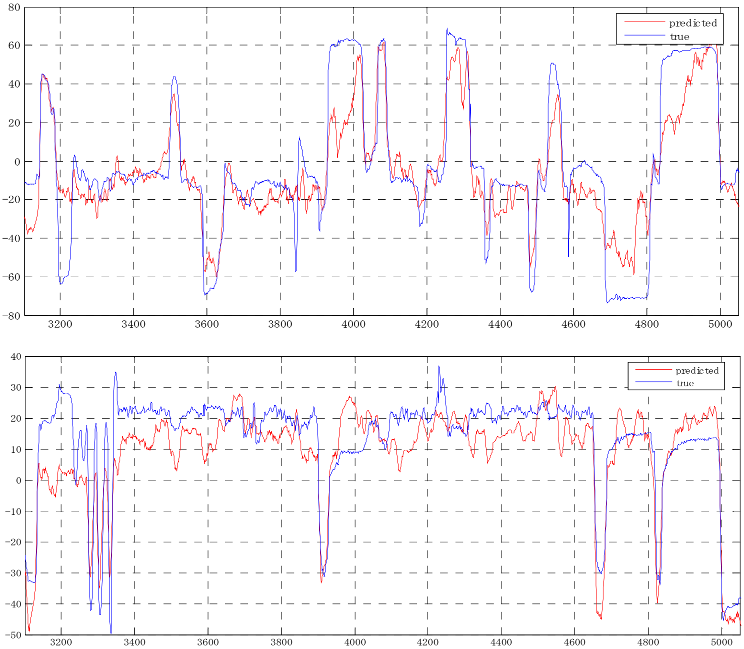


Fig. 6. Trajectories of true (red) and predicted (blue) views obtained by IsoScale for the facial gesture test sequence shown in Fig. 5, with pan (top) and tilt (bottom) measured in degrees

References

1. J. B. Tenenbaum, V. de Silva, and J. C. Langford, “A global geometric framework for nonlinear dimensionality reduction”, *Science* 290, (2000), 2319-2323.
2. M. Belkin, and P. Niyogi, “Laplacian Eigenmaps and Spectral Techniques for Embedding and Clustering”, *Adv. NIPS* 15, Vancouver, Canada, 2001
3. J. Heinzmann and A. Zelinsky, "3D facial pose and gaze point estimation using a robust real-time tracking paradigm", *in Proc. Int. Workshop on Automatic Face and Gesture Recognition*, Nara, (1998), 142-147.
4. M. Bichsel and A. Pentland, “Automatic interpretation of human head movements”, *Technical Report No. 186*, MIT Media Laboratory, Vision and Modeling Group, (1993)
5. S. J. McKenna and S. Gong, “Real-time face pose estimation”, *Real-Time Imaging* 4, (1998), 333-347.
6. K. Okada, “Analysis, Synthesis and Recognition of Human Faces with Pose Variations”, *Ph.D. thesis*, USC, (2001)
7. X. He, S. Yan, Y. Hu and H. J. Zhang, “Learning a Locality Preserving Subspace for Visual Recognition”, *in Proc. 9th ICCV*,(2003)
8. J. Gower, “Adding a point to vector diagrams in multivariate analysis”, *Biometrica*, 55, (1968), 582-585.
9. J. W. Sammon, “A nonlinear mapping for data structure analysis”, *IEEE Transactions on Computers*, C-18(5), (1969), 401-409.
10. M M. E. Tipping and D. Lowe, “Shadow targets: A novel algorithm for topographic projection by radial basis function network”, In *Proc. Int. Conf. Artificial Neural Networks*, Vol. 440, (1997), 7-12.

Accurate and Efficient Gesture Spotting via Pruning and Subgesture Reasoning

Jonathan Alon, Vassilis Athitsos, and Stan Sclaroff*

Computer Science Department
Boston University
Boston, MA 02215, USA

Abstract. Gesture spotting is the challenging task of locating the start and end frames of the video stream that correspond to a gesture of interest, while at the same time rejecting non-gesture motion patterns. This paper proposes a new gesture spotting and recognition algorithm that is based on the continuous dynamic programming (CDP) algorithm, and runs in real-time. To make gesture spotting efficient a pruning method is proposed that allows the system to evaluate a relatively small number of hypotheses compared to CDP. Pruning is implemented by a set of model-dependent classifiers, that are learned from training examples. To make gesture spotting more accurate a subgesture reasoning process is proposed that models the fact that some gesture models can falsely match parts of other longer gestures. In our experiments, the proposed method with pruning and subgesture modeling is an order of magnitude faster and 18% more accurate compared to the original CDP algorithm.

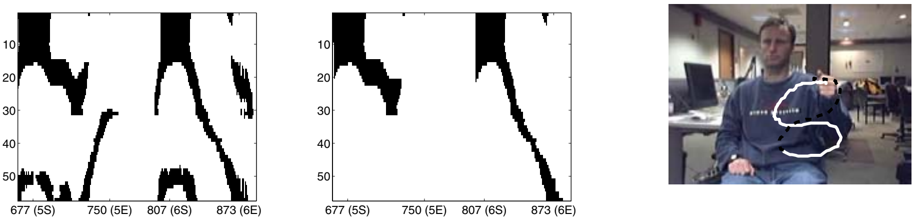
1 Introduction

Many vision-based gesture recognition systems assume that the input gestures are isolated or segmented, that is, the gestures start and end in some rest state. This assumption makes the recognition task easier, but at the same time it limits the naturalness of the interaction between the user and the system, and therefore negatively affects the user's experience. In more natural settings the gestures of interest are embedded in a continuous stream of motion, and their occurrence has to be detected as part of recognition. This is precisely the goal of *gesture spotting*: to locate the start point and end point of a gesture pattern, and to classify the gesture as belonging to one of predetermined gesture classes. Common applications of gesture spotting include command spotting for controlling robots [1], televisions [2], computer applications [3], and video games [4, 5].

Arguably, the most principled methods for spotting dynamic gestures are based on dynamic programming (DP) [3, 6, 7]. Finding the optimal matching between a gesture model and an input sequence using brute-force search would involve evaluating an exponential number of possible alignments. The key advantage of DP is that it can find the best alignment in polynomial time. This is

* This research was supported in part through U.S. grants ONR N00014-03-1-0108, NSF IIS-0308213 and NSF EIA-0202067

N. Sebe, M.S. Lew, and T.S. Huang (Eds.): HCI/ICCV 2005, LNCS 3766, pp. 189–198, 2005.



(a) (b) (c)

Fig. 1. Pruning (a,b): example dynamic programming table for matching input stream (x axis) to a model gesture for the digit "6" (y axis). Likely observations are represented by black cells in the table (a). The cells remaining after pruning (b). In this example 87% of the cells (shown in white) were pruned. Subgesture reasoning (c): example false detection of the digit "5", which is similar to a subgesture of the digit "8"

achieved by reducing the problem of finding the best alignment to many subproblems that involve matching a part of the model to parts of the video sequence. The main novelty of our method is a pruning technique that eliminates the need to solve many of these subproblems. As a result, gesture spotting and recognition become both faster and more accurate: faster because a smaller number of hypotheses need to be evaluated; more accurate because many of the hypotheses that could have led to false matches are eliminated at an early stage. In Figure 1(b) the number of hypotheses evaluated by the proposed algorithm is proportional to the number of black pixels, and the number of hypotheses that are evaluated by a standard DP algorithm but are pruned by the proposed algorithm is proportional to the number of white pixels.

A second contribution of this paper is a novel reasoning process for deciding among multiple candidate models that match well with the current portion of the input sequence. Comparing the matching scores and using class specific thresholds, as is typically done [3, 6], is often insufficient for picking out the right model. We propose identifying, for each gesture class, the set of "subgesture" classes, i.e., the set of gesture models that are similar to subgestures of that class. While a gesture is being performed, it is natural for these subgesture classes to cause false alarms. For example, in the online digit recognition example depicted in Figure 1(c), the digit "5" may be falsely detected instead of the digit "8", because "5" is similar to a subgesture of the digit "8". The proposed subgesture reasoning can reliably recognize and avoid the bulk of those false alarms.

2 Related Work

Gesture spotting is a special case of the more general pattern spotting problem, where the goal is to find the boundaries (start points and endpoints) of patterns of interest in a long input signal. Pattern spotting has been applied to different types of input including text, speech [8], and image sequences [6].

There are two basic approaches to detection of candidate gesture boundaries: the direct approach, which precedes recognition of the gesture class, and the indirect approach, where spotting is intertwined with recognition. Methods that belong to the direct approach first compute low-level motion parameters such as velocity, acceleration, and trajectory curvature [5] or mid-level motion parameters such as human body activity [9], and then look for abrupt changes (e.g., zero-crossings) in those parameters to find candidate gesture boundaries.

In the indirect approach, the gesture boundaries are detected using the recognition scores. Most indirect methods [3, 7] are based on extensions of Dynamic Programming (DP) algorithms for isolated gestures (e.g., HMMs [10] and DTW [11]). In those methods, the gesture endpoint is detected when the recognition likelihood rises above some fixed or adaptive [3] threshold, and the gesture start point can be computed, if needed, by backtracking the optimal DP path. One such extension, continuous dynamic programming (CDP), was proposed by Oka [7]. In CDP, an input sequence is matched with a gesture model frame-by-frame. To detect a candidate gesture, the cumulative distance between them is compared to a threshold.

After a provisional set of candidates has been detected, a set of rules is applied to select the best candidate, and to identify the input subsequence with the gesture class of that candidate. Different sets of rules have been proposed in the literature: peak finding rules [6], spotting rules [12], and the user interaction model [13].

One problem that occurs in practice but is often overlooked is the false detection of gestures that are similar to parts of other longer gestures. To address this problem [3] proposed two approaches. One is limiting the response time by introducing a maximum length of the nongesture pattern that is longer than the largest gesture. Another, is taking advantage of heuristic information to catch one's completion intentions, such as moving the hand out of the camera range or freezing the hand for a while. The first approach requires a parameter setting, and the second approach limits the naturalness of the user interaction. We propose instead to explicitly model the subgesture relationship between gestures. This is a more principled way to address the problem of nested gestures, which does not require any parameter setting or heuristics.

3 Gesture Spotting

In this section we will introduce the continuous dynamic programming (CDP) algorithm for gesture spotting. We will then present our proposed pruning and subgesture reasoning methods that result in an order of magnitude speedup and 18% increase in recognition accuracy.

3.1 Continuous Dynamic Programming (CDP)

Let $M = (M_1, \ldots, M_m)$ be a model gesture, in which each M_i is a feature vector extracted from model frame i. Similarly, let $Q = (Q_1, \ldots, Q_j, \ldots)$ be a continuous stream of feature vectors, in which each Q_j is a feature vector extracted from input frame j. We assume that a cost measure $d(i, j) \equiv d(M_i, Q_j)$ between two feature vectors M_i and Q_j is given. CDP computes the optimal path and the minimum cumulative distance $D(i, j)$ between the model subsequence $M_{1:i}$ and the input subsequence $Q_{j':j}, j' \leq j$. Several ways have been proposed in the literature to recursively define the cumulative distance. The most popular definition is:

$$D(i, j) = \min\{D(i-1, j), D(i-1, j-1), D(i, j-1)\} + d(i, j). \tag{1}$$

For the algorithm to function correctly the cumulative distance has to be initialized properly. This is achieved by introducing a dummy gesture model frame 0 that matches all input frames perfectly, that is, $D^{M^g}(0, j) = 0$ for all j. Initializing this way enables the algorithm to trigger a new warping path at every input frame.

In the online version of CDP the local distance $d(i, j)$ and the cumulative distance $D(i, j)$ need not be stored as matrices in memory. It suffices to store for each model (assuming backtracking is not required) two column vectors: the current column col_j corresponding to input frame j, and the previous column col_{j-1} corresponding to input frame $j - 1$. Every vector element consists of the cumulative distance D of the corresponding cell, and possibly other useful data such as the warping path length.

3.2 CDP with Pruning (CDPP)

The CDP algorithm evaluates Eq. 1 for every possible i and j. A key observation is that for many combinations of i and j, either the feature-based distance $d(i, j)$ or the cumulative distance $D(i, j)$ can be sufficiently large to rule out all alignments going through cell (i, j). Our main contribution is that we generalize this pruning strategy by introducing a set of binary classifiers that are learned from training data offline. Those classifiers are then used to prune certain alignment hypotheses during online spotting. In our experiments, this pruning results in an order of magnitude speedup.

The proposed pruning algorithm is depicted in Algorithm 1. The input to the algorithm is input frame j, input feature vector Q_j, a set of model dependent classifiers C_i, and the previous sparse column vector. The output is the current sparse column vector.

The concept of model dependent classifiers C_i that are learned from training data offline, and are used for pruning during online spotting is novel. Different types of classifiers can be used including: subsequence classifiers, which prune based on the cumulative distance (or likelihood); transition classifiers, which prune based on the transition probability between two model frames (or states); and single observation classifiers, which prune based on the likelihood of the current observation. In our experiments we use single observation classifiers:

$$C_i(Q_j) = \begin{cases} +1 \text{ if } d(i, j) \leq \tau(i) \\ -1 \text{ if } d(i, j) > \tau(i) \end{cases}, \tag{2}$$

where each $\tau(i)$ defines a decision stump classifier for model frame i, and is estimated as follows: the model is aligned, using DTW, with all the training

```
input  : input frame $j$, input feature vector $Q_j$, classifiers $C_i$, and
         previous sparse column vector $< ind_{j-1}, list_{j-1} >$.
output: current sparse column vector $< ind_j, list_j >$.
$i = 1$;
$ptr = ind_{j-1}(0)$;
while $i \leq m$ do
    if $C_i(Q_j) == +1$ then
        $nl$ = new element; // $nl$ will be appended to end of $list_j$
        $nl.D = \min\{ind_j(i-1).D, ind_{j-1}(i-1).D, ind_{j-1}(i).D\} + d(i,j)$;
        $nl.i = i$;
        append($list_j$, $nl$);
        $ind_j = \&list_j(i)$; // & is the address-of operator, as in C
        $i = i + 1$;
    else
        //previous column empty
        if $isempty\ (\ list_{j-1}\ )$ then
            break;
        if $ind_{j-1}(i) == NULL$ then
            while $ptr \rightarrow next\ != NULL\ and\ ptr \rightarrow next \rightarrow i \leq i$ do
                ptr = ptr→next;
            end
            //reached the end of previous column
            if $ptr \rightarrow next == NULL$ then
                break;
            $i$ =ptr→next→i;
        else
            $i = i + 1$;
        end
    end
end
```

Algorithm 1. The CDPP algorithm

examples of gestures from the same class. The distances between observation i and all the observations (in the training examples) which match observation i are saved, and the threshold $\tau(i)$ is set to the maximum distance among those distances. Setting the thresholds as specified guarantees that all positive training examples when embedded in longer test sequences will be detected by the spotting algorithm.

In order to maximize efficiency we chose a sparse vector representation that enables fast individual element access, while keeping the number of operations proportional to the sparseness of the DP table (the number of black pixels in Fig. 1(b)). The sparse vector is represented by a pair $< ind, list >$, where ind is a vector of pointers of size m (the model sequence length), and is used to reference elements of the second variable $list$. The variable $list$ is a singly linked list, where each list element is a pair that includes the cumulative distance $D(i, j)$ and the index i of the corresponding model frame. The length of $list$ corresponds to the number of black pixels in the corresponding column in Fig. 1(b).

We note that in the original CDP algorithm there is no pruning, only lines 5-10 are executed inside the while loop, and i is incremented by 1. In contrast, in CDPP whenever the classifier outputs -1 and a hypothesis is pruned then i is incremented by an offset, such that the next visited cell in the current column will have at least one *active* neighbor from the previous column.

Algorithm 1 is invoked separately for every gesture model M^g. For illustration purposes we show it for a single model. After the algorithm has been invoked for the current input frame j and for all the models, the end-point detection algorithm of Sec. 3.3 is invoked.

3.3 Gesture End Point Detection and Gesture Recognition

The proposed gesture endpoint detection and gesture recognition algorithm consists of two steps: the first step updates the current list of candidate gesture models. The second step uses a set of rules to decide if a gesture was spotted, i.e., if one of the candidate models truly corresponds to a gesture performed by the user. The end point detection algorithm is invoked once for each input frame j. In order to describe the algorithm we first need the following definitions:

- **Complete path:** a legal warping path $W(M_{1:m}, Q_{j':j})$ matching an input subsequence $Q_{j':j}$ ending at frame j with the complete model $M_{1:m}$.
- **Partial path:** a legal warping path $W(M_{1:i}, Q_{j':j})$ that matches an input subsequence $Q_{j':j}$ ending at the current frame j with a model prefix $M_{1:i}$.
- **Active path:** any partial path that has not been pruned by CDPP.
- **Active model:** a model g that has a complete path ending in frame j.
- **Firing model:** an active model g with a cost below the detection acceptance threshold.
- **Subgesture relationship:** a gesture g_1 is a subgesture of gesture g_2 if it is properly contained in g_2. In this case, g_2 is a supergesture of g_1.

At the beginning of the spotting algorithm the list of candidates is empty. Then, at every input frame j, after all the CDP costs have been updated, the best firing model (if such a model exists) is considered for inclusion in the list of candidates, and existing candidates are considered for removal from the list. The best firing model will be different depending on whether or not subgesture reasoning is carried out, as described below. For every new candidate gesture we record its class, the frame at which it has been detected (or the end frame), the corresponding start frame (which can be computed by backtracking the optimal warping path), and the optimal matching cost. The algorithm for updating the list of candidates is described below. The input to this algorithm is the current list of candidates, the state of the DP tables at the current frame (the active model hypotheses and their corresponding scores), and the lists of supergestures. The output is an updated list of candidates. Steps that involve subgesture reasoning are used in the algorithm CDPP with subgesture reasoning (CDPPS) only, and are marked appropriately.

1. Find all firing models and continue with following steps if the list of firing models is nonempty.
2. CDPPS only: conduct subgesture competitions between all pairs of firing models. If a firing model g_1 is a supergesture of another firing gesture model g_2 then remove g_2 from the list of firing models. After all pairwise competitions the list of firing models will not contain any member which is a supergesture of another member.
3. Find the best firing model, i.e., the model with the best score.
4. For all candidates g_i perform the following four tests:
   (a) CDPPS only: if the best firing model is a supergesture of any candidate g_i then mark candidate g_i for deletion.
   (b) CDPPS only: if the best firing model is a subgesture of any candidate g_i then flag the best model to not be included in the list of candidates.
   (c) If the score of the best firing model is better than the score of a candidate g_i and the start frame of the best firing model occurred after the end frame of the candidate g_i (i.e., the best firing model and candidate g_i are non-overlapping, then mark candidate g_i for deletion.
   (d) If the score of the best firing model is worse than the score of a candidate g_i and the start frame of the best firing model occurred after the end frame of the candidate g_i (i.e., the best firing model and candidate g_i are non-overlapping, then flag the best firing model to not be included in the list of candidates.
5. Remove all candidates g_i that have been marked for deletion.
6. Add the best firing model to the list of candidates if it has not been flagged to not be included in that list.

After the list of candidates has been updated then if the list of candidates is nonempty then a candidate may be "spotted", i.e., recognized as a gesture performed by the user if:

1. CDPPS only: all of its active supergesture models started after the candidate's end frame j^*. This includes the trivial case, where the candidate has an empty supergesture list, in which case it is immediately detected.
2. all current active paths started after the candidate's detected end frame j^*.
3. a specified number of frames have elapsed since the candidate was detected. This detection rule is optional and should be used when the system demands a hard real-time constraint. This rule was not used in our experiments.

Once a candidate has been detected the list of candidates is reset (emptied), and all active path hypotheses that started before the detected candidate's end frame are reset, and the entire procedure is repeated. To the best of our knowledge the idea of explicit reasoning about the subgesture relationship between gestures, as specified in steps 2, 4a, and 4b of the candidates update procedure and step 1 of the end-point detection algorithm, is novel.

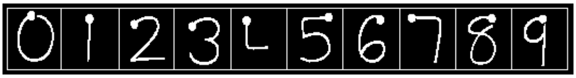


Fig. 2. Palm's Graffiti digits [14]

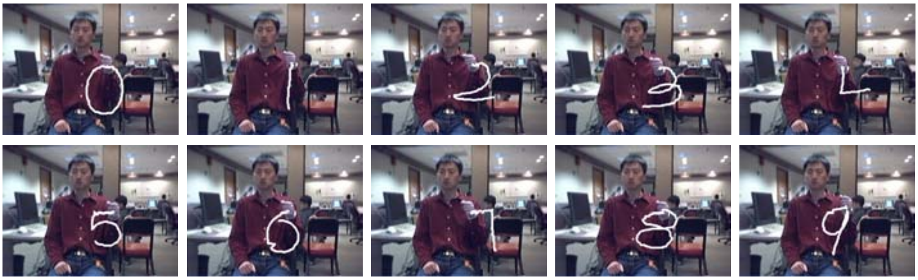


Fig. 3. Example model digits extracted using a colored glove

4 Experimental Evaluation

We implemented Continuous Dynamic Programming (CDP) [7] with a typical set of gesture spotting rules. In particular, we used a global acceptance threshold for detecting candidate gestures, and we used the gesture candidate overlap reasoning described in Sec. 3.3. This is the baseline algorithm, to which we compare our proposed algorithms. The proposed CDP with pruning algorithm (CDPP), is implemented as described in Sec. 3.2, with the same gesture spotting rules used in the baseline algorithm. The second proposed algorithm, CDPP with subgesture reasoning (CDPPS), includes the additional steps marked in Sec. 3.3.

We compare the baseline algorithm and the proposed algorithms in terms of efficiency and accuracy. Algorithm efficiency is measured by CPU time. Accuracy is evaluated by counting for every test sequence the number of correct detections and the number of false alarms. A correct detection corresponds to a gesture that has been detected and correctly classified. A gesture is considered to have been detected if its estimated end frame is within a specified temporal tolerance of 15 frames from the ground truth end frame. A false alarm is a gesture that either has been detected within tolerance but incorrectly classified, or its end frame is more than 15 frames away from the correct end frame of that gesture.

To evaluate our algorithm we have collected video clips of two users gesturing ten digits 0-9 in sequence. The video clips were captured with a Logitech 3000 Pro camera using an image resolution of 240×320, at a frame rate of 30 Hz. For each user we collected two types of sequences depending on what the user wore: three colored glove sequences and three long sleeves sequences; (a total of six sequences for each user). The model digit exemplars (Fig. 3) were extracted from the colored glove sequences, and were used for spotting the gestures in the long video streams. The range of the input sequence lengths is $[1149, 1699]$ frames. The range of the digit sequence lengths is $[31, 90]$ frames. The range of the (in between digits) non-gestures sequence lengths is $[45, 83]$ frames.

For the glove sequences the hand was detected and tracked using the glove color distribution. For the other sequences the hand was detected and tracked using color and motion. A hand mask was computed using skin and non-skin color distributions [15], and was applied to an error residual image obtained by a block-based optical flow method [16]. For every frame we computed the 2D hand centroid locations and the angle between two consecutive hand locations. The feature vectors (M_i and Q_j) used to compute the local distance $d(i, j)$ are the 2D positions only. The classifier used for pruning was combination of two classifiers: one based on the 2D positions and the other based on the angle feature. Those classifiers were trained on the model digits in the offline step. To avoid overpruning we added 20 pixels to the thresholds of all position classifiers and an angle of 25 degrees to all angle classifiers.

For the end-point detection algorithm we specified the following supergesture lists that capture the subgesture relationship between digits:

Subgesture	Supergestures
“0”	{“9”}
“1”	{“4”,“7”,“9”}
“4”	{“2”,“5”,“6”,“8”,“9”}
“5”	{“8”}
“7”	{“2”,“3”,“9”}

The experimental results are summarized in Table 1. For the baseline CDP algorithm we obtained 47 correct detections and 13 false matches. For the proposed CDPP algorithm without subgesture reasoning we obtained 51 correct detections and 9 false matches, and finally for the proposed CDPP algorithm with subgesture reasoning we obtained 58 correct detections and 2 false matches. The two false matches resulted from two examples of the digit 0 that were confused as 6. Compared to CDPP without subgesture reasoning, the proposed CDPP with subgesture reasoning corrected a single instance of the digit “3” initially confused as its corresponding subdigit “7”, four instances of the digit “8” initially confused as its corresponding subdigit “5”, and two instances of the digit “9” initially confused as its corresponding subdigit “1”.

Table 1. Comparison of gesture spotting accuracy results between the baseline and the proposed gesture spotting algorithms. The accuracy results are given in terms of correct detection rates and false matches. The total number of gestures is 60

Method	CDP	CDPP	CDPPS
Detection Rate	78.3%	85.0%	96.7%
False Matches	13	9	2

In our experiments CDPP executed 14 times faster compared to CDP in terms of CPU time, assuming feature extraction. The overall vision-based recognition system runs comfortably in real-time.

5 Conclusion and Future Work

This paper presented a novel gesture spotting algorithm. In our experiments, this novel algorithm is an order of magnitude faster and 18% more accurate compared to continuous dynamic programming. Our current work explores other classifiers that can be used for pruning. In order to further improve our system's accuracy, we plan to incorporate a module that can make use of the DP alignment information to verify that the candidate gesture that has been detected and recognized indeed belongs to the estimated class. This is commonly known as verification in word spotting for speech [8]. Finally, rather than specifying the subgesture relationships manually we plan to learn them from training data.

References

1. Triesch, J., von der Malsburg, C.: A gesture interface for human-robot-interaction. In: Automatic Face and Gesture Recognition. (1998) 546–551
2. Freeman, W., Weissman, C.: Television control by hand gestures. Technical Report 1994-024, MERL (1994)
3. Lee, H., Kim, J.: An HMM-based threshold model approach for gesture recognition. PAMI **21** (1999) 961–973
4. Freeman, W., Roth, M.: Computer vision for computer games. In: Automatic Face and Gesture Recognition. (1996) 100–105
5. Kang, H., Lee, C., Jung, K.: Recognition-based gesture spotting in video games. Pattern Recognition Letters **25** (2004) 1701–1714
6. Morguet, P., Lang, M.: Spotting dynamic hand gestures in video image sequences using hidden Markov models. In: ICIP. (1998) 193–197
7. Oka, R.: Spotting method for classification of real world data. The Computer Journal **41** (1998) 559–565
8. Rose, R.: Word spotting from continuous speech utterances. In: Automatic Speech and Speaker Recognition - Advanced Topics. Kluwer (1996) 303–330
9. Kahol, K., Tripathi, P., Panchanathan, S.: Automated gesture segmentation from dance sequences. In: Automatic Face and Gesture Recognition. (2004) 883–888
10. Starner, T., Pentland, A.: Real-time american sign language recognition from video using hidden Markov models. In: SCV95. (1995) 265–270
11. Darrell, T., Pentland, A.: Space-time gestures. In: Proc. CVPR. (1993) 335–340
12. Yoon, H., Soh, J., Bae, Y., Yang, H.: Hand gesture recognition using combined features of location, angle and velocity. Pattern Recognition **34** (2001) 1491–1501
13. Zhu, Y., Xu, G., Kriegman, D.: A real-time approach to the spotting, representation, and recognition of hand gestures for human-computer interaction. CVIU **85** (2002) 189–208
14. Palm: Grafitti alphabet. (http://www.palmone.com/us/products/input/)
15. Jones, M., Rehg, J.: Statistical color models with application to skin detection. IJCV **46** (2002) 81–96
16. Yuan, Q., Sclaroff, S., Athistos, V.: Automatic 2D hand tracking in video sequences. In: WACV. (2005)

A Study of Detecting Social Interaction with Sensors in a Nursing Home Environment

Datong Chen, Jie Yang, and Howard Wactlar

School of Computer Science, Carnegie Mellon University
{datong,yang+,hdw}@cs.cmu.edu

Abstract. Social interaction plays an important role in our daily lives. It is one of the most important indicators of physical or mental diseases of aging patients. In this paper, we present a Wizard of Oz study on the feasibility of detecting social interaction with sensors in skilled nursing facilities. Our study explores statistical models that can be constructed to monitor and analyze social interactions among aging patients and nurses. We are also interested in identifying sensors that might be most useful in interaction detection; and determining how robustly the detection can be performed with noisy sensors. We simulate a wide range of plausible sensors using human labeling of audio and visual data. Based on these simulated sensors, we build statistical models for both individual sensors and combinations of multiple sensors using various machine learning methods. Comparison experiments are conducted to demonstrate the effectiveness and robustness of the sensors and statistical models for detecting interactions.

1 Introduction

The worldwide population over age 65 is expected to more than double from 357 million in 1990 to 761 million by 2025 [17]. At present, five percent of Americans over age 65 reside in nursing homes, with up to 50 percent of those over the age of 85 likely being placed in a nursing home at some point in their lives [13]. Among these nursing home residents, about 80% of them are believed to suffer from a psychiatric disorder, and 90% of patients with Alzheimer's disease experience behavioral complications leading to increased functional disability, medical morbidity, mortality and premature institutionalization [32]. In many nursing homes, physicians might visit their patients for only a short period of time once a week. Assessment of a patient's progress is based mainly on reports from staff (nurses and nurse assistants). The reports may be incomplete or even biased, due to schedule shift and the fact that each staff person has to take care of many patients. This may result in insufficient observation for monitoring either progressive change or brief and infrequent occurrences of aberrant activity for diagnosing some diseases. For example, dementia is very common among residents in nursing facilities. One obvious characteristic of dementia is a sustained decline in cognitive function and memory [24]. Studies indicate that the elderly with dementia may exhibit measurable agitated behaviors that increase confusion, delusion, and other psychiatric disturbances [27][31]. In the early stages of dementia, these agitated behaviors occur occasionally and only last a very short period of time and are frequently missed by caregivers. Therefore, a long-term observation and care become increasingly important for the elderly with dementia in nursing homes [9]. Although no widely accepted measure exists for dementia care environ-

N. Sebe, M.S. Lew, and T.S. Huang (Eds.): HCI/ICCV 2005, LNCS 3766, pp. 199–210, 2005.

ments [5], quantitative measures of daily activities of these patients can be very useful for dementia assessments.

In this research, we are interested in automatically extracting information from sensors for geriatric care applications within skilled-care facilities. We would develop a system that can automatically extract and classify important antecedents of psychosocial and health outcomes. One such indicator is the frequency, duration and type of social interactions of the patients with one another and their caregivers. Interaction with others is generally considered a positive and necessary part of our daily life. Changes in interaction patterns can reflect mental and physical status of a person. Naturally, the level of social interaction of a person depends on a wide range of factors, such as his/her health condition, his/her personal preference, and aptitude for social interaction. More important, most social interactions are observable. This makes it possible for detecting them using an automatic system.

This paper explores the feasibility of building such a sensor-based analyzer to detect social interactions in a nursing home environment. Automatic detection of social interaction in a nursing home requires a set of physical and algorithmic sensors. For example, we can use an RF (Radio Frequency) sensor to track the location of each patient or a speech detector (an algorithm) from audio signals. However, the development and deployment of physical and algorithmic sensors are not trivial tasks. Furthermore, attaching physical sensors on bodies of patients is not practical. To this end, we employ a Wizard of Oz approach that enables the effectiveness study of various combinations of sensors and multiple models from a wide range of plausibly simulated sensors.

One important goal of this study is to obtain critical knowledge of detecting social interactions without physically developing and deploying ineffective or unnecessary sensors. This study also aims to find out the intrinsic structure among social interaction events and answer the following questions: how to construct necessary sensors to analyze social interactions, and how far from building these sensors we are using current technologies. Due to the fact that human beings infer interaction activities mainly from audio and visual cues, we are able to simulate potential useful sensors using the knowledge of human experts from audio and visual channels. Therefore, this study can be performed on the basis of long-term digital audio and video recording of a nursing home environment. We first evaluate the importance of each individual sensor and then employ a variety of machine learning techniques to create statistical models to identify interactions between people using simulated sensor data.

2 Related Work

Social interaction consists of multiple individual human activities among multiple people. The work presented in this paper is closely related to location awareness and human activity analysis, which have been addressed by many researchers in different areas such as multimedia processing, pervasive computing, and computer vision.

Various wearable sensors have been developed in recent years to address person tracking and activity analysis in the ubiquitous computing area. Global Position System [24], active bat location system [15], and PlusOn time modulated ultra wideband technology [34] provide location measures from meter to centimeter precision. Some wearable sensors have been applied to health monitoring [23], group interaction analysis [16], and memory augmentation [29].

Elderly individuals are usually unwilling to adapt to even tiny changes in environment, including wearable sensors in their clothes. Some non-contact sensors are considered to be more practical in our task. Power line network [4] and Ogawa's monitoring system use switches and motion detectors to track human activities indoors. The data provided by switches and motion sensors are reliable and very easy to process. However, they cannot provide detailed information. For example, a motion sensor can only tell that there is a person in the monitored area but cannot tell the exact location.

A vision-based system can non-obtrusively provide location information. Many computer vision algorithms have been developed for not only recovering 3D locations of a person, but also providing detailed appearance information of the person and his/her activities. Koile et al [21] at MIT proposed a computer vision system to monitor the indoor location of a person and his/her moving trajectory. The Living Laboratory [20] was designed by Kidd, et. al. for monitoring the actions and activities of the elderly. Aggarwal, et. al. [1] has reviewed different methods for human motion tracking and recognition. Various schemes such as single or multiple camera schemes, and 2D and 3D approaches, have been broadly discussed in this review.

A large number of algorithmic sensors have been proposed to detect activities from audio and visual signals, including gait recognition [3], hand gesture analysis [11], facial expression understanding [10], sitting, standing and walking analysis [23], and speech detection [26]. Hudson, et. al examined the feasibility of using sensors and statistical models to estimate human interruptibility in an office environment [18]. These sensors are still mostly research challenges today, but can be potentially applicable in the future. Combinations of these sensors for analyzing human behaviors have been applied in some constrained environment, such as meeting rooms [36] and sports fields [19].

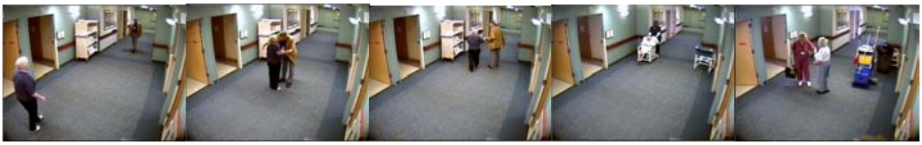


Fig. 1. Examples of interaction patterns in a nursing home

3 Data Collection and Preprocessing

Four cameras and four audio collectors were carefully placed in two rooms and a hallway of a nursing facility. Recording was performed from 9am to 5pm for 10 days. Overall, 320 hours were recorded at the nursing facility. Each video and its corresponding audio channels were digitalized and encoded into an MPEG-2 stream in real time and recorded onto hard disks through a PC. The video data was captured and finally recorded in 24-bit color with a resolution of 640x480 pixels at 30 frames per second. The audio data was recorded at 16-bit 44.1KHz. Figure 1 illustrates some examples of interaction patterns from the data. In this paper, only the hallway videos are manually ground-truthed and used for analysis.

Since we only focus on multi-person activities, we developed a preprocessing algorithm to segment audio/video streams into shots, and classify the shots into three classes: non-activity, individual activity, and multi-person activity using audio and video event detection techniques.

3.1 Video Event Detection

For the video channel, we use a background subtraction algorithm to detect frames that contain human activities. To speed up this detection process, only video from one camera in the network is used. The background of a frame is obtained by the adaptive background method [33]. We employ a threshold to extract pixels that have high differences between the current frame and its background. To remove noise, we group extracted pixels into regions and only keep those regions that contain more than 15 pixels. We consider the frame *f* to contain a visual interaction event $V_f=1$ if any of the following rules is satisfied; otherwise $V_f=0$:

1. There are two or more regions in the frame.
2. There is region that does not touch the bottom the frame, whose width to height ratio is more than 0.7.

We choose these thresholds to detect as many interactions as possible without inducing excess false alarms. The output of the detection is reported every second. For a second of NTSC video, we output the percentage of visual cues in its 30 frames as:

$$C_v = \frac{1}{30}\sum_{f=1}^{30} v_f$$

3.2 Audio Event Detection

To detect events using an audio stream, we use a very simple power-based method similar to the one proposed by Clarkson and Pentland in [6][7]. This method adaptively normalizes signal power to zero mean and unity variance using a finite-length window; segments where the normalized power exceeds some threshold are designated "events." [6] and [7] describe an ambulatory system which could be exposed to arbitrary acoustic environments; adaptive normalization allows such a system to compensate for unusually loud or quiet environments and still detect events reliably. Our task differs from that system in that we have a *stationary* system where changes in power level really do indicate events and not just changes of venue. As such, instead of adaptive normalization, we use global normalization. That is, a single mean and variance is calculated for each two-hour recording and the globally-normalized power is threshold to detect events a_f.

In this implementation, we extracted 16-bit mono audio from the audio-video stream, and used analysis windows 200*ms* in length with a 50% overlap. This window length results in a frame rate of 10 frames per second, which is more than adequate to detect events using the power-based approach. After signal power is calculated and normalized, it is passed through a simple 3-frame averaging filter for smoothing. We then apply the power threshold; any segment which exceeds the threshold is designated an event. We also stipulate a minimum event time of 1 second in order to filter out isolated auditory transients. The confidence of audio event per second is defined as:

$$C_a = \frac{1}{10}\sum_{f=1}^{10} a_f$$

3.3 Fusing Video and Audio Event Detection

Video and audio streams are synchronized and segmented into one-second non-overlapping patches. The final event detection of each patch combines the video event confidence and audio event confidence linearly:

$$C_d = \alpha C_v + (1-\alpha)C_a$$

We consider a one-second patch to contain an interaction if its confidence C_d is higher than 0.5.

To evaluate the preprocessing algorithm, we labeled 10 hours of video/audio data. Using only video detection, we extract 33.3% of the entire video as candidate interaction shots, which is listed in Table 1. In order to not miss any interactions, we only filter out the one-second-long video segments with zero confidence.

Table 1. Results of event detection from video

	Total Event Time	Event Time as % of Total Signal
No activity	13711	38.1%
Individual	6700	18.6%
Multi-person	15589	33.3%

Table 2. Results of event detection from audio

Threshold	Total Event Time	Event Time as % of Total Signal
1.1	6705	18.6%
1.6	5582	15.5%
2.1	4327	12.0%

Using only audio detection with varying thresholds, we obtain the results listed in Table 2. The table shows the total event time and percentage of data in the recordings using three thresholds.

Table 3. Preprocessing results based on the ground-truth

	Recall	Precision	Process speed
Video	98%	13%	real time
Audio	71%	28%	10% real time
Multimodal	92%	21%	

By fusing the audio (threshold 1.6) and video results, we extracted total 9435 seconds from the entire 10 hours data. In this way, 85 our of 91 interactions in the ground truth are covered by the candidate shots, which obtain reasonable recall and precision in terms of event time as listed in Table 3. The audio has a lower recall due to the presence of silent interactions such as walking assistance of a wheelchair-bound patient. The audio precision is actually higher in general than is reported here. The hallway environment is a poor representative of audio precision, as many events that are audible in the hallway are off-camera and not in the ground-truth labels; thus audio event detection generates many false alarms. Even so, our results show that by fusing audio and video results, we can achieve more than 90% recall and 20% precision. We project even better precision when we test our fused system over the entire set of the data. The multi-person activity shots are then manually labeled using events selected by a group of doctors. Our study focuses on detecting interactions in multi-person activities, since social interaction is mutual or reciprocal action that involves only two people.

4 Sensor Simulation

A sensor is usually defined as a device that receives a signal or stimulus and responds to it in a distinctive manner. As we mentioned in the introduction, we consider both physical and algorithmic sensors in this study. For example, in order to investigate the temporal referencing probability of detecting an interaction, we consider "temporal interaction reference" as an algorithmic sensor, which is a detection result of another 1-second interval related to the current interval. On the other hand, to reduce number of candidate sensors, we omit some sensors that are impossible to implement from current technologies, such as speech recognition and facial expression understanding. There are some compromises between the technology capability and the medical request. We keep some sensors, for instance "hand trembling", which are very important for human experts but are questionable for real implementation. In detail, we select 21 events from the Pittsburgh Agitation Scale, which are listed in Table 4 and their occurrences in temporal neighborhoods as simulated sensors:

Table 4. Sensors defined on events and temporal neighborhood

Approaching	Leaving		- 5s	= Sensors
Standing	Hand trembling		- 4s	
Talking	Pushing a wheelchair		- 3s	
Shaking hands	Passing		- 2s	
Hand touch body slowly	Sitting		- 1s	
Hand touch body normally	Walking	×	0s	
Hand touch the body quickly	Hand in hand		+ 1s	
Hugging	Kiss		+ 2s	
Face turning	Kick		+ 3s	
Walking (moving) together	Sitting down		+ 4s	
Temporal interaction reference			+ 5s	

We label each shot second by second. The range of the temporal neighborhood is chosen from 5 seconds ahead to 5 seconds behind the current one. Overall we obtained 230 (21×11-1) simulated sensors, including 21 events times and 11 temporal neighbors, except the "temporal interaction reference (T-reference)" in the current interval, which is not considered as a sensor. All the sensors are labeled as binary events since there is no ambivalent in human experts' judgments during the labeling. We can see that one-second recording content may contain more than one direct or derived event detected by the simulated sensors.

To know which sensors would be most useful, we first analyze the effectiveness of individual sensors in detecting social interactions. The first measure that we use to study individual sensors is information gain [30]. Information gain indicates the potential power of each sensor in predicting an interaction. The details of how this technique works will not be covered in this paper. Table 5 lists top 28 sensors selected by information gain with respect to a correct prediction of a social interaction.

Table 5. Top 28 sensors selected by information gain technique

1	T-reference-1	8	Walking 0	15	Talking-2	22	Approaching+1
2	T-reference+1	9	T-reference-5	16	Walking+2	23	Walk together 0
3	T-reference-2	10	T-reference+4	17	Talking-3	24	Walking+3
4	T-reference+2	11	Walking+1	18	Talking+2	25	Talking-5
5	T-reference-3	12	Walking-1	19	Approaching 0	26	Approaching-1
6	T-reference+3	13	T-reference+5	20	Walking-2	27	Talking+3
7	T-reference-4	14	Talking+1	21	Talking-4	28	Leaving 0

The table shows that the T-reference of an interaction has obvious temporal consistency. Most interactions take longer than one second, and this consistency information is so important that it occupies the top 7 ranks with respect to the information gain scores.

Besides the temporal consistency, it also shows that the sensors of walking and talking are very important cues associated with an individual person; and relative location, such as approaching, leaving, walking together, and hand gesture are important between two persons. These sensors are important even in our daily experience. However, some sensors, such as "hand normal" and "pushing", which are also obvious evidence of an interaction, have very low ranks in information gain. They are either co-occurrences with some high rank sensors or omitted by the information gain technique due to their small number of examples.

Information gain takes an empirical risk to rank the sensors, which can be biased when training samples are redundant in some interaction patterns. For example, a long sequence of standing conversation will lead to higher ranks for talking and standing than that of a short sequence. It tends to omit the sensors with small numbers of examples in the training set, even though these sensors are very powerful in predicting social interactions. To avoid this kind of bias, we also analyze the power of each sensor using a structural risk based support vector machine (SVM) method [2]. This method trains an SVM using a subset of the training set from all sensors, and then eliminates sensors with low weight in representing the decision hyper-plane. Because the decision hyper-plane is trained to maximize the margin between the closest positive support vectors and negative support vectors, repeated patterns in the training set don't affect the result. Therefore, it is robust to the training set which contains a biased number of training examples for different sensors.

Table 6 lists the top 28 sensors selected by the SVM method. These 28 sensors cover most events in our total 21 events. Only "sitting" and "passing" are not included. This selection is more reasonable since the high rank sensors, such as "walk-together", "hand touch body normally", "talking", and "pushing", are obvious evidence of an interaction. The sensors with the top 2 ranks are still "T-reference" in the closest neighborhoods. This indicates that the 1-second interval is small and precise enough for analyzing social interactions in a nursing home environment. In comparison with the information gain results, the sensor "talking" is a common important sensor selected by both methods. The "walking" sensor is replaced by "walk together" and "pushing". They all overlap the sensor "walking", but provide more specific information. Hand related sensors are also ranked higher, which indicates that social interaction may benefit from developing better hand analysis sensors.

Table 6. Top 28 sensors selected by SVM

1	T-reference+1	8	Pushing+4	15	Pushing-3	22	Face turning 0
2	T-reference-1	9	Hand in hand 0	16	Walking+2	23	Walk together 0
3	Walk together 0	10	Kick 0	17	Face turning+1	24	Shaking hand+5
4	Hand normal 0	11	Hand slow 0	18	Approaching 0	25	Pushing+3
5	Talking 0	12	Hand-trem 0	19	Pushing-4	26	Hug+2
6	Pushing 0	13	T-reference-2	20	Hand normal+3	27	Standing+2
7	Talking+1	14	Leaving 0	21	Walk together+4	28	T-reference+2

Temporal information is included in our simulated sensors. We evaluated the effectiveness of temporal orders by averaging the two selection results together and computing the histogram of the temporal orders. Figure 2 illustrates the effectiveness of temporal order in detecting social interactions.

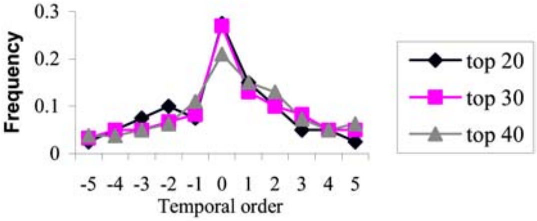


Fig. 2. Effectiveness of temporal order

The effectiveness of the temporal order drops quickly as the time span from the current time increases from zero. The effect of events more than 3 seconds away from the current one is very limited and can provide little useful information for analyzing social interactions. Sensor selection only analyzes the effectiveness of individual sensors; in the next section we will investigate the power of combinations of sensors using statistical models.

5 Detection Models

It should be noted that there are some overlaps among simulated sensors, e.g., “walking together” implies “walking”. The first goal of this section is to explore proper statistical models to detect social interactions. We consider that the detection of a social interaction is a binary classification problem: interaction and non-interaction. The other goal of this section is to further investigate the associations between different sensors. This will enable us to replace some impracticable sensors with combinations of sensors that can be more easily developed. Since we have considered including temporal information in the simulated sensors, the interaction detection problem can be simplified as a problem to classify the sensor outputs of each 1-second interval into two classes, indicating interaction and non-interaction respectively.

To find a proper model for classifying interactions, we evaluated various machine learning algorithms: decision tree [28], naive Bayesian [22], Bayes network [17], logistic regression [14], support vector machine [35], adaboost [25], and logitboost [12]. We will not describe details of these algorithms in this paper; interested readers can find these details in the references.

The evaluations are shown in Table 7. We use equal size training and testing data. Standard 5-fold cross-validation is performed to find optimal parameters for each model. We then perform the resulted optimal models on the testing set to report the numbers in Table 7.

Table 7. Performance of interaction detection using different models

	With T-reference			**Without T-reference**		
Model	**Prec.**	**Recall**	**F-measure**	**Prec.**	**Recall**	**F-measure**
Decision tree	99.5%	99.2%	99.3%	97.1%	**96.4%**	**96.8%**
Naive Bayesian	98.4%	92.9%	95.6%	96.3%	90.1%	93.1%
Bayes network	98.4%	93.0%	95.6%	96.3%	90.4%	93.3%
Logistic reg.	99.6%	98.7%	99.2%	96.5%	94.5%	95.5%
SVM	99.5%	**99.5%**	**99.5%**	**98.0%**	95.1%	96.5
adaboost	**99.7%**	99.1%	99.4%	95.4%	93.9%	94.6%
logitboost	**99.7%**	99.1%	99.4%	96.0%	95.6%	95.8%

We can see that under the ideal conditions (all sensors output correct result without any ambiguity), all these models obtain good detection results. To our surprise, the simplest method, decision tree, employs only four kinds of sensors: “T-reference”, “talking”, “walking”, and “leaving”, but achieves very good performance. None of these sensors except “T-reference” requires complex visual and audio analysis in comparison to sensors such as “face turning” and “hand in hand”. It seems possible that social interaction can be detected by just developing good “talking”, “walking”, and “leaving” sensors. It is true if the “T-reference” sensor can be successfully derived from these three kinds of sensors.

To remove the effect of the temporal information of the derived sensor “T-reference”, we assume that the “T-reference” sensor is not available to its neighbors. We remove all “T-reference” sensor outputs from feature vectors and evaluate the above methods. The results are also listed in Table 7. After removing the “T-reference” sensor, the performance drops about 3-5%, which indicates that we can achieve around 90% accuracy in detecting current interaction with the temporal information of interaction decisions in neighborhoods. As we assume outputs of other sensors are under ideal conditions, the real accuracy of the current “T-reference” sensor output is expected to be about 90% of the average accuracy of all the other sensors’ outputs. The decision tree still achieved the best performance even without the “T-reference” sensors. However, the resulting decision tree includes all kinds of sensors. The top 10 sensors are:

Rank	**Sensor**	**Rank**	**Sensor**
1	Talking	6	Hand in hand
2	Walk together	7	Standing
3	Walking	8	Leaving
4	Pushing	9	Approaching
5	Hand normal	10	Passing

A drawback of the decision tree is that it is sensitive to the noise in sensor outputs. In practice, outputs of sensors might be ambiguous or even incorrect. Some of the sensor outputs have to be represented by probabilities, e.g., 60% “talking” or 30% “hand in hand”. The uncertainties of sensor outputs can only be determined from real data of experiments. What we can do in a simulation is to add some noise into outputs of sensors. Table 8 lists results of adding 20% noise (20% sensors have wrong outputs) into the data without "T-reference" sensors.

Table 8. Performances of interaction detection using different models with 20% noise

Model	Prec.	Recall	F-measure
Decision tree	90.0%	90.4%	90.2%
Naive Bayesian	88.6%	75.3%	81.4%
Bayes network	88.1%	77.6%	82.5%
Logistic regression	90.1%	93.5%	91.8%
SVM	91.4%	95.3%	93.3%
adaboost	89.6%	93.8%	91.6%
logitboost	90.1%	95.6%	92.8%

The performance of the decision tree decreases from 96.8% (F-measure) to 90.2%, or loses 6.6% accuracy. At the same time, the performance of the SVM model decreases from 96.5% to 93.3%, or only loses 3.2% accuracy. Notably, the recall of the SVM only decreases 0.5% with 20% noise. The logitboost model also proved to be robust against noise; the recall remains the same after adding noise. The F-measure loses only 3% accuracy. This indicates that the SVM model is potentially more robust than the decision tree model in real applications.

It should be noted that the noise level of 20% is an empirical assumption. Real sensors will have different accuracies. According to our preliminary implementations of some real sensors, walking related sensors introduce around 15% noise on average, the "standing" sensor only has 6% noise, the "talking" sensor produces about 30% noise, and face and hand related sensors are still under development. If the noise range of face and hand related sensors are 40%-60%, 20% noise on average for all sensors is reasonable.

6 Conclusions

This paper presents a study of feasibility of sensor-based analysis of social interaction patterns in a skilled nursing facility. We have analyzed the capabilities of various individual sensors for detecting social interactions. The relative location related sensors, hand related sensors, talking sensors, and temporal consistency information are ranked high priorities in the task of detecting interactions.

We have also compared various statistical models to explore overlapped spaces of multiple sensors. The experimental results have indicated that the decision tree model could achieve more than 99% accuracy with only three kinds of sensors: "talking", "walking", and "leaving", plus temporal information under noise free conditions. This indicates the possibility of achieving good interaction detection performance by developing perfect "talking", "walking", and "leaving" sensors, instead of developing complex ones, such as face and hand gesture sensors. We also demonstrated the robustness of various models when noisy sensors are considered. The SVM model and the logitboost model have been proven to be more robust against noise than other models. Based on the promising results from this study, we will develop working systems with real sensors. We will further classify social interaction patterns and evaluate those systems and algorithms.

Reference

1. Aggarwal, J. K., Cai, Q. Human Motion Analysis: A Review. *Computer Vision and Image Understanding*, Vol. 73, pp. 428-440, 1999.
2. Brank, J., Grobelnik, M., Milic-Frayling, N. and Mladenic, D. Feature selection using linear support vector machines. MSR-TR-2002-63, Microsoft research 2002.
3. Bregler, C. Learning and Recognizing Human Dynamics in Video Sequences. In CVPR, pages 568-574, 1997.
4. Brumitt, B., Krumm, J., Meyers, B. and Shafer, S. Ubiquitous computing and the role of geometry. In Special Issue on Smart Spaces and Environments, volume 7-5, pages 41-43. IEEE Personal Communications, October 2000.
5. Carp, F. Assessing the environment. Annul review of gerontology and geriatrics, 14, pages: 302-314, 1994.
6. Clarkson, B. and Pentland, A. Framing Through Peripheral Perception. Proc. of ICIP, Vancouver, September 2000.
7. Clarkson, B. and Pentland, A. Unsupervised Clustering of Ambulatory Audio and Video. Proc. of the ICASSP, Phoenix, 1998.
8. Emler N., Gossip, reputation, and social adaptation. In R.F.Goodman and A. Ben-Ze'ev (Eds.) Good Gossip, pages.117-138. Wichita, Kansas, USA: University Press of Kansas 1994
9. Eppig, F. J. and Poisal, J. A. Mental health of medicare beneficiaries: 1995. Health Care Financing Review, 15, pages: 207-210, 1995.
10. Essa, I. and Pentland, A. Facial expression recognition using a dynamic model and motion energy. In Proc. 5th Intl. Conf. on Computer Vision, pages 360--367, 1995.
11. Freeman, W. T. and Roth, M. Orientation histograms for hand gesture recognition. In International Workshop on Automatic Face and Gesture Recognition, pages 296-301, June 1995.
12. Friedman, J., Hastie, T. and Tibshirani, R. Additive logistic regression: a statistical view of boosting. Annals of Statistics, 28:307--337, 2000.
13. German, P.S., Rovner, B.W., Burton, L.C., Brant, L.J. and Clark, R. The role of mental morbidity in the nursing home experience. Gerontologist, 32(2): 152-158, 1992.
14. Hastie, T. and Tibshirani, R. Nonpararmetric logistic and proportional odds regression. Applied statistics 36:260-276, 1987.
15. Harter, A., Hopper, A., Steggles, P., Ward, A. and Webster, P. The anatomy of a context-aware application. In Proceedings of the 5th Annual ACM/IEEE International Conference on Mobile Computing and Networking, pages 59-68, Seattle, WA, August 1999.
16. Holmquist, L., Falk, J. and Wigstrm, J. Supporting group collaboration with interpersonal awareness devices. *Personal Technologies*, 3:13–21, 1999.
17. Hooyman, N.R. and Kiyak, H.A. Social Gerontology: A Multidisciplinary Perspective. 6th ed., Allyn and Bacon 2002.
18. Hudson, S., Fogarty, J., Atkeson, C., Avrahami, D., Forlizzi, J., Kiesler, S., Lee, J. and Yang, J. Predicting Human Interruptibility with Sensors: A Wizard of Oz Feasibility Study. In Proc. of the SIGCHI Conference on Human Factors in Computing Systems, pages 257-264 2003.
19. Jug, M., Pers, J., Dezman, B. and Kovacic, S. Trajectory based assessment of coordinated human activity. In *ICVS 2003*, pages 534–543, 2003.
20. Kidd, C. D., Orr, R., Abowd, G. D., Atkeson, C. G., Essa, I. A., Macintyre, B., Mynatt, E. and Starner, T. E. and Newstetter, W. The Aware Home: A Living Laboratory for Ubiquitous Computing Research. *Proc. of CoBuild '99*, pp.191-198, 1999.

21. Koile, K., Tollmar, K., Demirdjian, D., Shrobe, H. E., Darrell, T. Activity Zones for Context-Aware Computing. Ubicomp 2003, pp. 90-106, 2003.
22. Kononenko I., Semi-naive bayesian classifier. In Proceedings of sixth European Working Session on Learning, pages 206-219. Springer-Verlag, 1991.
23. Lee, S. and Mase, K. Activity and location recognition using wearable sensors. In *1st IEEE International Conference on Pervasive Computing and Communications*, pages 24–32, 2002.
24. Lubinski, R. Dementia and communication. Philadelphia: B. C. Decker, 1991.
25. Margineantu, D. D. and Dietterich, T. G. Pruning adaptive boosting. In 14th Int. Conf. on Machine Learning, pages 211-218. Morgan Kaufmann, 1997.
26. Martin, A., Karrray, L. and Gilloire, A. High Order Statistics for Robust Speech/Non-Speech Detection. In Eusipco, Tampere, Finland, Sept. 2000, pp. 469--472.
27. Nelson, J. The influence of environmental factors in incidents of disruptive behavior. Journal of Gerontological Nursing 21(5):19-24, 1995.
28. Quinlan, J. R. C4.5: programs for machine learning. Morgan Kaufmann 1993.
29. Rhodes, B. The wearable remembrance agent: A system for augmented memory. In *Proceedings of the 1st International Symposium on Wearable Computers*, pp: 123–128, 1997.
30. Schraudolph, N. and Sejnowski, T. J. Unsupervised discrimination of clustered data via optimization of binary information gain. In Stephen Jose Hanson, Jack D. Cowan, and C. Lee Giles, editors, Advances in Neural Information Processing Systems, volume 5, pages 499-506. Morgan Kaufmann, San Mateo, 1993.
31. Sloane, P. D., Mitchell, C. M., Long, K. and Lynn, M. TESS 2+ Instrument B: Unit observation checklist – physical environment: A report on the psychometric properties of individual items, and initial recommendations on scaling. University of North Carolina 1995.
32. Steele, C., Rovner, B. W., Chase, G. A. and Folstein, M. Psychiatric symptoms and nursing home placement in Alzheimer's disease. *American Journal of Psychiatry*, 147(8): pp.1049-1051, 1990.
33. Stauffer, C. and Grimson, W. E. L. Adaptive background mixture models for real-time tracking. Proc. of CVPR 1999.
34. Time Domain Corporation, 7057 Old Madison Pike, Huntsville, AL 35806. PulsON Technology: Time Modulated Ultra Wideband Overview, 2001.
35. Vapnik, V.N. The nature of statistical learning theory. Springer Verlag, New York, 1995.
36. Zhang, D., Li, S. Z., Gatica-Perez, D. Real-Time Face Detection Using Boosting Learning in Hierarchical Feature Spaces. 17th International Conference on Pattern Recognition 2004.

HMM Based Falling Person Detection Using Both Audio and Video*

B. Uğur Töreyin[1], Yiğithan Dedeoğlu[2], and A. Enis Çetin[1]

[1] Department of Electrical and Electronics Engineering,
Bilkent University 06800 Bilkent, Ankara, Turkey
{bugur,cetin}@bilkent.edu.tr
[2] Department of Computer Engineering,
Bilkent University 06800 Bilkent, Ankara, Turkey
yigithan@bilkent.edu.tr

Abstract. Automatic detection of a falling person in video is an important problem with applications in security and safety areas including supportive home environments and CCTV surveillance systems. Human motion in video is modeled using Hidden Markov Models (HMM) in this paper. In addition, the audio track of the video is also used to distinguish a person simply sitting on a floor from a person stumbling and falling. Most video recording systems have the capability of recording audio as well and the impact sound of a falling person is also available as an additional clue. Audio channel data based decision is also reached using HMMs and fused with results of HMMs modeling the video data to reach a final decision.

1 Introduction

Detection of a falling person in an unsupervised area is a practical problem with applications in safety and security areas including supportive home environments and CCTV surveillance systems. Intelligent homes will have the capability of monitoring activities of their occupants and automatically provide assistance to elderly people and young children using a multitude of sensors including surveillance cameras in the near future [1, 2, 3]. Currently used worn sensors include passive infrared sensors, accelerometers and pressure pads. However, they may produce false alarms and elderly people simply forget wearing them very often. Computer vision based systems propose non-invasive alternatives for fall detection. In this paper, a video based falling person detection method is described. Both audio and video tracks of the video are used to reach a decision.

Video analysis algorithm starts with moving region detection in the current image. Bounding box of the moving region is determined and parameters describing the bounding box are estimated. In this way, a time-series signal describing the motion of a person in video is extracted. The wavelet transform of this signal is computed and used in Hidden Markov Models (HMMs) which were trained according to possible human being motions. It is observed that the wavelet transform domain signal provides better results than the time-domain signal because wavelets capture sudden changes in the signal and ignore stationary parts of the signal.

* This work is supported in part by European Commission 6th Framework Program with grant number FP6-507752 (MUSCLE Network of Excellence Project)

N. Sebe, M.S. Lew, and T.S. Huang (Eds.): HCI/ICCV 2005, LNCS 3766, pp. 211–220, 2005.

Audio analysis algorithm also uses the wavelet domain data. HMMs describing the regular motion of a person and a falling person were used to reach a decision and fused with results of HMMs modeling the video data to reach a final decision.

In [4] motion trajectories extracted from an omnidirectional video are used to determine falling persons. When a low cost standard camera is used instead of an omnidirectional camera it is hard to estimate moving object trajectories in a room. Our fall detection method can be also used together with [4] to achieve a very robust system, if an omnidirectional camera is available. Another trajectory based human activity detection work is presented in [5]. Neither [4] nor [5] used audio information to understand video events.

In Section 2, the video analysis algorithm is described and in Section 3, the audio analysis algorithm is presented. In Section 4, experimental results are presented.

2 Analysis of Video Track Data

Our video analysis consists of three steps: i) moving region detection in video, ii) calculation of wavelet coefficients of a parameter related with the aspect ratio of the bounding box of the moving region, and iii) HMM based classification using the wavelet domain data. Each step of our video analysis algorithm is explained in detail next.

i) Moving region detection: The camera monitoring the room is assumed to be stationary. Moving pixels and regions in the video are determined by using a background estimation method developed in [6]. In this method, a background image B_{n+1} at time instant $n+1$ is recursively estimated from the image frame I_n and the background image B_n of the video as follows:

$$B_{n+1}(k,l) = \begin{cases} aB_n(k,l) + (1-a)\, I_n(k,l), & \text{if } I_n(k,l) \text{ stationary} \\ B_n(k,l), & \text{if } I_n(k,l) \text{ moving} \end{cases} \tag{1}$$

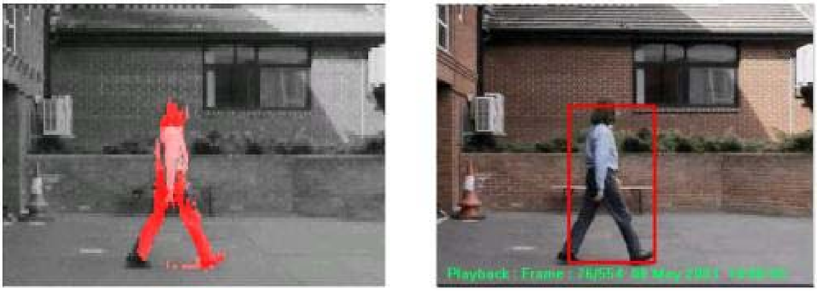
where $I_n(k, l)$ represents a pixel in the n^{th} video frame I_n, and a is a parameter between 0 and 1. Moving pixels are determined by subtracting the current image from the background image and adaptive thresholding (cf. Fig. 1a). For each pixel an adaptive threshold is estimated recursively in [6]. Pixels exceeding thresholds form moving regions and they are determined by connected component analysis.

We do not need very accurate boundaries of moving regions. Hence the above computationally efficient algorithm is sufficient for our purpose of estimating the aspect ratios of moving regions in video. Other methods including the ones described in [7] and [8] can also be used for moving pixel estimation but they are computationally more expensive than [6].

ii) Feature extraction from moving regions and the wavelet transform: After a post-processing stage comprising of connecting the pixels, moving regions are encapsulated with their minimum bounding rectangles (cf. Fig.1b). Next, the aspect ratio, ρ, for each moving object is calculated. The aspect ratio of the i^{th} moving object is defined as:

$$\rho_i(n) = \frac{H_i(n)}{W_i(n)} \tag{2}$$

where $H_i(n)$ and $W_i(n)$ are the height and the width of the minimum bounding box of the i^{th} object at image frame n, respectively, We then calculate the corresponding wavelet coefficients for ρ. Wavelet coefficients, w_i's, are obtained by high-pass filtering followed by decimation as shown in Fig. 2.



(a) (b)

Fig. 1. (a) Moving pixels, and (b) their minimum bounding boxes are determined

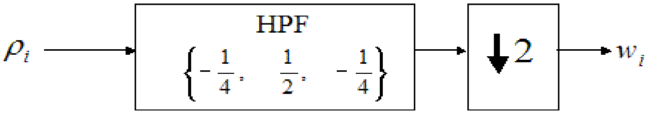


Fig. 2. Wavelet coefficients, w_i corresponding to aspect ratio ρ_i are evaluated with an integer arithmetic high-pass filter (HPF) corresponding to Lagrange wavelets [9] followed by decimation

The wavelet transform of the one-dimensional aspect ratio signal is used as a feature signal in HMM based classification in this paper. It is experimentally observed that the aspect ratio based feature signal exhibits different behaviour for the cases of walking and falling persons. A quasi-periodic behaviour is obviously apparent for a walking person in both $\rho(n)$ and its corresponding wavelet signal as shown in Fig. 3. On the other hand, the periodic behaviour abruptly ends and $\rho(n)$ decays to zero for a falling person or a person sitting down. This decrease and the later stationary characteristic for fall is also apparent in the corresponding subband signal (cf. Fig. 4).

Using wavelet coefficients, w, instead of aspect ratios, ρ, to characterize moving regions has two major advantages. The primary advantage is that, wavelet signals can easily reveal the aperiodic characteristic which is intrinsic in the falling case. After the fall, the aspect ratio does not change or changes slowly. Since, wavelet signals are high-pass filtered signals, slow variations in the original signal lead to zero-mean wavelet signals. Hence it is easier to set thresholds in the wavelet domain which are robust to variations of posture sizes and aspect ratios for different people. This constitutes the second major advantage. We set two thresholds, $T1$ and $T2$ for defining Markov states in the wavelet domain as shown in Fig. 3. The lower threshold $T1$ basically determines the wavelet signal being close to zero. After the fall, ideally the wavelet signal should be zero but due to noise and slow movements of the fallen person the wavelet coefficients wiggle around zero. The use of wavelet domain information also makes the method robust to variations in object sizes. This is achieved by the use of the second threshold *T2* to detect high amplitude variations in the wavelet signal, which correspond to edges or high-frequency changes in the original signal. When the wavelet coefficients exceed the higher threshold *T2* in a frequent manner this means that the object is changing its shape or exhibiting periodic behaviour due to walking or running.

iii) HMM based classification: Two three-state Markov models are used to classify the motion of a person in this paper. Non-negative thresholds *T1* < *T2* introduced in wavelet domain, define the three states of the Hidden Markov Models for walking and falling, as shown in Fig. 5a and b, respectively.

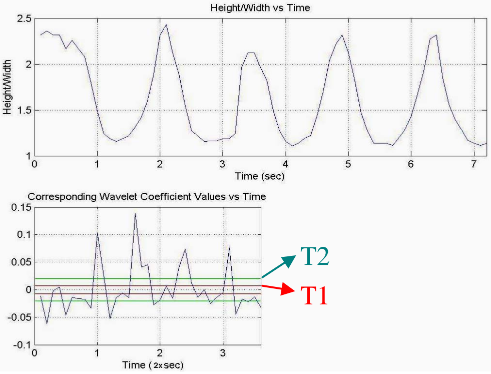


Fig. 3. Quasi-periodic behaviour in ρ vs. time (top), and the corresponding wavelet coefficients *w* vs. time for a walking person (sampling period is half of the original rate in the wavelet plot). Thresholds *T1* and *T2* introduced in the wavelet domain are robust to variations in posture sizes and aspect ratios of different people

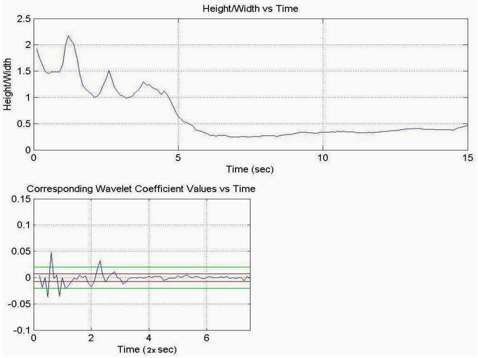


Fig. 4. Aspect ratio ρ vs. time (top), and the corresponding wavelet coefficients *w* vs. time for a falling person (sampling period is half of the original rate in the wavelet plot)

At time *n,* if $|w_i(n)| < T1$, the state is in S1; if $T1 < |w_i(n)| < T2$, the state is S2; else if $|w_i(n)| > T2$, the state S3 is attained. During the training phase of the HMMs transition probabilities a_{uv} and b_{uv}, $u,v = 1, 2, 3,$ for walking and falling models are estimated off-line, from a set of training videos. In our experiments, 20 consecutive image frames are used for training HMMs.

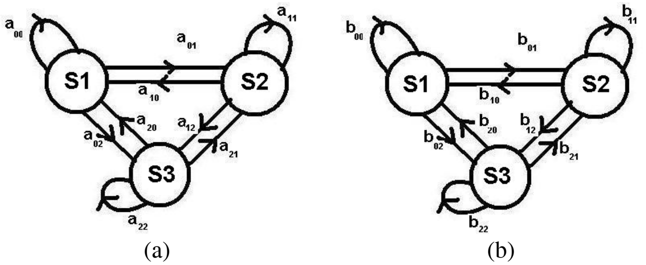


Fig. 5. Three state Markov models for (a) walking, and (b) falling

For the walking person, since the motion is quasi-periodic, we expect similar transition probabilities between the states. Therefore the values of *a's* are close to each other. However, when the person falls down, the wavelet signal starts to take values around zero. Hence we expect a higher probability value for b_{00} than any other b value in the falling model, which corresponds to higher probability of being in *S1*. The state *S2* provides hysteresis and it prevents sudden transitions from *S1* to *S3* or vice versa.

During the recognition phase the state history of length 20 image frames are determined for the moving object detected in the viewing range of the camera. This state sequence is fed to the walking and falling models. The model yielding higher probability is determined as the result of the analysis for video track data. However, this is not enough to reach a final decision of a fall. Similar w vs. time characteristics are observed for both falling and ordinary sitting down cases. A person may simply sit down and stay stationary for a while. To differentiate between the two cases, we incorporate the analysis of the audio track data to the decision process.

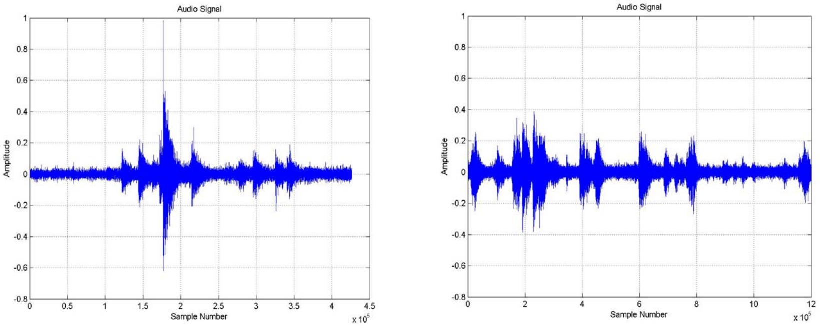
3 Analysis of Audio Track Data

In this paper, audio signals are used to discriminate between falling and sitting down cases. A typical stumble and fall produces high amplitude sounds as shown in Fig. 6a, whereas the ordinary actions of bending or sitting down has no distinguishable sound from the background (cf. Fig. 6b). The wavelet coefficients of a fall sound are also different from bending or sitting down as shown in Fig. 7. Similar to the motivation in the analysis of video track data for using wavelet coefficients, we base our audio analysis on wavelet domain signals. Our previous experience in speech recognition indicates that wavelet domain feature extraction produces more robust results than Fourier domain feature extraction [10]. Our audio analysis algorithm also consists of three steps: i) computation of the wavelet signal, ii) feature extraction of the wavelet signal, and iii) HMM based classification using wavelet domain features.

i) Wavelet signal: We use the same high-pass filter followed by a decimation block shown in Fig. 2, to obtain a wavelet signal corresponding to the audio signal accompanying the video track data. The wavelet signals corresponding to the audio track data in Fig. 6a and b, are shown in Fig. 7a and b, respectively.

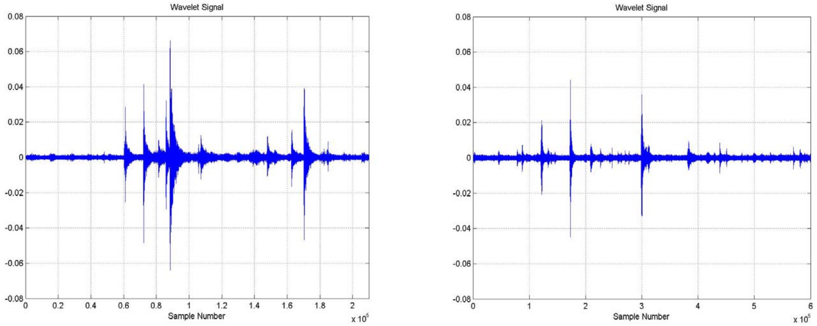
ii) Analysis of wavelet signals: The wavelet signals corresponding to audio track data are further analyzed to extract features in fixed length short-time windows. We take 500-sample-windows in our implementation. Our sampling frequency is 44.1

KHz. We determine the variance, σ_i^2, and the number of zero crossings, Z_i, in each window i.



(a) (b)

Fig. 6. Audio signals corresponding to (a) a fall, which takes place at around sample number 1.8x10^5, and (b) talking (0 - 4.8x10^5), bending (4.8x10^5 – 5.8x10^5), talking (5.8x10^5 – 8.9x10^5), walking (8.9x10^5 – 10.1x10^5), bending (10.1x10^5 - 11x10^5), and talking (11x10^5 - 12 x10^5) cases. The sound signals are sampled with 44,100 Hz



(a) (b)

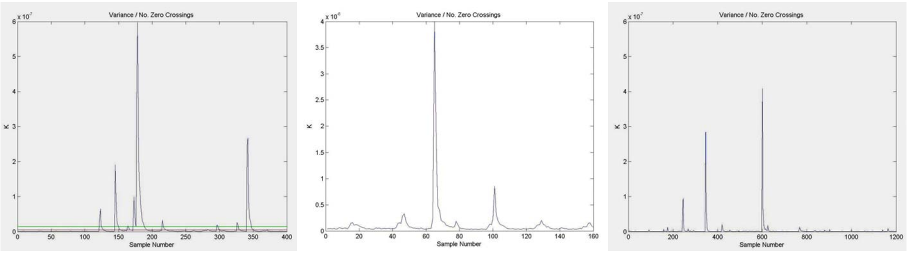
Fig. 7. The wavelet signals corresponding to the audio signals in (a) falling (0.9 x10^5), and (b) talking (0 – 2.4x10^5), bending (2.4x10^5 – 2.9x10^5), talking (2.9x10^5 – 4.5x10^5), walking (4.5x10^5 – 5x10^5), bending (5x10^5 – 5.5x10^5), and talking (5.5x10^5 - 6 x10^5)

We observe that, walking is a quasi-periodic sound in terms of σ_i^2 and Z_i. However, when a person stumbles and falls, Z_i decreases whereas σ_i^2 increases. So we define a feature parameter κ in each window as follows:

$$\kappa = \frac{\sigma^2}{Z_i} \tag{3}$$

where the index i indicates the window number. The parameter κ takes non-negative values.

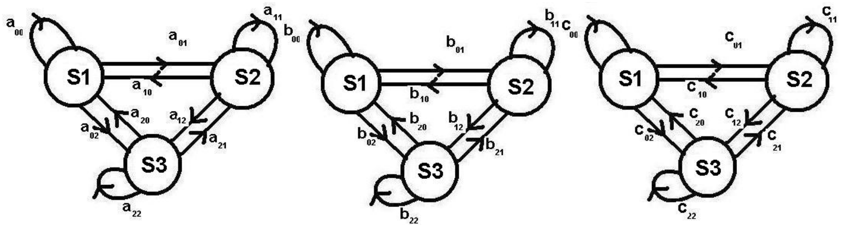
Talking has a varying σ_i^2-Z_i characteristic depending on the utterance. When vowels are uttered, σ_i^2 increases while Z_i decreases, which results in larger κ values compared to consonant utterances. Variation of κ values versus sample numbers for different cases, are shown in Fig. 8.



(a) (b) (c)

Fig. 8. The ratio of variance over number of zero crossings, κ, variations for **(a)** falling (180), **(b)** walking, and **(c)** talking (0 – 480), bending (480 – 590), talking (590 – 900), walking (900 – 1000), bending (1000 – 1100), and talking (1100 - 1200). Note that, κ values for (b) the walking case, are an order of magnitude less than (a) falling and (c) talking cases. Thresholds *T1' < T2'*, are defined in κ domain

iii) HMM based classification: In this case, three three-state Markov models are used to classify the walking, talking and falling sounds. The non-negative thresholds *T1' < T2'* introduced in κ domain, define three states of the Hidden Markov Models for walking, talking and falling, as shown in Fig. 9a, b, and c, respectively.



(a) (b) (c)

Fig. 9. Three-state Markov models for **(a)** walking, **(b)** talking, and **(c)** falling sound classification

For the i^{th} window of the wavelet signal, if $|\kappa_i| < T1'$, state S1; if $T1' < |\kappa_i| < T2'$, state S2; else if $|\kappa_i| > T2'$, state S3 is attained. During the training phase, transition probabilities, a_{uv}, b_{uv}, and c_{uv}, $u,v = 1, 2, 3,$ for walking, talking, and falling models, respectively, are estimated off-line. These probabilities are estimated from 20 consecutive κ values corresponding to 20 consecutive 500-sample-long wavelet windows for training HMMs.

During the classification phase a state history signal consisting of 20 κ values are estimated from the sound track of the video. This state sequence is fed to the walking, talking, and falling models in running windows. The model yielding highest probability is determined as the result of the analysis for audio track data. We then combine this result with the result of the video track analysis step using the logical "and" operation. Therefore, a "falling person detected" alarm is issued only when both video and audio track data yield the highest probability in their "fall" models.

4 Experimental Results

The proposed algorithm works in real-time on an AMD AthlonXP 2000+ 1.66GHz processor. As described above HMMs are trained from falling, walking, and walking and talking video clips. A total of 64 video clips having 15,823 image frames are used. In all of the clips, only one moving object exists in the scene. Contents of the test video clips are summarized in Table 1.

Table 1. Video content distribution in the test set

Video Content	Include Audio	No. of Clips
Walking + Talking	Yes	16
Sitting down + Talking	Yes	5
Sitting down	Yes	4
Walking + Falling	Yes	25
Walking + Falling	No	14

As can be seen from Table 1, 14 of the clips having falls do not have audio track data, hence we only make use of the video track data analysis part of our method to determine whether falling takes place. Image frames from the above video clips are shown in Figure 10.

The classification results for the above test data with only video analysis and both audio and video analysis are presented in Table 2. There is no way to distinguish a person intentionally sitting down on the floor from a falling person, if only video track data is used. When both modalities are utilized, they can be distinguished and we do not get any false positive alarms for the videos having a person sitting down as shown in Table 2.

Table 2. Detection results for the test set

Video Content	Include Audio	No. of Clips	No. of Clips in which Falling is Detected	
			Video	Audio+Video
Walking + Talking	Yes	16	0	0
Sitting down + Talking	Yes	5	5	0
Sitting down	Yes	4	4	0
Walking + Falling	Yes	25	25	25
Walking + Falling	No	14	14	14

5 Conclusion

A method for automatic detection of a falling person in video is developed. Main contribution of this work is the use of both audio and video tracks to decide a fall in video. The audio information is essential to distinguish a falling person from a person simply sitting down or sitting on a floor. Three-state HMMs are used to classify events. Feature parameters of HMMs are extracted from temporal wavelet signals describing the bounding box of moving objects. Since wavelet signals are zero-mean signals, it is easier to define states in HMMs and this leads to a robust method against variations in object sizes.

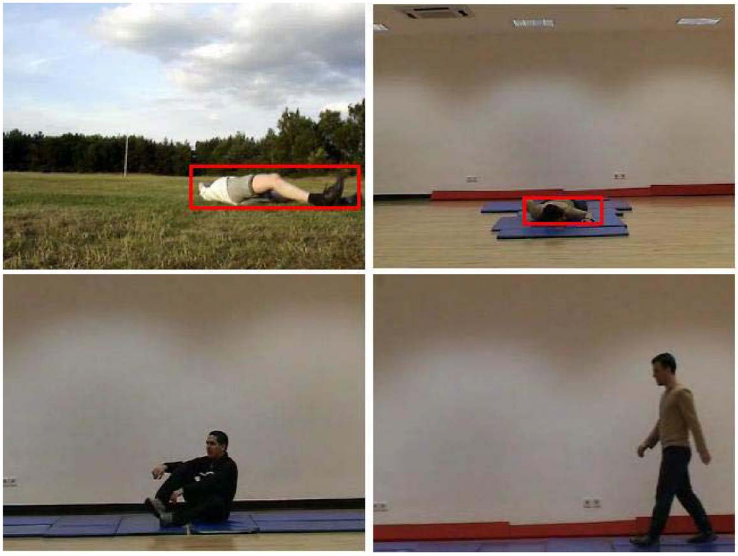


Fig. 10. Image frames from falling, sitting, and walking and talking clips

The method is computationally efficient and it can be implemented in real-time in a PC type computer.

Similar HMM structures can be also used for automatic detection of accidents and stopped vehicles in highways which are all examples of instantaneous events occurring in video.

References

1. Barnes, N.M., Edwards, N.H., Rose, D.A.D., Garner, P.: Lifestyle Monitoring: Technology for Supported Independence. IEE Comp. and Control Eng. J. (1998) 169-174
2. Bonner, S.: Assisted Interactive Dwelling House: Edinvar Housing Assoc. Smart Tech. Demonstrator and Evaluation Site In: Improving the Quality of Life for the European Citizen (TIDE), (1997) 396–400
3. McKenna, S.J., Marquis-Faulkes, F., Gregor, P., Newell, A.F.:Scenario-based Drama as a Tool for Investigating User Requirements with Application to Home Monitoring for Elderly People. In Proc. of HCI, (2003)
4. Nait-Charif, H., McKenna, S.: Activity Summarisation and Fall Detection in a Supportive Home Environment. In Proc. of ICPR'04, (2004) 323-326
5. Cuntoor, N.P., Yegnanarayana, B., Chellappa, R.: Interpretation of State Sequences in HMM for Activity Representation. In Proc. of IEEE ICASSP'05, (2005) 709-712
6. Collins, R.T., Lipton, A.J., Kanade, T., Fujiyoshi, H., Duggins, D., Tsin, Y., Tolliver, D., Enomoto, N., Hasegawa, O., Burt, P., Wixson, L.: A System for Video Surveillance and Monitoring: VSAM Final Report.Tech. Report CMU-RI-TR-00- 12, Carnegie Mellon University (1998)

7. Bagci, M., Yardimci, Y., Cetin, A.E.:Moving Object Detection Using Adaptive Subband Decomposition and Fractional Lower Order Statistics in Video Sequences. Elsevier, Signal Processing. (2002) 1941—1947
8. Stauffer, C., Grimson, W.E.L.: Adaptive Background Mixture Models for Real-Time Tracking. Proceedings of IEEE Computer Society Conference on Computer Vision and Pattern Recognition. (1999) 246-252
9. Kim, C.W., Ansari, R., Cetin, A.E.: A class of linear-phase regular biorthogonal wavelets. In Proc. of IEEE ICASSP'92 (1992) 673-676
10. Jabloun, F., Cetin, A.E., Erzin, E.: Teager Energy Based Feature Parameters for Speech Recognition in Car Noise. IEEE Signal Processing Letters (1999) 259-261

Appearance Manifold of Facial Expression

Caifeng Shan, Shaogang Gong, and Peter W. McOwan

Department of Computer Science
Queen Mary, University of London, London E1 4NS, UK
{cfshan,sgg,pmco}@dcs.qmul.ac.uk

Abstract. This paper investigates the appearance manifold of facial expression: embedding image sequences of facial expression from the high dimensional appearance feature space to a low dimensional manifold. We explore Locality Preserving Projections (LPP) to learn expression manifolds from two kinds of feature space: raw image data and Local Binary Patterns (LBP). For manifolds of different subjects, we propose a novel alignment algorithm to define a global coordinate space, and align them on one generalized manifold. Extensive experiments on 96 subjects from the Cohn-Kanade database illustrate the effectiveness of the alignment algorithm. The proposed generalized appearance manifold provides a unified framework for automatic facial expression analysis.

1 Introduction

The ability to recognize affective states of a person is indispensable and very important for successful interpersonal social interaction. Human-Computer Interaction (HCI) designs need to include the ability of affective computing, in order to become more human-like, more effective, and more efficient [13]. Affective arousal modulates all nonverbal communication cues such as facial expressions, body postures and movements. Facial expression is perhaps the most natural and efficient means for humans to communicate their emotions and intentions, as communication is primarily carried out face to face. Therefore, automatic facial expression analysis has attracted much attention [5, 12, 20] in recent years. Though much progress has been made [3, 4, 19], recognizing facial expression with a high accuracy remains to be difficult due to the complexity and variety of facial expressions.

A face image with N pixels can be considered as a point in the N-dimensional image space, and the variations of face images can be represented as low dimensional manifolds embedded in the high dimensional image space [6–8, 14, 17]. It would be desired to analyze facial expressions in the low dimensional subspace rather than the ambient space. However, research on the manifold of facial expression has been very limited as far as it goes. Chang et al. [2] made first attempt to apply two types of embedding, Locally Linear Embedding (LLE) [14] and Lipschitz embedding, to learn the structure of the expression manifold. In [3], they further proposed a probabilistic video-based facial expression recognition method on manifolds. A complete expression sequence becomes a path on

N. Sebe, M.S. Lew, and T.S. Huang (Eds.): HCI/ICCV 2005, LNCS 3766, pp. 221–230, 2005.

the expression manifold, and the transition between basic expressions is represented as the evolution of the posterior probability of the six basic paths. Based on a expression manifold obtained by Isomap embedding [17], they also proposed an approach for facial expression tracking and recognition [9]. However, the existing research learned the expression manifold in the feature space described by a set of facial landmarks such as 58 facial points [2, 3]; the appearance manifold of facial expression is still unknown. Moreover, the existing research was conducted on data sets containing only several subjects [2, 3]; there is no published work on the expression manifold carried out on a large number of subjects.

A number of nonlinear techniques have been proposed to learn the structure of a manifold, e.g., Isomap [17], LLE [14], and Laplacian Eigenmap (LE) [1]. However, these techniques yield maps that are defined only on the training data, and it is unclear how to evaluate the maps for new test data, although some mapping methods were discussed in [14]. Therefore, they may not be suitable for expression recognition tasks. Recently He and Niyogi [7] proposed a general manifold learning method called Locality Preserving Projections (LPP) (Section 2), which are obtained by finding the optimal linear approximations to the eigenfunctions of the Laplace Beltrami operator on the manifold. Different from PCA, which implicitly assumes that the data space is Euclidean, LPP assumes that the data space is a linear or nonlinear manifold. LPP shares some similar properties with LLE and LE, such as locality preserving. More crucially, LPP is defined everywhere in the ambient space rather than just on the training data, and so it has significant advantage over LLE and LE in locating and explaining new test data in the reduced subspace. LPP was shown to have superior discriminating power than PCA and LDA in face recognition [8].

In this paper, we investigate the appearance manifold of facial expression, which provides a unified framework for automatic facial expression analysis. We explore Locality Preserving Projections to learn the structure of the expression manifold from two kinds of feature space: raw image data and Local Binary Patterns (LBP) [16]. For manifolds of different subjects, we propose a novel alignment method to keep the semantic similarity of facial expression from different subjects on one generalized manifold (Section 3). We show in Section 4 the experimental results on the Cohn-Kanade Database [10]. Expression manifolds of 96 subjects are successfully aligned on the generalized manifold. Expression recognition performed on the generalized manifolds further demonstrate the effectiveness of the alignment method. Conclusions are drawn in Section 5.

2 Locality Preserving Projections (LPP)

The generic problem of linear dimensionality reduction is the following. Given a set $x_1, x_2, \ldots, x_m$ in R^n, find a transformation matrix W that maps these m points to $y_1, y_2, \ldots, y_m$ in $R^l (l \ll n)$, such that y_i represent x_i, where $y_i = W^T x_i$. Let $\mathbf{w}$ denote the transformation vector, the optimal projections preserving locality can be obtained by solving the following minimization problem [7]:

$$\min_{\mathbf{w}} \sum_{i,j} (\mathbf{w}^T x_i - \mathbf{w}^T x_j)^2 S_{ij} \qquad (1)$$

where S_{ij} evaluate the local structure of data space. It can be defined as follows:

$$S_{ij} = \begin{cases} e^{-\frac{\|x_i - x_j\|^2}{t}} & \text{if } x_i \text{ and } x_i \text{ are ``close''} \\ 0 & \text{otherwise} \end{cases} \tag{2}$$

or in a simpler form as

$$S_{ij} = \begin{cases} 1 & \text{if } x_i \text{ and } x_i \text{ are ``close''} \\ 0 & \text{otherwise} \end{cases} \tag{3}$$

where "close" can be defined by $\|x_i - x_j\|^2 < \epsilon$, or x_i is among k nearest neighbors of x_j or x_j is among k nearest neighbors of x_i. The objective function with the choice of symmetric weights $S_{ij}(S_{ij} = S_{ji})$ incurs a heavy penalty if neighboring points x_i and x_j are mapped far apart. Therefore, minimizing it is an attempt to ensure that if x_i and x_j are "close" then $y_i(= \mathbf{w}^T x_i)$ and $y_j(= \mathbf{w}^T x_j)$ are close as well. S_{ij} can be seen as a similarity measure between objects. The objective function can be reduced to:

$$\begin{aligned} \frac{1}{2}\sum_{ij}(\mathbf{w}^T x_i - \mathbf{w}^T x_j)^2 S_{ij} &= \sum_i \mathbf{w}^T x_i D_{ii} x_i^T \mathbf{w} - \sum_{ij} \mathbf{w}^T x_i S_{ij} x_j^T \mathbf{w} \\ &= \mathbf{w}^T X(D - S)X^T \mathbf{w} = \mathbf{w}^T X L X^T \mathbf{w} \end{aligned} \tag{4}$$

where $X = [x_1, x_2, \ldots, x_m]$ and D is a diagonal matrix whose entries are column (or row, since S is symmetric) sums of S, $D_{ii} = \sum_j S_{ji}$. $L = D - S$ is the Laplacian matrix. The bigger the value D_{ii} (corresponding to y_i) is, the more important is y_i. Therefore, a constraint is imposed as follows:

$$\mathbf{y}^T D\mathbf{y} = 1 \Rightarrow \mathbf{w}^T X D X^T \mathbf{w} = 1 \tag{5}$$

The transformation vector $\mathbf{w}$ that minimizes the objective function is given by the minimum eigenvalue solution to the generalized eigenvalue problem:

$$XLX^T\mathbf{w} = \lambda XDX^T\mathbf{w} \tag{6}$$

Note that the two matrices XLX^T and XDX^T are both symmetric and positive semi-definite. The obtained projections are actually the optimal linear approximation to the eigenfunctions of the Laplace Beltrami operator on the manifold [7]. Therefore, though it is still a linear technique, LPP recovers important aspects of the intrinsic nonlinear manifold structure by preserving local structure. A more detailed derivation and justifications of LPP can be found in [7].

By applying LPP to LBP appearance feature space, image sequences of facial expressions of an individual are mapped into the embedded space as shown in Fig 1. The embedded manifolds of another three subjects are shown in Fig 2. It is observed that image sequences representing basic expressions with increasing intensity become curves on the manifold extended from the center (neutral faces) to the typical expressions.

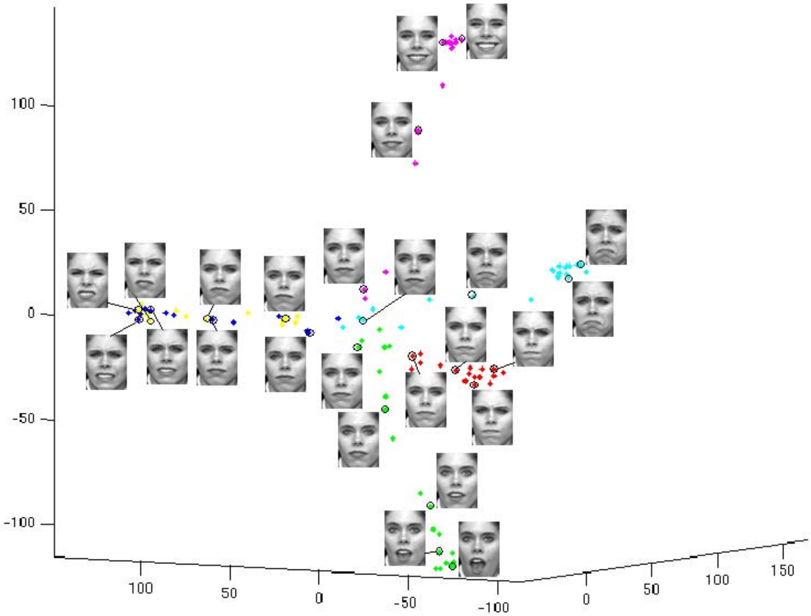


Fig. 1. Six image sequences of basic expressions of an individual mapped into the embedding space described by the first 3 coordinates of LPP. Different sequences are represented by different colors: red: Anger; yellow: Disgust; blue: Fear; magenta: Joy; cyan: Sadness; green: Surprise. (Note: the meaning of colors keeps same in all figures)

3 Alignment of Manifolds of Different Subjects

Image sequences of facial expressions of an individual makes a continuous manifold in the embedding space; however, due to significant appearance variation across different subjects, the manifolds of different subjects vary much in the covered regions and the stretching directions. Fig 3 shows the embedded manifold of image sequences from six subjects, which clearly shows that different subjects correspond to different clusters. Manifolds of different subjects should be aligned in a way that the images from different subjects with semantic similarity can be mapped to the near region. Chang et al [2] proposed a nonlinear method to align the manifolds of different subjects in the space of Lipschitz embedding. Their alignment method was evaluated on image sequences from two subjects. Here we propose a novel algorithm to align manifolds of different subjects in a global space, and verify its effectiveness on $O(10^2)$ subjects.

As shown in Fig 1, an image sequence representing facial expression with increasing intensity is embedded as a curve on the manifold, from the neutral face to the typical expression. If we define a global coordinate space, in which different typical expressions (including neutral faces and six basic expressions) from multiple subjects are well clustered and separated, the image sequences from different subjects with the same expression will be embedded as curves between the same two clusters: neutral faces and the typical expression. In this way, the manifolds of different subjects will be aligned on one generalized manifold.

We propose to define the global coordinate space based on images of typical expressions. For the data set containing images of typical expressions from different subjects, as appearance varies a lot cross different subjects, there is significant overlapping among different expression classes. Therefore, the original LPP, which performs in an unsupervised manner, fails to embed the data set in low dimensional space in which different expression classes are well clustered. Here we proposed a Supervised Locality Preserving Projections (SLPP) algorithm to solve the problem, which not only preserves local structure, but also encodes class information in the embedding. SLPP preserves class information when constructing the neighborhood graph. The local neighborhood of a sample x_i from class c should be composed of samples belonging to class c only. This can be achieved by increasing the distances between samples belonging to different classes, but leaving them unchanged if they are from the same class. Let $Dis(i, j)$ denote the distance between x_i and x_j, the distance after incorporating class information is defined as

$$SupDis(i, j) = Dis(i, j) + \alpha M \delta(i, j) \qquad \alpha \in [0, 1] \tag{7}$$

where $M = \max_{i,j} Dis(i, j)$, and $\delta(i, j) = 1$ if x_i and x_j belong to the same class, and 0 otherwise. SLPP introduces an additional parameter α to quantify the degree of supervised learning. When $\alpha = 0$, one obtains the unsupervised LPP; when $\alpha = 1$, the result is fully supervised LPP. For fully supervised LPP, distances between samples in different classes will be larger than the maximum distance in the entire data set; this means neighbors of a sample will always be picked from that class it belongs to. Varying α between 0 and 1 gives a partially supervised LPP, where a embedding is found by introducing some separation between classes. SLPP ($\alpha = 1$) is used in this paper. By preserving local structure of data belonging to the same class, SLPP obtains a subspace in which different image classes can be well separated.

By applying SLPP to the data set of images of typical expressions, a subspace is derived, in which different expression classes are well clustered and separated (as shown in Fig 5). The subspace provides global coordinates for the manifolds of different subjects, which are aligned on one generalized manifold. Image sequences representing facial expressions from beginning to apex are mapped on the generalized manifold as the curves from the neutral faces to the cluster of the typical expressions. For comparison, Fig 3 and Fig 6 show the unaligned manifolds and the aligned manifolds of six subjects. The generalized manifold map the images with sematic similarity but from different subjects in the near region; so it provides a unified framework for automatic facial expression analysis.

4 Experiments

The optimal data set for expression manifold learning should contain $O(10^2)$ subjects, and each subject has $O(10^3)$ images that cover basic expressions. However, until now, there is no such database that can meet this requirement. Chang et al [2, 3] conducted experiments on a small data set builded themselves, e.g., only two subjects (one male and one female) were used in [2]. Here we conduct experiments on the Cohn-Kanade database [10] which consists of 100 university students in age from 18 to 30 years, though each subject only has several tens frames of basic expressions. Image sequences from neutral to target expression were captured, and the duration of the expression varied. In our experiments, 316 image sequences (5,876 images in total) of basic expressions were selected from the database, which come from 96 subjects, with 1 to 6 emotions per subject.

Following Tian [18], we normalized the faces to a fixed distance between the centers of the two eyes. Facial images of 110×150 pixels were cropped from original frames based on the two eyes location. No further alignment of facial features such as alignment of mouth, or remove of illumination changes [18] were performed in our experiments. So variations due to illumination, and pose exist in our data. In [2], Active Wavelets Networks were applied on the image sequences to reduce these variations.

Two kinds of appearance features were used: raw image data (IMG) and Local Binary Patterns (LBP). LBP was proposed originally for texture analysis [11]. Face images can be seen as a composition of micro-patterns which can be effectively described by the LBP features. In our previous research [15, 16], LBP features were shown to be effective and efficient for facial expression analysis. Each face image was represented by a LBP histogram with length of $2,478$ (see [16] for details). When considering IMG features, for computational efficiency, we down-sampled face images to 55×75 pixels, and represented each image with a $4,125$-dimensional vector.

Appearance Manifold of Facial Expression. We selected six subjects from the data set, each of which has six image sequences corresponding to six basic expressions. At first, we applied LPP to image sequences of each subject respectively to learn the expression manifold of each subject. 3-D visualization of the embedded manifold in LBP feature space of one subject is shown in Fig 1. Due to limitation of space, we only show the embedded manifolds of another three subjects in Fig 2. It is observed that images of facial expressions of an individual were embedded as a smooth manifold, and every image sequence is mapped to a curve on the manifold that begins from the neutral face and extends in distinctive direction with varying intensity of expression.

Next we applied LPP to image sequences of all six subjects, and 3-D visualization of the embedded manifold are shown in Fig 3. It is observed that there are six clusters in the embedded space, and image sequences of different subjects are mapped to different regions. As said above, due to the significant appearance variation across subjects, it is very hard for LPP to keep images with similar expression but from different subjects in the near region on the manifold. Fig 4 shows the 3-D embedded manifolds of all image sequences from 96 subjects, which consists of many manifolds with different centers (neutral faces), covering regions and stretching directions.

Alignment of Appearance Manifolds. We selected one neutral face and three peak frames (during the apex of expression) of every sequence to build a

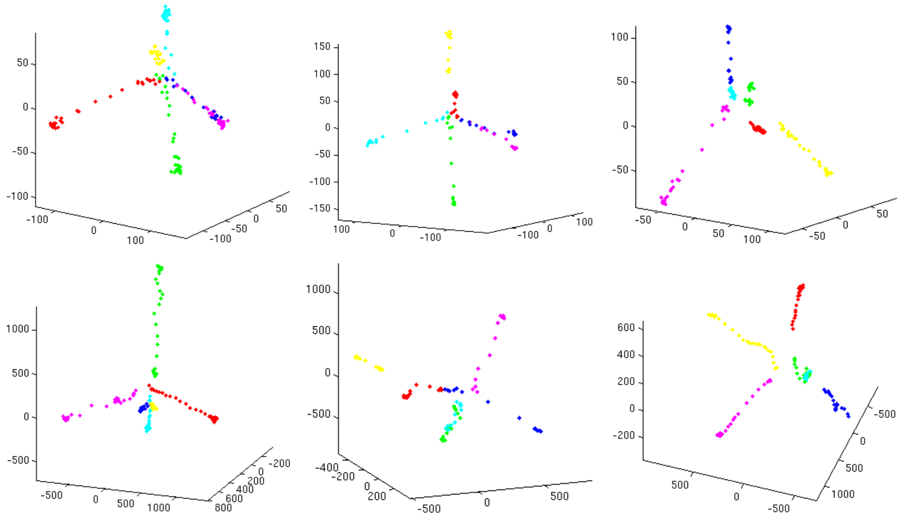


Fig. 2. 3-D visualization of expression manifolds of three subjects (from left to right). The first row: LBP; the second row: IMG

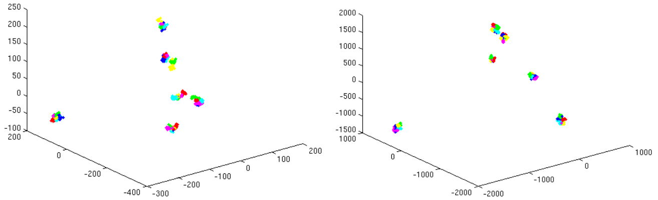


Fig. 3. Image sequences of six subjects mapped into the embedding space described by the first three coordinates of LPP. Left: LBP; Right: IMG

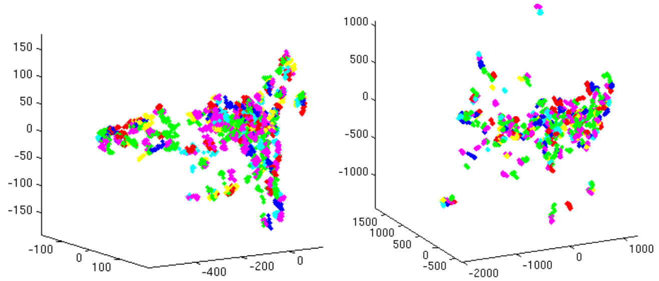


Fig. 4. Image sequences of 96 subjects mapped into the embedding space described by the first three coordinates of LPP. Left: LBP; Right: IMG

data set that consists of images of 7-class basic expressions. The Supervised LPP was explored to embed the data set to a subspace as shown in Fig 5. Different expressions were well clustered and separated in the subspace. It is also observed that different expression classes are better separated with LBP features. The

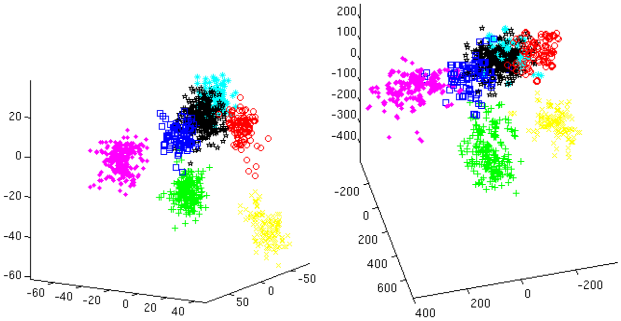


Fig. 5. Images of typical facial expressions mapped into the embedding space described by the first three coordinates of SLPP. Left: LBP; Right: IMG

distributions obtained reflect the human observation that Joy and Surprise can be clearly separated, but Anger, Disgust, Fear and Sadness are easily confused. In many existing research such as [4, 18], most confusions also come from Anger, Disgust, Fear and Sadness.

The subspace derived by SLPP provides global coordinates for the manifolds of different subjects. Fig 6 plots appearance manifolds of the six subjects in the global space, which are successfully aligned on one generalized manifold. The manifolds of 96 subjects are also aligned on the generalized manifold as shown in Fig 7. We can conclude that the images with semantic similarity but from different subjects are successfully embedded in the near region in the global space. A supplementary video[1] demonstrates image sequences of different subjects are embedded on the generalized manifold.

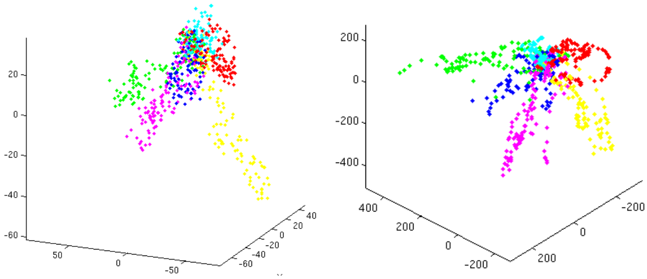


Fig. 6. The aligned manifolds of the six subjects. Left: LBP; Right: IMG

The global space is learned from images of typical basic expressions. So it is simple and easy to implement. Although only image sequences of basic expressions are discussed until now, the generalized appearance manifold provides a global semantic representation for all possible facial expressions. For example, the blends of expression will lie between the curves of basic expressions, so can

[1] Available at *http://www.dcs.qmul.ac.uk/˜cfshan/demos/manifold_align.avi*

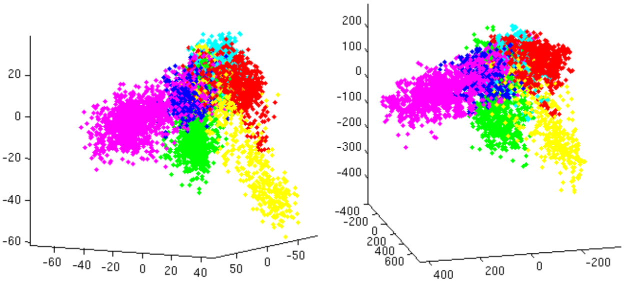


Fig. 7. The aligned manifolds of 96 subjects. Left: LBP; Right: IMG

be analytically analyzed based on the basic curves. Intensity of expression can also be defined easily on the generalized manifold. Therefore, the analysis of facial expression will be facilitated on the generalized manifold.

Facial Expression Recognition. Following Chang et al [2], we applied a k-Nearest Neighbor method to classify expressions on the aligned expression manifold. Since there is no clear boundary between neutral face and the expression of a sequence, we manually labelled neutral faces, which introduced noise in our recognition. The recognition results are presented in Table 1. The experimental results further demonstrate the effectiveness of our alignment method.

Table 1. Expression recognition results on the generalized appearance manifold of facial expression

	k-NN ($k = 9$)	k-NN ($k = 11$)	k-NN ($k = 13$)
IMG	92.04%	91.27%	89.98%
LBP	90.71%	90.79%	90.67%

5 Conclusions

This paper investigates the appearance manifold of facial expression, which provide a general framework for automatic facial expression analysis. Locality Preserving Projections (LPP) is explored to learn expression manifolds from two kinds of feature space: raw image data and Local Binary Patterns (LBP). For manifolds of different subjects, we propose a novel alignment algorithm by learning a global space from images of typical expressions. The semantic similarity of facial expression from different subject is well kept on the generalized manifold. Extensive experiments on 96 subjects from the Cohn-Kanade database illustrate the effectiveness of the alignment algorithm.

References

1. M. Belkin and P. Niyogi. Laplacian eigenmaps and spectral techniques for embedding and clustering. In *NIPS*, 2001.

2. Y. Chang, C. Hu, and M. Turk. Mainfold of facial expression. In *IEEE International Workshop on Analysis and Modeling of Faces and Gestures*, 2003.
3. Y. Chang, C. Hu, and M. Turk. Probabilistic expression analysis on manifolds. In *CVPR*, 2004.
4. I. Cohen, N. Sebe, A. Garg, L. Chen, and T. S. Huang. Facial expression recognition from video sequences: Temporal and static modeling. *CVIU*, 2003.
5. B. Fasel and J. Luettin. Automatic facial expression analysis: a survey. *Pattern Recognition*, 36:259–275, 2003.
6. D. Fidaleo and M. Trivedi. Manifold analysis of facial gestures for face recognition. In *ACM SIGMM Multimedia Biometrics Methods and Application Workshop*, 2003.
7. X. He and P. Niyogi. Locality preserving projections. In *NIPS*, 2003.
8. X. He, S. Yan, Y. Hu, P. Niyogi, and H. Zhang. Face recognition using laplacianfaces. *IEEE PAMI*, 27(3):328–340, Mar 2005.
9. C. Hu, Y. Chang, R. Feris, and M. Turk. Manifold based analysis of facial expression. In *CVPR Workshop on Face Processing in Video*, 2004.
10. T. Kanade, J.F. Cohn, and Y. Tian. Comprehensive database for facial expression analysis. In *IEEE FG*, 2000.
11. T. Ojala, M. Pietikäinen, and T. Mäenpää. Multiresolution gray-scale and rotation invariant texture classification with local binary patterns. *IEEE PAMI*, 2002.
12. M. Pantic and L. Rothkrantz. Automatic analysis of facial expressions: the state of art. *IEEE PAMI*, 22(12):1424–1445, 2000.
13. M. Pantic and L. Rothkrantz. Toward an affect-sensitive multimodal human-computer interaction. In *Proceeding of the IEEE*, 2003.
14. L. K. Saul and S. T. Roweis. Think globally, fit locally: Unsupervised learning of low dimensional manifolds. *Journal of Machine Learning Research*, 2003.
15. C. Shan, S. Gong, and P. W. McOwan. Conditional Mutual Information Based Boosting for Facial Expression Recognition. In *BMVC*, 2005.
16. C. Shan, S. Gong, and P. W. McOwan. Robust facial expression recgnition using local binary patterns. In *IEEE ICIP*, 2005.
17. J. B. Tenenbaum, V. Silva, and J. C. Langford. A global geometric framework for nonlinear dimensionality reduction. *Science*, 290, Dec 2000.
18. Y. Tian. Evaluation of face resolution for expression analysis. In *CVPR Workshop on Face Processing in Video*, 2004.
19. Y. Tian, T. Kanade, and J. Cohn. Recognizing action units for facial expression analysis. *IEEE PAMI*, 23(2), 2001.
20. Y. Tian, T. Kanade, and J. Cohn. Facial Expression Analysis, *Handbook of Face Recognition*. Springer, 2003.

Author Index