The Structure of the Digitised Line:
With Applications to
Line Drawing and Ray Tracing
in Computer Graphics

Thesis submitted by

Peter David STEPHENSON

BSc(Hons) *JCUNQ* CertSciComm *ANU*

in May 1998

for the degree of Doctor of Philosophy

in the Department of Computer Science

James Cook University of North Queensland

JAMES COOK UNIVERSITY

North Queensland • Australia

To Karin,

For Dora,

With Honor.

Abstract

The structure of the digitised line $y = \alpha x + \beta$ where $\alpha, \beta \in \mathcal{R}$ is described by considering the structure of the hierarchy of runs within the pattern of the intersection of the line and the lattice. The definition of a run is extended to include pixels and runs of runs and the properties of runs within different levels of the hierarchy are discussed. The definition of higher order runs differs from those previously used in the structural techniques developed in digital image analysis as does some of the parameters used to characterise the hierarchy. The definition allows for an efficient conditional style algorithm for reproducing the run structure of digital line at any level within the hierarchy. The relationship between the hierarchical run structure of the digitised line, the least remainder gcd algorithm and continued fractions is fully explained.

The theoretical results pertaining to a line with rational slope and intercept are applied to the application of two-dimensional line-drawing. An algorithm based on runs of runs is presented which outperforms existing techniques except for very short lines. The theory serves to unify line drawing algorithms such as Bresenham's pixel–based and run length slice algorithms. Suggestions are made to an algorithm which adjusts the level used of the run hierarchy based on the characteristics of the line being drawn and to extending the `setPixel` command to allow for higher order patterns within the line to be set.

The theoretical results are applied to the application of ray tracing a two-dimensional uniform spatial subdivision. An algorithm based on runs of runs is presented which outperforms the DDA algorithm. Initial results are also presented for three-dimensional ray tracing. Suggestions are made as to a spatial subdivision scheme for

ray tracing which fully utilises the run–based structure of the intersection pattern.

To answer questions regarding the similarity of two digitised lines, a pattern matching technique is presented. The technique is used to show the necessity and sufficiency of the run–based definition of the digitised line we present and the containment of one digitised line within another.

Preface

Life in a lattice means you are never far away from number theory and, in describing the structure of the digital line, most of the arguments we will be presenting have a mathematical nature. The lemma, theorem, proof, corollary structure we will be using is not prevalent in literature regarding computer graphics. However, it has provided the simplest and most powerful means for us to argue our case. While a picture may paint a thousand words, volumes are written about equations.

We have endeavoured to keep the proofs to our results as short as possible while keeping the argument easy to follow. Some proofs however are lengthy and algebraically tedious but are required for completeness. Upon your first reading, it is recommended that you delay your interest in the reasoning and concentrate on the main results.

In Chapter 4 we describe the links between the run structure of a line with rational slope, Euclid's gcd algorithm, the least remainder gcd algorithm and continued fractions. To assist the reader who is not familiar with number theory, we have included some background to these algorithms. For those well-read, you may well want to take only a cursory look at sections 4.2 and 4.3 to familiarise yourself with the notation.

To assist the reader, Appendix D contains a glossary of symbols and Appendix C contains a summary of the main results of Chapters 2 and 3 which are used throughout the dissertation.

Acknowledgements

First and foremost, I must thank my supervisor, Dr. Bruce Litow whose guidance, interest, time and advice have been crucial and freely provided throughout the preparation of this work. The initial idea sprang from him when he asked me a question I could not answer. An event which repeated itself many times. Above all, he has taught me that to solve a problem you should look for the right question and not simply the right answer.

Thanks to Prof. Jack Bresenham, Assoc. Prof. Xiaolin Wu and Prof. Azriel Rosenfeld for their time, comments and criticisms in the refereeing of this dissertation. Thanks also to Prof. Jack Bresenham during my visit To Winthrop College for his comments on our line drawing work. And to Dr. Alan Watt at Sheffield University and the staff at Griffith University, the University of Sydney and the Australian National University for their kind support during my visits. To my teachers, students and colleages at James Cook University of North Queensland, thanks for the many experiences and memories I have had over the past fifteen years.

I must thank my family, Philip, Nicola, Natalie, Barry and now little Emma, for their support over the past few years, especially my mother, Honor. And to Karin...

Statement of Access

I, the undersigned, the author of this thesis, understand that James Cook University of North Queensland will make it available for use within the University Library and, by microfilm or other means, allow access to users in other approved libraries. All users consulting this thesis will have to sign the following statement:

> In consulting this thesis I agree not to copy or closely paraphrase it in whole or in part without the written consent of the author; and to make proper public written acknowledgment for any assistance which I have obtained from it.

Beyond this, I do not wish to place any restriction on access to this thesis.

....

Peter Stephenson

October 23, 1998

Statement of Sources

I declare that this thesis is my own work and has not been submitted in any form for another degree or diploma at any university or other institution of tertiary education. Information derived from the published or unpublished work of others has been acknowledged in the text and a list of references is given.

....

Peter Stephenson

October 23, 1998

Contents

1 Introduction **1**

1.1 Describing the Digitised Line 2

1.1.1 Connectivity 3

1.1.2 Representing the Digitised Line 4

1.1.3 Symmetry and Special Cases 6

1.2 The Use of Lines in Computer Graphics 7

1.2.1 Line Drawing 7

1.2.2 Ray Tracing 12

1.3 The Structure of the Digital Line. 14

1.3.1 When Does a Digital Arc Constitute a Line? 15

1.3.2 The Structure of Runs within the Digital Line. 16

1.4 Introducing the Run–Based Structure of the Digitised Line 17

1.4.1 Generating the Intersection Pattern 23

2 The Intersection Pattern of a Line with Rational Slope **27**

2.1 Introduction 27

2.2 The Structure of the Intersection Pattern 30

2.2.1 Extending the Definition of a Run 30

2.2.2 The Structural Run Parameters 34

2.2.3 Type and Shape 37

2.3 The Necessity and Sufficiency of the Run Parameters 39

2.4 Generating the Intersection Pattern 48

2.4.1 Generating the Numerator Sequence 49

2.4.2 Generating the Intersection Pattern from the Numerator Sequence 55

2.5 The Calculation of the Run Parameters and Numerator Sequence . . 58

2.5.1 Calculating the Run Parameters 58

2.5.2 Calculating the Numerator Sequence and Intersection Pattern 59

2.6 Examples 61

2.6.1 The Example of $y = \frac{17}{41}x$. 62

2.6.2 The Example of $y = \frac{29}{41}x$. 63

2.7 Conclusions 63

3 The Intersection Pattern of a Line with a Real Slope and Nonzero Intercept 65

3.1 Introduction 65

3.2 The Effect of a Non–Zero Intercept on a Line with Rational Slope . . 66

3.2.1 The Initial Partial Run Length 71

3.2.2 The Initial Numerator Value 73

3.2.3 The Final Partial Run Length 76

3.3 Lines with Irrational Slope 77

3.3.1 Zero Intercept 77

3.3.2 Non–Zero Intercept 79

3.3.3 Corner Intersections 82

3.4 Conclusions 83

3.4.1 The Sensitivity of the Digital Line to Changes in Slope and Intercept 83

4 The Greatest Common Divisor, Continued Fractions and the Run Parameters 87

4.1 The Link Between Number Theory and the Structure of Runs 88

4.2 The Greatest Common Divisor (GCD) 95

4.2.1 Euclid's Recursive GCD Algorithm 96

4.2.2 Euclid's Subtractive GCD Algorithm 96

4.2.3 The Least Remainder GCD Algorithm 97

4.3 Continued Fractions 98

4.3.1 Ordinary and Regular Continued Fractions 99

4.3.2 Calculating the Regular Continued Fraction Expansion of a Real Number 100

4.3.3 Convergence 100

4.3.4 Euclid's Recursive Algorithm and Regular Continued Fractions 101

4.4 The Least Remainder GCD Algorithm and the Run Parameters . . . 102

4.5 Continued Fractions and the Run Parameters 104

4.5.1 The Continued Fraction Expansion of a Rational Slope 104

4.5.2 The Continued Fraction Expansion of an Irrational Slope . . . 109

4.6 Applications 111

4.6.1 The GCD Properties of the Run Parameters 111

4.6.2 The Depth of the Hierarchy of Runs 112

4.6.3 Equivalence of Intersection Patterns 112

4.7 Conclusions 117

5 Application to Line Drawing in Computer Graphics 119

5.1 Introduction 119

5.2 Line Drawing Algorithms based on the Numerator Sequence 121

5.2.1 The Order 0 Algorithm 121

5.2.2 The Order 1 Algorithm 122

5.2.3 Special Cases 127

5.2.4 The Order 2 Algorithm 129

5.3 The Recursive Run–Based Line Drawing Algorithm 136

5.4 Experimental Results And Analysis 140

5.4.1 The Order 0 Algorithm 149

5.4.2 The Order 1 Algorithm 150

5.4.3 The Order 2 Algorithm 151

5.4.4 The Run Length Slice Algorithm 152

5.4.5 The Recursive Run–Based Algorithm 155

5.5 Conclusions 156

6 Application to Ray Tracing in Computer Graphics **158**

6.1 Introduction 158

6.2 Ray Tracing a Two-Dimensional Line 160

6.2.1 Tracing a Ray with Rational Slope 163

6.2.2 Tracing a Ray with Irrational Slope 167

6.3 Ray Tracing a Three-Dimensional Line 169

6.4 Experimental Results and Analysis 170

6.4.1 The Two-Dimensional Ray Tracing Algorithm 170

6.4.2 The Three-Dimensional Ray Tracing Algorithm 171

6.5 Conclusion 174

6.5.1 Use in Nonuniform Spatial Subdivision Techniques 174

7 Conclusions **177**

7.1 Future Work 179

7.1.1 Parallelisation 179

7.1.2 The Properties of the Numerator Sequence 179

7.1.3 Line Drawing 180

7.1.4 Ray Tracing 180

7.1.5 Higher-Dimensional Objects 181

A Matching Patterns of Runs **182**

B The Correspondance between the Run-Based and Bresenham's Line Drawing Algorithms. **193**

B.1 Bresenham's Line Drawing Algorithm 193

B.2 Bresenham's Run Length Slice Algorithm 194

B.3 Analysis 201

B.3.1 Run Length Slice Algorithm 201

B.3.2 Adapted Run Length Slice Algorithm 202

C Summary. 203

D Glossary. 206

List of Figures

1.1 Possible abstractions of pixel space by a lattice. 3

1.2 The connectivity of equivalent intersection patterns. 4

1.3 Chain code representation of incremental pixel movements. 5

1.4 Chain code representation of the digital line. 6

1.5 The 8–way symmetry of a point defined in Cartesian coordinates. . . 7

1.6 The intersection pattern of the line of best-fit from pixel (0, 0) to (11, 7). 8

1.7 The intersection pattern of the line of intersection $y = \frac{7}{11}x$. 13

1.8 The intersection pattern of the line $y = \frac{17}{41}x$. 19

1.9 The construction of order 2 runs in the line $y = \frac{17}{41}x$. 21

1.10 The construction of order 3 runs in the line $y = \frac{17}{41}x$. 22

1.11 The intercept geometry of the j-th run of order 1 in the line. 24

2.1 The length of the first order 1 run in the intersection pattern. 31

2.2 The intercept geometry of a run of order i. 49

2.3 The construction of runs and the numerator sequence of all orders in the line $y = \frac{17}{41}x$. 62

2.4 The construction of runs and the numerator sequence of all orders in the line $y = \frac{29}{41}x$. 63

3.1 The effect of a non-zero intercept, $\beta = \frac{23}{41}$, on the intersection pattern of a line with rational slope, $\ell : y = \frac{17}{41}x$. 69

3.2 The geometry of a near corner intersection of a line with irrational slope and intercept. 79

3.3 The spirograph corresponding to the line $\ell : y = \frac{3}{10}x + \beta$. 84

3.4 The bounds to changes of the slope of the line $\ell : y = \frac{3}{10}x$. 85

4.1 The intersection patterns of the lines $\ell' : y = \frac{a'}{d'}$ contained in the line $\ell : y = \frac{29}{41}x$ where $a' \leq 29$ and $d' \leq 41$. 114

5.1 Possible abstractions of pixel space by a lattice. 119

5.2 Translation of the line for use in the line drawing algorithm. 120

5.3 The dependence of execution speed of the scan conversion of the line drawing algorithms on the slope of the line. 144

5.4 The dependence of execution speed of the scan conversion of the line drawing algorithms on the length of the line. 145

5.5 The dependence of execution speed of the line drawing algorithms on the slope of the line. 146

5.6 The dependence of execution speed of the line drawing algorithms on the length of the line. 147

5.7 The number of runs of order i in the intersection pattern of the line $y = \frac{a}{d}x$ where $0 < a < d$ and $d = 1024$. 152

5.8 The number of runs length slices in the intersection pattern of the line $y = \frac{a}{d}x$ where $0 < a < d$ and $d = 1024$. 154

6.1 Spatial subdivision types. 159

6.2 The intersection pattern of $\ell' : y = \frac{17}{41}x + \frac{23}{41}$. 164

6.3 Step 1: The initial partial run length and initial numerator value of order 1. 164

6.4 Step 2: The initial partial run length and initial numerator value of order 2. 165

6.5 Step 3: Tracing the full length runs until a numerator value of 0. . . . 166

6.6 Step 4: The corner intersection is detected and the remaining full length runs are traced. 167

6.7 Execution speed in cpu seconds of the run–based algorithm and the DDA algorithm implemented using fixed point arithmetic for the intersection of the line $y = \alpha x + \beta$ and the two-dimensional lattice of size (f, f). 171

6.8 Average execution speed in cpu seconds of the run–based algorithm and the DDA algorithm implemented using fixed point arithmetic for the intersection of the line $y = \alpha x + \beta$ and the two-dimensional lattice of size (f, f). 172

6.9 Execution speed in cpu seconds of the run–based algorithm of order 1 and the 3D-DDA algorithm implemented using fixed point arithmetic for the intersection of the line $y = \alpha x$ and the three-dimensional lattice of size (f, f, f). 173

6.10 A run–based nonuniform spatial subdivision scheme. 176

List of Tables

4.1 Calculation of the chain code of the line $\ell : y = \frac{16}{131}x$. 92

4.2 Transforming the chain code for slope 17/131. 94

4.3 The run parameters of the lines ℓ' that produce similar intersection patterns to ℓ. 115

5.1 Operator counts for the order 0 run–based algorithm. 149

5.2 Operator counts for the order 1 run–based algorithm. 150

5.3 Operator counts for the order 2 run–based algorithm. 152

B.1 Operator counts for the run length slice algorithm. 201

B.2 Operator counts for the adapted run length slice algorithm. 202

Definitions

2.1.1 Run length, what is long, l, and short, s. 28

2.2.1 The run, its order and shape. 30

2.2.2 Rosenfeld's Chord Property. 31

2.2.3 Type, $t^{[i]}$. 34

2.2.4 The structural run parameters, $p^{[i]}$, $m^{[i]}$ and $n^{[i]}$. 34

2.2.5 Shape, $s^{[i]}$. 39

2.2.6 Composition, $c^{[i]}$. 39

2.4.1 The run sequence, intercept sequence and numerator sequence. 48

3.2.1 The initial and final partial run lengths. 70

2.3.1 The length of an intersection pattern. 40

2.3.2 The containment of an intersection pattern within another. 40

4.2.1 The divisor of an integer. 95

4.2.2 The common divisor of two integers. 95

4.2.3 The greatest common divisor. 95

4.3.1 The continued fraction. 98

Theorems

2.2.1 In the intersection pattern of a line with rational slope, there can be only two run lengths of order 1, which are consecutive integers. 31

2.2.2 In the intersection pattern of a line with rational slope, there can be only two run lengths of order i, which are consecutive integers. 31

2.2.3 For order i, between any two of one of the possible run lengths, there must be at least one run of the other. 32

2.2.4 The length of a short run of order $i > 0$ is, $r^{[i]} = \lfloor p^{[i-1]}/p^{[i]} \rfloor$. 33

2.2.6 The relationship between the structural run parameters, $p^{[i]}$, $m^{[i]}$, $n^{[i]}$ and type, $t^{[i]}$. 35

2.2.10 The relationship between type, $t^{[i]}$, and shape, $s^{[i]}$. 37

2.4.2 A sequential and parallel method for generating the numerator sequence based on modular arithmetic. 51

2.4.6 A sequential method for generating the numerator sequence based on additive integer arithmetic. 54

2.4.7 A method for generating the intersection pattern of a line with rational slope, based on two consecutive elements of the numerator sequence. 55

2.4.8 A method for generating the intersection pattern of a line with rational slope, based on single elements of the numerator sequence. 56

3.2.1 Necessary and sufficient conditions for the equivalence of the intersection patterns generated by two lines with rational slope and rational and real intercepts. 67

3.2.3 Calculation of the initial partial run length. 71

3.2.4 Calculation of the initial numerator value using modular arithmetic. 73

3.2.5 Calculation of the initial numerator value using additive and multiplicative integer arithmetic. 74

3.2.6 Calculation of the final run length. 76

2.3.6 A necessary condition for the equivalence of the intersection patterns generated by two lines with rational slope. 47

3.3.1 A necessary condition for the equivalence of the intersection patterns generated by lines with rational and irrational slope. 77

3.3.2 The use of continued fractions to approximate a line with irrational slope by a line with rational slope. 78

3.3.5 A necessary condition for the equivalence of the intersection patterns generated by a line with irrational slope and intercept and a line with rational slope and intercept. 81

4.2.1 The GCD Recursion Theorem. 96

4.5.1 An alteration to the continued fraction algorithm to calculate the possible run lengths of all orders in the intersection pattern of a line by calculating the continued fraction expansion of its rational slope. 104

5.4.3 The time complexity of the order 0 run–based line drawing algorithm. . 149

5.4.2 The time complexity of the order 1 run–based line drawing algorithm. . 150

5.4.3 The time complexity of the order 2 run–based line drawing algorithm. . 151

5.4.4 The time complexity of the run length slice line drawing algorithm. 153

5.4.5 The time complexity of the recursive run–based line drawing algorithm. 155

Algorithms

2.1 Calculating the run parameters for all orders of the line $y = \frac{a}{d}x$. 58

2.2 Calculating the run parameters for all orders of the line $y = \frac{a}{d}x$. 59

2.3 Calculating the numerator sequence for all orders and the intersection pattern $\mathcal{P}_\ell$ of the line $y = \frac{a}{d}x$. 60

2.4 Calculating the numerator sequence for all orders and the intersection pattern $\mathcal{P}_\ell$ of the line $y = \frac{a}{d}x$. 61

4.1 Euclid's recursive gcd algorithm 96

4.2 The subtractive gcd algorithm 97

4.3 The least remainder gcd algorithm 97

4.4 The least remainder gcd algorithm in terms of the run parameters 103

4.5 Extending the continued fractions algorithm to calculate the run parameters for all orders. 108

4.6 Extending the continued fractions algorithm to calculate the rational approximation of the slope for the frame (f, f) and the run parameters associated with the approximation. 110

5.1 The order 0 line drawing algorithm. 122

5.2 The order 1 line drawing algorithm. 126

5.3 The order 2 line drawing algorithm 133

6.1 The two-dimensional run–based ray tracing algorithm. 161

B.1 The order 0 line drawing algorithm 193

B.2 Bresenham's line drawing algorithm 194

B.3 The run length slice line drawing algorithm 195

B.4 The adapted run length slice line drawing algorithm 199

Chapter 1

Introduction

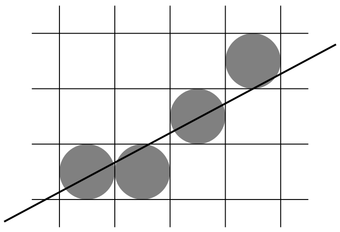
The digitised line constitutes a fundamental area of research in computer science, especially within the fields of computer graphics and digital geometry where, with the exception of the pixel, the line is the simplest geometric object. Due to the importance of the digitised line in a number of differing areas of computer science, the development of research on the problem has followed two main distinct and separate paths [22]. In the fields related to digital geometry and in particular digital image understanding, the majority of work has concentrated on developing structural techniques to investigate the properties and structure of the digitised line. For computer graphics, the emphasis has been on developing efficient conditional algorithms primarily for use in line drawing and ray tracing applications. The use of the structural information developed for techniques in computer graphics has held much promise, however what has been lacking to date is a clear connection between the structure of the digitised line and the conditional techniques used in computer graphics.

In this dissertation, we present a structural definition of the digitised line based on the hierarchy of runs in the pattern of lattice cells of the intersection of the line and the lattice. We describe how the digitised line can be generated using a conditional style algorithm at any level within the hierarchy thus bridging the gap between the structural and conditional descriptions. The description of the hierarchy of runs and the conditional run–based algorithm provides a way of unifying line drawing algorithms such as Bresenham's pixel–based line drawing algorithm and run length slice algorithm and developing even more time–efficient algorithms. We extend the definition of a run in a natural way to include the definition of a pixel (lattice cell) and the recursive definition of runs of runs. The connection between the structure of runs in the digitised line, the least remainder gcd algorithm and continued fractions are described. This connection allows for continued fractions to be considered from a solely geometric viewpoint and forms a more complete description than those offered before. It also allows for the rational based approximation of lines with irrational slope and real intercept. This result is applied to developed a purely integer–based algorithm for ray tracing a uniform spatial subdivision where the use of structural features of the intersection pattern of the lattice and a ray have not been used before.

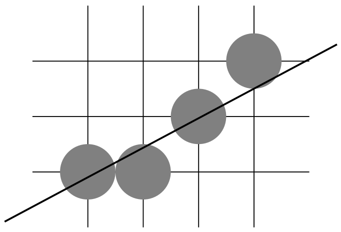
1.1 Describing the Digitised Line

Digitisation is the process of describing an object by replacing segments of the object by a basic unit or digit. Due to the digital nature of the information stored, represented and displayed by computer machinery, the problems of digitisation are at the heart of the computer sciences. This is particularly true for raster–based computer graphics where every display or reproduction medium used to present images is based on the unit of the pixel (picture element). For representation in the raster, the process of digitisation, that is converting an object into a set of pixels, is called *scan conversion*.

Although alternative raster structures are available such as the hexagonal or triangular based raster, the predominant form of the raster is rectilinear. The reason for this choice of geometry lies in the simplicity of mapping the raster to the linear structure of memory and the ease of implementation of the hardware and software appliances for representing and manipulating the raster. Both of these features of the rectilinear raster emanate from the abstraction of the raster by the two–dimensional lattice, $\mathcal{L}$, defined as the two–dimensional domain of integers, $\mathcal{Z} \times \mathcal{Z}$. Two possibilities exist to map the pixel space to the lattice. Each pixel can be defined by a lattice position or a lattice cell as illustrated in Figure 1.1.



(a) A pixel defined as the interior of a cell.



(b) A pixel defined at a lattice position.

Figure 1.1: Possible abstractions of pixel space by a lattice.

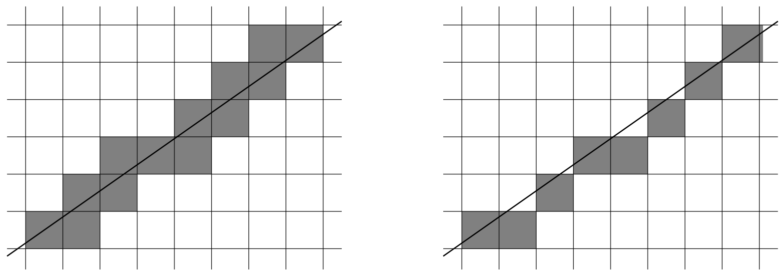
Due to the limited size of the raster and the spatial subdivisions used in computer graphics, we will often be considering only an axis–aligned rectagonal section of the lattice of finite size, refered to as a frame $\mathcal{L}_{(f_x,f_y)}$. The size of the frame is defined by the lattice points bounded by and including the diagonally opposite corners of the rectagonal section, $(0,0)$ to (f_x, f_y).

We will discuss the digitisation of the line in the raster and the tracing of a ray through a uniform spatial subdivision by considering the pattern of lattice cells intersected by the line called the *intersection pattern*. We will start by describing some relevant properties and representations of the intersection pattern.

1.1.1 Connectivity

The connectivity of the intersection pattern can be defined in one of two ways, 4–connected or 8–connected. A region is 4–connected if each pixel (x, y) has a neighbor from the set $\{(x+1, y), (x, y+1), (x-1, y), (x, y-1)\}$ and 8–connected if each pixel (x, y) has a neighbor from the set $\{(x+1, y), (x+1, y+1), (x, y+1), (x-1, y+1), (x-1, y), (x-1, y-1), (x, y-1), (x+1, y-1)\}$. Hence a 4–connected region is also 8–connected.

For the applications in computer graphics we will be considering, line drawing and ray tracing, both forms of connectivity are apparent. In line drawing where antialiasing techniques are not applied, the set of pixels which represent the line is 8–connected. For ray tracing a uniform spatial subdivision, the set of cells which



(a) 4-connected (b) 8-connected

Figure 1.2: The connectivity of equivalent intersection patterns.

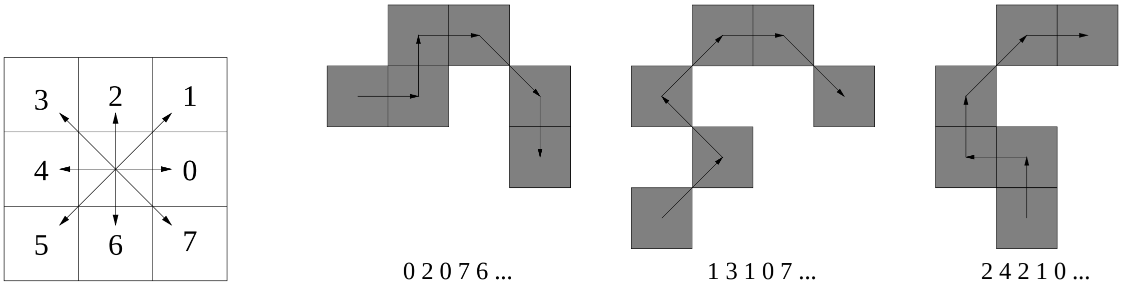
is pierced by a ray is 4–connected if corner intersections can be ignored. However, whether the line represents a line to be drawn or a ray and the lattice represents a raster or a spatial subdivision, the 8–connected and 4–connected representations of the line are equivalent as each is a sub or superset of and can be derived from the other.

We will considering in depth how to represent and calculate the 8–connected set of lattice cells which represents the intersection pattern. We will be working with the 8–connected intersection pattern such that the line intersects the left-side of each cell in the intersection pattern at the point y where $0 \leq y < 1$. The underlying reason for this choice is that the number of pixels in the digitised line equals the size of the bounding frame along the X–axis. The number of runs in the line equals the size of the bounding frame along the Y–axis. This correspondance forms the basis of the simplicity of our description of the hierarchy of runs within the digitised line and the descriptions which have preceded ours.

1.1.2 Representing the Digitised Line

Freeman [55, 56] introduced the chain code [29, 87, 141, 74, 134] as a technique for representing 8–connected lines and arcs. The chain code of the line, $c_j \in \{0, 1, \ldots, 7\}$, is the sequence of incremental pixel movements to a neighbor where the neighboring pixel is designated by a number.

Freeman described a number of properties of the chain code representation.



(a) Definition. (b) An example chain code rotated by $\frac{\pi}{4}$ and $\frac{\pi}{2}$.

Figure 1.3: Chain code representation of incremental pixel movements.

1. Rotation by $\frac{\pi}{4}$ or 45°can be achieved by

$$c'_j \equiv c_j + i \pmod 8 \qquad \text{counter clockwise} \tag{1.1}$$

$$c'_j \equiv c_j - i \pmod 8 \qquad \text{clockwise} \tag{1.2}$$

   however rotations by $\frac{i\pi}{4}$ where i is odd cause a distortion in the rotated image due to the rectangular division of the space. This can be seen in Figure 1.3(b) which shows a chain code rotated by $\frac{\pi}{4}$ (45°) and $\frac{\pi}{2}$ (90°).

2. Each code in the chain code of a digitised line belongs to the set $c_j \in \{c, c+1 \pmod 8\}$ such that $c \in \{0, \ldots, 7\}$. That is, there can be at most two values for each code in the chain. If there are two possible values, they differ by at most 1 modulo 8. A chain code consisting of one repeated code value represents a line at angle $0, \frac{\pi}{4}, \frac{\pi}{2}$ or $\frac{3\pi}{4}$.

Therefore for the values of slope $0 < \alpha < 1$, the chain code of the line will consist only of two code values 0 and 1. For example, from Figure 1.4 we have the chain code representation of the intersection pattern of the line $y = \frac{7}{16}x$, $0010101001010101 = (001)(01)^2(001)(01)^3$. For this range of slope, the chain code can be calculated by

$$c_i = \lfloor \alpha i + \beta \rfloor - \lfloor \alpha(i+1) + \beta \rfloor$$

An alternative to defining the intersection pattern as a set of pixels involves the use of run length encoding. A run is defined as a contiguous set of pixels with the same y coordinate. It is represented by the position of the left-most pixel, (x, y), and its length, r, which is the number of pixels in the run. As the intersection pattern

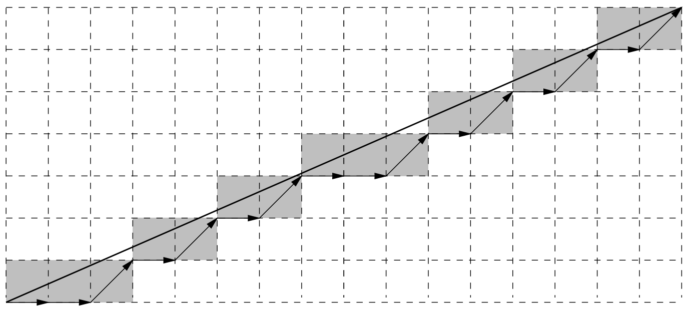


Figure 1.4: Chain code representation of the digital line.

contains only 8–connected sets of runs we can substitute the chain code segments which correspond to each run in the intersection pattern simply by the length of the run to produce an equally informative description. From our example of Figure 1.4 the chain code segment 001 corresponds to a run of length 3, 01 to a run of length 2. Therefore we can represent the intersection pattern by the sequence of run lengths $3, 2, 2, 3, 2, 2, 2$.

1.1.3 Symmetry and Special Cases

To simplify the problem of defining and generating the digitised line, the 8–way symmetry about the origin of a point defined in Cartesian coordinates is typically used [26]. As described in Figure 1.5, given that the intersection pattern can be represented by a set of lattice coordinates (x, y), all other lines can be generated from the line of slope α where $0 \leq \alpha \leq 1$. Therefore the use of symmetry allows us to limit the range of slopes we need to consider.

The concept of 8–way symmetry is mirrored in the rotational property of chain codes described by Freeman. Freeman also showed that for lines overlying the axes of the reflection, $\alpha = 0$ or $\alpha = 1$, the intersection patterns can be generated simply by repeating a single chain code value, forming two special cases

Therefore for the remainder of this work, we will restrict our attention to the line $\ell : y = \alpha x + \beta$ where $0 < \alpha < 1$ and $0 \leq \beta < 1$. All other lines can be achieved via symmetry or special case.

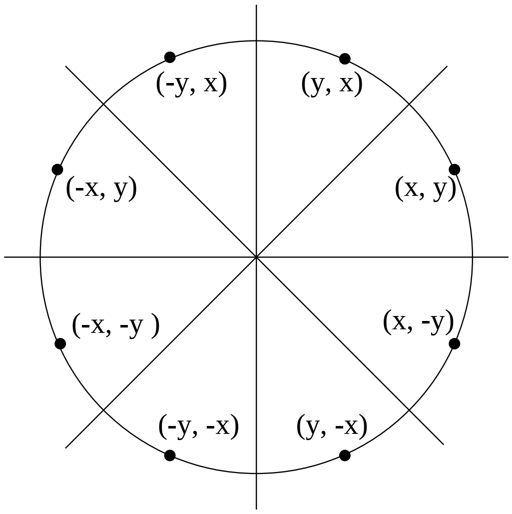


Figure 1.5: The 8–way symmetry of a point defined in Cartesian coordinates.

1.2 The Use of Lines in Computer Graphics

The two most common applications of the intersection of a line and a lattice in computer graphics are drawing a line defined between two pixels locations and the ray tracing of spatial subdivisions.

1.2.1 Line Drawing

Line drawing is the problem of calculating the pixels which best represent a line drawn between two pixels where the two defining pixels (x_0, y_0) and (x_1, y_1) represent the endpoints of the line. To draw a line without anti–aliasing, we seek the minimal set of 8–connected pixels which are closest to the line. Based on our mapping of pixel space to the lattice, the set of pixels which constitute the digital line can be calculated from the intersection of the line $\ell : y = \frac{y_1 - y_0}{x_1 - x_0} + \frac{1}{2}$. An example line, drawn from pixel $(0, 0)$ to $(11, 7)$ is given in Figure 1.6.

Algorithms answering the problem of line drawing are among some of the earliest and best known in computer graphics. Due to the fundamental nature of the problem, a wealth of literature has been presented on the subject and a number of studies have arisen which charactertise and summarise the properties of line drawing algorithms and the problem of line drawing [99, 54, 58, 122, 52, 10, 18, 26, 92, 131, 13, 19, 90].

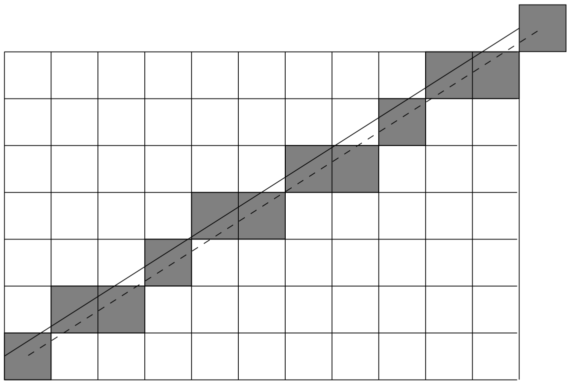


Figure 1.6: The intersection pattern of the line of best-fit from pixel (0, 0) to (11, 7).

The first algorithms presented for line drawing were tied to the incremental movements of digital plotting devices [124, 14, 33, 129, 15, 97, 12]. As raster–based graphical systems superceded vector–based systems, these techniques flowed over to this emerging technology due to the similar lattice–based mathematical structure of the display apparati.

From these early algorithms the classical work in the area was provided by Bresenham [14, 15]. This algorithm has a conditional structure where for each pixel in the line, a choice between only two is required. The decision is based on testing the sign of an integer decision parameter. The decision parameter is then updated by adding a single integer value based on the choice of the next pixel. Even though a number of different derivations of the algorithm exist [122] as well as a set of other similarly structured line algorithms [48, 108, 65, 57, 137, 86, 123, 77], Bresenham's algorithm still remains the standard line drawing algorithm, especially for hardware appliances where small, logically simple appliances are preferred. However, many different approaches have since been taken to the problem of line drawing which find their use in a variety of applications.

One set of techniques aim to accelerate or improve the performance of Bresenham's algorithm [64, 72, 8, 49, 91, 130, 104, 27, 28, 142, 7, 139, 143, 4, 109, 68, 66, 145]. The acceleration techniques which predominate include the use of symmetries within the line [64, 130, 145], the use of the gcd function to delimit subgroups of repeated pixels within the line [49, 104, 27, 28, 4], parallel implementations [139, 4] and altering the algorithm to calculate not just the next pixel but multiple steps of pixels per iteration [142, 7, 68, 66]. From these, the acceleration techniques most commonly applied to Bresenham's algorithm are the combination [143, 109] of the symmetry of the line about it's midpoint [64] and the double step technique [142].

A number of generalised approaches have also been presented for drawing nonparametric curves which do not improve the efficiency of line drawing but can also be applied to the problem [34, 76, 115, 9, 105, 127].

With the theoretical structural descriptions of the digital line being developed within the realms of digital image understanding, some of the results of this work was also applied to the problem of line drawing. These techniques include the use of linguistic methods such as context sensitive langauges [114] and context free grammars [21, 22], the fast Fourier transform [87], techniques related to cellular automation [126, 100] and a fractal technique based on the recursive bisection algorithm [106, 69, 70]. Some of these techniques as well as others have also been presented for parallel applications [5, 114, 131, 100, 139, 140, 94, 69, 70].

It is however one of the first structural techniques used for improving the efficiency of line drawing applications which forms the focus of this dissertation: the use of the hierachical structure of runs within the digitised line. While the initial techniques to use the hierarchy of runs [21, 22, 5, 30]. improved the scan conversion rates of digitising the line, they had a major drawback for line drawing as their representation of the intersection pattern was based on recursive manipulations of the chain code of the line which had to be calculated in its entirity before the line could be drawn. The use of run lengths however found its application to line drawing through techniques based only on runs of pixels [107, 49, 16, 17]. which improved upon the efficiency of pixel–based algorithms for lines of reasonable length. Of these run–based techniques another algorithm by Bresenham, the run length slice algorithm [20, 17, 61, 145], gained acceptance and in recent years a number of patents for devices that use the run length slice algorithm have been produced [42, 35, 36, 37, 38, 39, 40, 41]. The success of the run length slice algorithm is again based on the structure of the internal decision loop being of a similar structure to Bresenham's pixel–based algorithm. For each run in the line, a choice between two lengths is made based on testing the sign of an integer–based decision parameter which is then updated by a constant value based on the choice made.

Given the vast range of algorithms which exist for line drawing, the question begs to be asked, which algorithm is best? The answer of course is not as simple and depends on the application and the architecture of the target system. Hardware developers of graphics systems still prefer small, logically simple algorithms such as Bresenham's pixel–based algorithm. For appliances in which execution speed is more important than program size, which style of algorithm is most applicabale depends on the length of the line to be drawn: the shorter the line, the greater the proportional effect the initialisation costs have compared to the decision loop. Using acceleration or run–based techniques causes a comparatively large increase in the initialisation costs of the algorithm. Therefore it was suggested by Fung *et al.* [61] that: for very short lines Bresenham's original pixel–based algorithm should be used, for short lines the accelerated version of Bresenham's pixel–based algorithm, for intermediate length lines the run length slice algorithm, and for longer lines an accelerated version of the run length slice algorithm. However the position of the boundaries of the lengths which decide which algorithm is most applicable is dependent on the architecture of the system due to the use of different arithmetic in the initialisation stages of each style of algorithm.

There appears to be little statistical analysis of line length in various applications. While it is suggested that in interactive graphics applications short lines predominate, in applications such as laser printing, lines can be extremely long in comparison. If you consider that a line two inches long printed on a 600dpi laser writer contains 1200 pixels, the importance on techniques that improve the efficiency of line drawing for longer lines becomes apparent [61].

For use in interactive graphics systems, it should also be noted that the majority of computer monitors are still based on the cathode ray tube which was first developed in 1897 by the German scientist K. F. Braun. The future use of the cathode ray tube in computer monitors is surely limited due to its size, power requirements and limited range of resolution. Improvements in the manufacture of Liquid Crystal Displays (LCD) have seen the relative number of these units in use increase reducing space and power requirements. However this technology does not greatly improve upon the resolution rates available and typically reduces the angle at which the monitor can be viewed. What is more promising is the development of Light Emitting Polymers (LEP) which provides a large increase in the resolution rates possible.

The trend to higher resolution rates in a variety of applications indicates that the development of techniques to more efficiently draw longer lines will become increasingly important. This need is echoed by the suggestion of a technique for line drawing that foregoes the best approximation of the line for increased efficiency [106, 69, 70]. However improvements in line drawing have almost solely concentrated on improving the rate of scan conversion of the line. Depending on the architecture of the system being used the importance of scan conversion can be misconstrued. A bottleneck in line drawing implementations is individually setting each pixel in the display memory using the `setPixel(x,y)` command. A large increase in efficiency of the scan conversion stages of the algorithm may cause only a small increase in overall speed. While some simple rearrangements of the line algorithm and the pixel setting command can help improve this bottleneck [91], it would be more prudent to use the higher-order patterns or regularities within the line to be able to set more than one pixel at a time. While only one memory location can be addressed in any one memory cycle, memory organisations are available [136] where a memory primitive can be developed which sets a run of pixels or a run of runs, in a comparable time to setting a single pixel.

Therefore developments to improve the efficiency of line drawing algorithms especially for longer lines are as important now as they have been in the past. One important avenue will be the use of higher order structures within the line. What would also be useful is one algorithm that unifies a variety of algorithms into a common definition which can be used very simply to decide which level of description, the pixel, the run or a run of runs etc., is the most applicable to draw the line based on its length.

1.2.2 Ray Tracing

Ray tracing has become one of the two major global illumination models to generate high quality photo-realistic images used in computer graphics, the other being radiosity. In ray tracing, the objects in the scene are rendered by calculating the amount and colour of the light they reflect, transmit or emit to the eye or lens of the camera by tracing the path the light must take. After all it is the light disturbed or emanating from an object we see, not the object itself. Therefore for each pixel in the image, a set of rays are cast into the scene. If the rays strike or intersect an object, the amount of light that would be reflected, transmitted or emitted back along that ray is calculated and the pixel is set to the colour and intensity of the light accordingly.

To calculate the amount of light reflected or transmitted from the point intersected by a ray, the contribution of all light sources must be totaled. However the point may also be in shadow. Therefore a set of rays is sent out to each light source and if they strike any object before the light source, the point is in shadow from that source. Secondary reflected and transmitted light must also be accounted for. Therefore rays are cast along the path reflected and transmitted light from other objects would follow and if the ray strikes an object, the amount of light emanating from that point on the second object is calculated and added to the light reflected by the object itself. Therefore it is apparent that a major problem presented by the ray tracing technique is the combinatorial explosion of the number of rays needed to render a scene.

Early analysis of ray tracing by Whitted [135] who introduced the technique, indicated that over 95% of the time taken to render an image was spent in calculating the point of closest intersection. Therefore one of the main focii to accelerate ray tracing has been to more efficiently calculate the point of closest intersection. While the techniques employed have been very successful, due to the increased desire for higher quality images and more complex scenes, the original estimate of Whitted is still considered to be conservative [6].

In ray tracing, one of the most effective techniques to calculate the point of closest intersection is the use of uniform and nonuniform spatial subdivisions [6]. Each ray is traced through the subdivision and if a spatial division contains an object, the object is tested against the ray for an intersection. The first object-ray intersection found is therefore the closest. One of the common subdivision schemes is the uniform spatial subdivision based on the lattice. The most common subdivision scheme is the octree which if fully extended can also be abstracted by the lattice. Therefore as the ray can be represented by a line with real slope and intercept, the application of techniques for intersecting a line and a lattice is apparent.

The problem is therefore to find, in order, the 4–connected set of cells of the subdivision that are pierced by the ray. However we also need to handle corner intersections. Therefore if a corner of a lattice cell is intersected by the ray, each section of the intersection pattern divided by that cell corner will be 4–connected, and these two subsections will be 8–connected. An example line of intersection and the resultant intersection pattern is given in Figure 1.7.

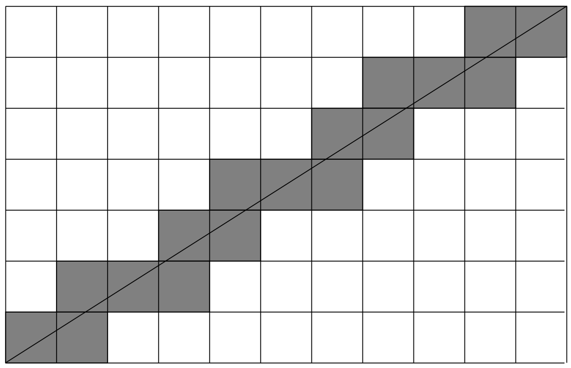


Figure 1.7: The intersection pattern of the line of intersection $y = \frac{7}{11}x$.

Based on the intersection pattern produced by a line and the lattice, various algorithms have been developed to intersect a ray and a uniform spatial subdivision, [59, 60, 1, 144, 120, 85], and nonuniform spatial subdivisions [59, 60, 128, 121, 63]. A survey of most of these methods and others is given by Endl and Sommer [50].

The algorithms used to trace a ray through a lattice–based subdivision still typically based on the the Digital Differential Analyser (DDA) [119] algorithm or similar algorithms which calculate the path of the ray a cell at a time. The application of the structural definition of the digital line to ray tracing is untreated. This is surprising as a number of results would be useful in accelerating the techniques employed. The use of higher order structures within the intersection pattern could be used to calculate more than one cell intersected by the ray at one time. Also the rays used to sample multiple reflections could possibly be calculated by perturbing the parameters used to define the digitisation of the ray. The structural definition of the intersection pattern of the line and the lattice has a lot to offer the application of ray tracing. What has been missing in the past is the link between the structural definition of the digitised line and the conditional techniques used to trace a ray through spatial subdivisions.

1.3 The Structure of the Digital Line.

In parallel with the development of the techniques in computer graphics, there has been a vast depth of work presented on the structure of the digitised line, particularly within the fields related to digital geometry [112]. These works have led to a detailed understanding of the structure and properties of the digitised line.

- The requirements of a set of pixels to represent a line have been described [111, 113, 141, 74, 73, 110, 84, 95, 83, 116, 117].
- Various properties, models and analyses of digital line segments have also been presented [107, 111, 21, 11, 62, 5, 30, 79, 141, 45, 2, 3, 102, 132, 88, 23, 24, 89, 32, 96].
- A variety of techniques for deriving a continuous representation of the line from its digital counterpart have been presented [138, 46, 45, 43, 118, 44, 101, 47, 146, 147].
- The accuracy of a digital line in representing a continuous line has also been evaluated [81, 133, 43, 18, 13, 25].
- The link between the structure of the digitised line and techniques in number theory such as Euclid's gcd algorithm, continued fractions and the Farey series has been illustrated [45, 22, 103, 93].

Much of this work has a direct application to the problem of representing and generating the structure of the hierarchy of runs within the digitised line.

1.3.1 When Does a Digital Arc Constitute a Line?

In common definition of the term line is less restricted than the mathematical definition which includes the condition of straightness. While in much of the literature the term *digital straight line or line segment* is commonly used, within this dissertation we will be using the mathematical definition.

A large proportion of the work presented in digital geometry looks at the requirements of a contiguous set of pixels to constitute a line. Much of this work can be used within computer graphics to quantify the correctness of line–based algorithms.

Freeman [55, 56] described three properties of a chain code which represents a digitised line. For the chain code to represent a digitised line:

- the chain code must consist of two codes n and m except when the line is vertical, horizontal or diagonal where the chain code consists of one repeated code,
- groups of these codes can be repeated within the line, also discussed in [49, 4], and
- the two codes should *alternate to a maximum extent*.

Rosenfeld [111] formalised Freeman's properties showing that they are consequences of the chord property of a digitised line. He defined and used the chord property to show the converse statements

- the digitisation of a line segment is a digital arc and has the chord property,
- if a digital arc has the chord property, it is the digitisation of a line segment.

Wu [141] showed the equivalence of Freeman's and Rosenfeld's properties and went on to show that these are sufficient conditions for the digitisation of a line. Hung and Kasvand [74] introduced another technique for determining if a pattern of pixels is the representation of a line. By taking segments of the line of equal length, if the sum of the values of the chain code differ by more than one, the pattern must represent an arc and not a line.

1.3.2 The Structure of Runs within the Digital Line.

The literature from digitial geometry also describes two important properties of the run structure of the digital line. They are that the composite runs can only occur in two lengths which are consecutive integers and that between any two runs of one of these lengths there must be at least one of the other. The existence of only two run lengths is suggested by the third property of the chain code of a line described by Freeman [55], that the horizontal and diagonal pixels movements which comprise the intersection pattern of the line should alternate to the maximum extent to assure minimum deviation from the true straight-line path.

The first to use run lengths to define the digitised line was Reggiori [107]. However he limited the slope of the line he was considering to $0 < \alpha < \frac{1}{2}$.

The formal presentation of the two fundamental properties of run lengths in the digital line was presented by Rosenfeld [111] who used the chord property to proof these results and others. He went on to state without proof that, *"for the run length that occurs in runs, these runs can themselves have only two lengths, which are consecutive integers; and so on"* recognising the hierarchical structure of runs in the digitisation of a line.

An algorithm to generate the chain code of a digital line of rational slope and zero intercept using the hierarchical structure of runs within the line was presented by Brons [21, 22]. It was the first of a set of algorithms that are based on the continued fractions algorithm and related to Euclid's gcd algorithm.

Arcelli and Massarotti [5] developed a modified version of the algorithm presented by Brons for the application of line drawing. They showed the line generated possessed the chord property described by Rosenfeld [111].

Cederberg [30] also presented an algorithm for a line with rational slope. He applied this result to line drawing by using a technique to shift the line structure such that the endpoints of the line coincide with two pixel locations. He showed that the line generated by the algorithm was the same as that generated by Bresenham's algorithm. Wu [141] extended the treatment to lines with real slope and intercept as did Dorst and Duin [45].

The use of Euclid's gcd algorithm and continued fractions was also investigated in the series of papers produced by Pitteway, Castle and Green [103, 27, 28] in which they endeavored to use aspects common to Euclid's algorithm and the continued fractions algorithm to speed up line drawing algorithms based on Bresenham's algorithm.

A problem with these techniques is that the link between continued fractions and the structure of the run–based hierarchy of the digitised line has not been completely understood. As we will discuss in Chapter 4, the link lies not with Euclid's gcd algorithm and regular continued fractions which are calcluated by the continued fractions algorithm but with ordinary continued fractions and the least remainder gcd algorithm. Ordinary continued fractions are a more general form of continued fractions than regular continued fractions [71, 78]. However the divergence from the regular form only appears in a minority of cases.

Due to the discrete nature of the lattice, number theory plays an important role in characterising the digital line. As well as to continued fractions [93], the link between the structure of the digitised line, Farey series [45, 22, 93] and Diophantine equations [93, 4] have been illustrated.

1.4 Introducing the Run–Based Structure of the Digitised Line

To begin our exploration of the structure of runs in the digitised line, we will start by informally introducing the ideas we will present in the next chapter. During this introduction we will be considering within the frame $\mathcal{L}_{(d,a)}$ the intersection pattern of the line $\ell : y = \frac{a}{d}x$ where $a, d \in \mathcal{N}$, $0 < \frac{a}{d} < 1$ and $\gcd(a, d) = 1$, To calculate the intersection pattern of a line and a lattice early techniques dealt with the pattern at the pixel level, determining the next pixel in the pattern based on the location of the current pixel. Consider that the intersection pattern of the line ℓ within the frame $\mathcal{L}_{(d,a)}$ contains d pixels which corresponds to the X-axis size of the frame, the process of calculating the next pixel therefore has to be repeated d times. Thus the emphasis for providing an efficient algorithm for determining the intersection pattern was on reducing the cost of determining the next pixel in each iteration.

The success of techniques such as Bresenham's pixel–based algorithm [15] and the DDA algorithm [119] in their relevant applications is based on their use of the 8–way symmetry of Cartesian coordinates about the origin. Using this symmetry, the range of slopes of the lines to be considered can be restricted to lie between 0 and 1. For this range of slopes, the next pixel in the intersection pattern can only come from a choice of two. If the current pixel position is (x, y), the next pixel position must be $(x + 1, y)$ or $(x + 1, y + 1)$.

As the ideas behind the techniques for calculating the intersection pattern developed, higher order constructs or sub-patterns were described in the intersection pattern. The aim of these techniques was not to reduce the cost of calculating the next pixel but to combine the calculation of several pixels into one iteration thus reducing the number of iterations needed. One of the first higher order constructs used was the *run*defined as a contiguous set of pixels with the same y-coordinate. A run is typically described by the position of its left-most pixel and its length measured in pixels.

Reggiori [107] first noted that the digitised line can only contain runs of two lengths and these run lengths must be consecutive integers. Rosenfeld [111] later provided a formal proof of these facts. Reggiori implemented a line drawing algorithm based on his observations which improved upon Bresenham's pixel–based line drawing algorithm for lines of slope between 0 and $\frac{1}{2}$ where the run lengths possible are guaranteed to be greater than one. The conditional structure of Reggiori's algorithm was similar to Bresenham's except that instead of deciding which pixel from a choice of two should be the next pixel, the choice was of which run length should be used.

Lines of slope greater than $\frac{1}{2}$ were not considered as the increased initialisation costs outweighed the improvement in improving the efficiency of the decision loop. Run–based algorithms require the integer component of a floating point division to calculate the possible run lengths which was relatively more expensive at the time that Reggiori presented his algorithm.

To understand the structure of runs in the intersection pattern, consider that within the intersection pattern of the line $\ell : y = \frac{a}{d}x$ there are a runs which correspond to the Y-axis size of the lattice. As the possible run lengths are only two and consecutive integers we will refer to them as short and long. To determine what run lengths are possible within the intersection pattern, there are d pixels to be distributed among a runs, the distribution being as even as possible. If we divide up the d pixels into a runs of length $r = \left\lfloor \frac{d}{a} \right\rfloor$ we have $n \equiv d \bmod a$ pixels remaining, $0 \leq n < a$, which have to be distributed along the pattern. Therefore in the intersection pattern of ℓ there are n long runs each with $r+1$ pixels and $a-n$ short runs with r pixels each.

For example, consider the line depicted in Figure 1.8. The line contains 41 pixels and 17 runs. The possible run lengths are $r = \left\lfloor \frac{41}{17} \right\rfloor = 2$ and $r+1 = 3$. There are $41 \bmod 17 \equiv 7$ long runs of length three shown in light grey and $17 - 7 = 10$ short runs of length two shown in dark grey.

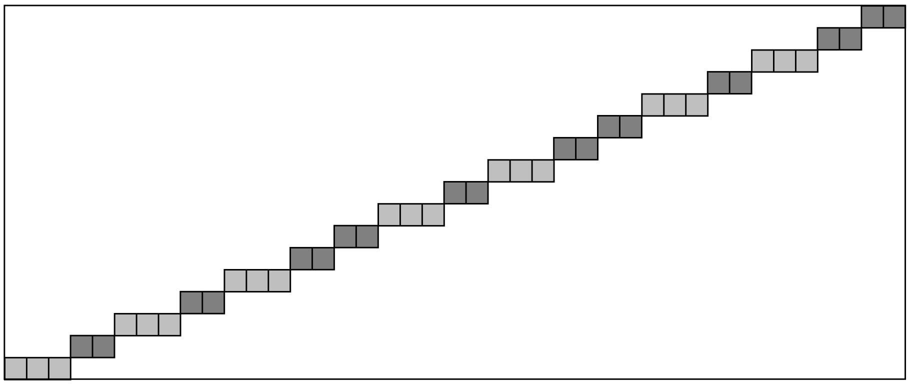


Figure 1.8: The intersection pattern of the line $y = \frac{17}{41}x$.

Therefore using a run–based algorithm improves upon pixel–based algorithms as only a and not d iterations of the decision loop are required. Although this improvement must be weighed against the increased initialisation costs especially for slopes of lines approaching one and very short lines where the decision loop is iterated only a few times.

The use of runs provides a higher level of description than pixels of the intersection pattern of the line and can be used to improve the efficency of line-lattice intersection algorithms. However higher orders of organisation exist within the intersection pattern. Rosenfeld [111], as well as proving that runs can only occur in two lengths which are consecutive integers also proved that between any two runs of one length there must be at least one run of the other. That is, single occurences of run length interdisperse sequences, one or more, of the second.

To describe the higher orders of organisation of runs within the intersection pattern, we will use the term l to represent a long run and s a short run. We will combine these symbols to discuss sequences of runs. For example, the term ls^+ represents a long run followed by one or more short runs, l^+s represents one or more long runs followed by a short run. Let us consider grouping together the sequences of runs into *runs of runs* such that we have four possible shapes of runs of runs ls^+, l^+s, sl^+ and s^+l which correspond to the four possible permutations of grouping a singularly occuring run with a sequence of runs.

From geometry it can be shown that the first run in the intersection pattern of the line $\ell : y = \frac{a}{d}x$ must be long. Therefore we need only two shapes, ls^+, a long run followed by one or more short runs if there are more short runs than long runs or l^+s if there are more long runs than short. In the case where there are equal numbers of runs the two cases are equivalent.

For example in Figure 1.8 the complete sequence of runs is $lslslsslslsslslss$ therefore we have singularly occuring long runs separating sequences of shorts runs: the intersection pattern is constructed of runs of runs with the shape ls^+.

The terminology of runs of runs is cumbersome so let us define these runs of runs to be *second order runs*. Where a run is defined by its position and length, a second order run is defined by its position, its length defined by the number of runs which comprise it and its shape being determined by whether there are more long or short runs in the pattern. By analogy with second order runs, we can define runs to be first order runs and pixels to be zero order runs. A hierarchy starts to emerge.

To determine the possible lengths of the second order runs let us consider the case where there are more short runs than long and continue the use of our example. In this case, the number of second order runs must be the same as the number of long runs in the pattern by definition. If there are n long runs in the line there are a runs to divide amongst n second order runs. Therefore if the division is to be as even as possible, the length of a short second order run, $r^{[2]}$, will be $r^{[2]} = \left\lfloor \frac{a}{n} \right\rfloor$. There will be $n^{[2]} \equiv a \bmod n$ long runs and $a - n^{[2]}$ short runs of order 2.

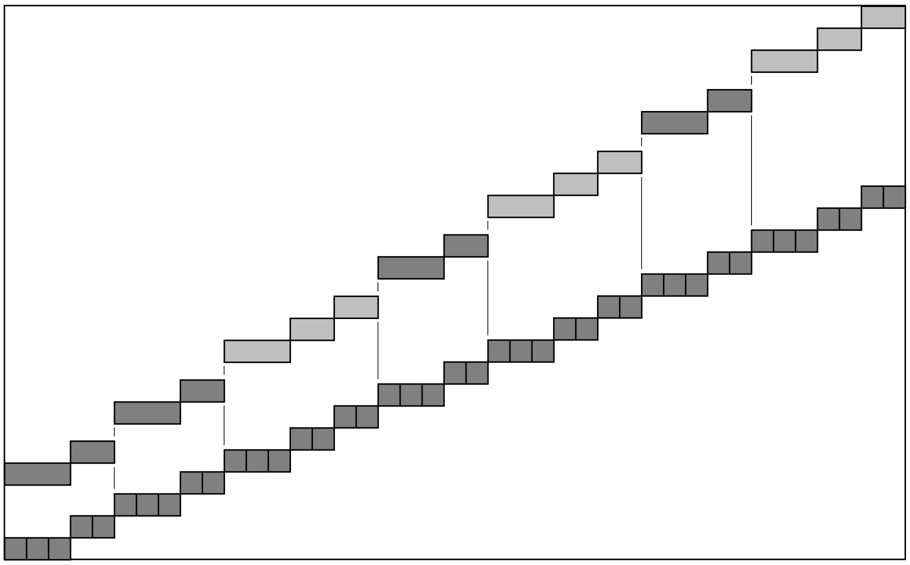


Figure 1.9: The construction of order 2 runs in the line $y = \frac{17}{41}x$.

Figure 1.9 shows the intersection pattern of the example line and the second order runs which comprise it. There are more short runs than long so the second order runs have the shape ls^+, one long run followed by a sequence of short runs. There are three long runs of length three shown in light grey and four short runs of length two shown in dark grey.

There is no reason to end this hierarchical description at order 2. We can define a recursive hierarchical description of the intersection pattern of the line ℓ based on defining runs of higher order. For orders three and above, there is no restriction that the first run in the intersection pattern must be long and therefore all four possible shapes can occur. For order i, if there are more short runs of order $i-1$ in the intersection pattern, the shape of the order i runs will be s^+l or ls^+. If there are more long runs, the shape will be l^+s or sl^+.

An example of third order runs in the intersection pattern of ℓ is shown in Figure 1.10. There are three runs of order 3 amongst which the seven runs of order 2 are to be distributed as evenly as possible. Therefore the length of a short run of order 3 is $r^{[3]} = \left\lfloor \frac{7}{3} \right\rfloor = 2$, and amongst the three order 3 runs there will be $7 \bmod 3 \equiv 1$ long run of length $r^{[3]} + 1 = 3$. The shape of the order 3 runs is s^+l.

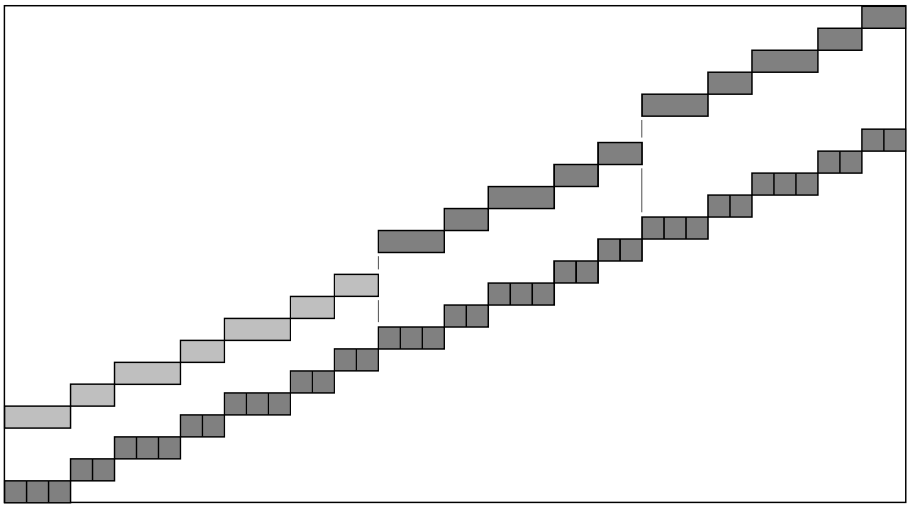


Figure 1.10: The construction of order 3 runs in the line $y = \frac{17}{41}x$.

For lines with a rational gradient, the hierarchical description of ordered runs within the intersection pattern will be be bounded, as the intersection pattern is eventually repeated if a and d are not coprime.

For the example of $\ell : y = \frac{17}{41}x$ the process reaches its conclusion at order 4. There are three order 3 runs of which one is long and two are short. There will be therefore one order 4 run containing all three order 3 runs and starting with the only long order 3 run. The order 4 run is the intersection pattern of the line $\ell : y = \frac{17}{41}x$ and the lattice $\mathcal{L}_{(41,17)}$.

1.4.1 Generating the Intersection Pattern

Having defined the structure of the hierarchy of runs within the digitised line as we have, the important question is what does this definition provide us which previous definitions do not? The benefit of this definition is that a simple scheme can be devised in which the lengths of each run of any order in the line can be decided using an efficient conditional style algorithm.

The algorithms used for line drawing or ray tracing in computer graphics all possess a similar conditional structure regardless of whether they are based on pixels such as Bresenham's pixel–based algorithm or the DDA algorithm, runs such as Reggiori's algorithm, or a mixture of runs and runs of runs such as the run length slice algorithm. The slopes that are considered are bounded to lie in the range $0 < \alpha < 1$. For pixel–based algorithms that limits the choice of the next pixel to a possible set of two. For run–based algorithms, the choice is made between the two possible run lengths that can exist in the line. In all of these algorithms the choice is made by checking the value of a decision parameter against the value of zero. For values less than zero one element of the possible set of choices is used, for values greater than zero, the other choice. For a value of zero, each technique handles this case differently. For line drawing applications, either choice is equally applicable and for ray tracing this typically signifies a corner intersection.

The algorithm we describe in this dissertation to construct the runs in the digitial line at any level of runs within the hierarchy possesses an identical conditional structure. In fact we will show that because the run–based structure we have described defines the structure of runs within the digitised line and the definition of a run can be naturally extended to include pixels and the recursive definition of runs of runs, it is possible to provide a universal description of conditional line drawing algorithms such as Bresenham's pixels–based and run length slice algorithms.

To introduce the ideas we will present for our conditional algorithm, we will be considering the simplest level of runs in the hierarchy, order 1. The first question we should answer is what is the decision parameter and how should it be used and updated. In Figure 1.11, the geometry of a run of order 1 in respect to the continuous representation of the line is described. From Figure 1.11 we have the current position of the run to be set and we require only for its length to be decided. The decision parameter we will be using is the numerator of the fractional component of the intercept at the beginning of each run, $b_j^{[1]}$.

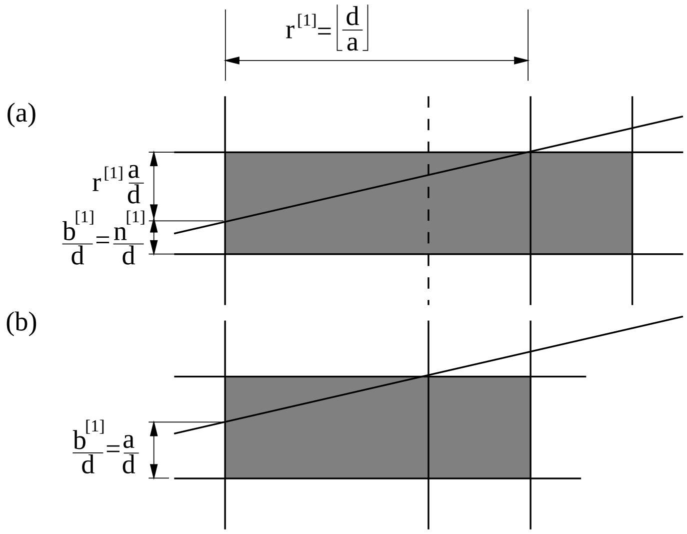


Figure 1.11: The intercept geometry of the j-th run of order 1 in the line.

The sequence of values of the decision parameter we will refer to simply as the numerator sequence of order 1. The numerator sequence of order 1 is defined for $j = 0, 1, 2, \ldots$ by

$$b_j^{[1]} = ax_j^{[1]} - dy_j^{[1]} \tag{1.3}$$

where $(x_j^{[1]}, y_j^{[1]})$ is the position of the start of the run. Where $r_j^{[1]}$ is the length of the j–th run, the difference between two successive values in the numerator can be derived from Equation 1.3 by considering that position of the start of the $(j+1)$–th run, $(x_{j+1}^{[1]}, y_{j+1}^{[1]}) = (x_j^{[1]} + r_j^{[1]}, y_j^{[1]} + 1)$. Therefore

$$\begin{aligned} b_{j+1}^{[1]} - b_j^{[1]} &= a(x_j^{[1]} + r_j^{[1]}) - d(y_j^{[1]} + 1) \\ &= ar_j^{[1]} - d \end{aligned} \tag{1.4}$$

In Figure 1.11(a) we have set the value of the numerator of the intercept such that the line intersects the point $(x_j^{[1]} + r^{[1]}, y_j^{[1]} + 1)$. This is the critical value of the decision parameter. If the intercept has this value, the run is short and the value of the next numerator value $b_{j+1}^{[1]} = 0$. If the intercept has a greater value, the line is shifted upwards and the run must be short. For a lesser value, the run must be long. Hence from Equation 1.4 as $b^{[1]}_{j+1} = 0$ we have $b^{[1]}_j = n^{[1]}$. Therefore if the value of $b^{[1]}_j < n^{[1]}$, the next run in the line must be long. If $b^{[1]}_j \geq n^{[1]}$, the next run must be short.

To describe how to calculate the next value of the numerator sequence at order 1 based on the decision of the length of the next run in the line, consider the case where the next run is short. From Equation 1.4 we have $r^{[1]}_j = r^{[1]}$ and

$$\begin{aligned} b^{[1]}_j - b^{[1]}_j &= ar^{[1]} - d \\ &= -n^{[1]} \end{aligned} \tag{1.5}$$

If the next run is long, $r^{[1]}_j = r^{[1]} + 1$ and

$$\begin{aligned} b^{[1]}_j - b^{[1]}_j &= a(r^{[1]} + 1) - d \\ &= a - n^{[1]} \\ &= m^{[1]} \end{aligned} \tag{1.6}$$

Therefore, we have a decision parameter based on the numerator sequence and depending on the choice of the next run length the parameter is adjusted by a single integer addition for each iteration. The values of the numerator sequence can also be translated by $-n^{[1]}$ to test against the value of zero which can be implemented as a simple sign check.

From our example line $\ell : y = \frac{17}{41}x$ the numerator sequence of order 1 is $b_j[1] = \{0, 10, 3, 13, 6, 16, 9, 2, 12, 5, 15, 8, 1, 11, 4, 14, 7\}$ and the run sequence of order 1 is $r^{[1]}_j = lslslsslslsslslss$ for $j = 0, \ldots, 16$. The long runs coincide with values of the numerator sequence which are less than $n^{[1]} = 7$, $b^{[1]}_j = \{0, 3, 6, 2, 5, 1, 4\}$. The short runs coincide with values of the numerator sequence which are greater than $n^{[1]} = 7$, $b^{[1]}_j = \{10, 13, 16, 9, 12, 15, 8, 11, 14, 7\}$.

In the next chapter we will also describe many of the properties of the numerator sequence. One of the most important is that for our example at order 1, all the values of the sequence are less than a. This is explained geometrically in Figure 1.11(b). If the line was raised any further, the run would have an invalid length. It is also interesting the note that every value in the range 0 to a is represented in the numerator sequence.

The case at order 1 is however the simplest. For higher orders the derivation is complicated by the added characteristic of the shape of the run. However the extension we provide does not deviate from the basic conditional algorithm that exists at order 1.

Chapter 2

The Intersection Pattern of a Line with Rational Slope

In this chapter we will restrict our attention to the intersection pattern, $\mathcal{P}_\ell$, defined by the line $\ell : y = \frac{a}{d}x$ where $a, d \in \mathcal{N}$ and are coprime, and $0 < a < d$ hence $0 < \frac{a}{d} < 1$, within the bounded frame of the lattice $\mathcal{L}_{(d,a)}$. All other lines can be considered using symmetry or special case. A line with $g = \gcd(a, d) > 1$ will have g repeated patterns of $\mathcal{P}_\ell$ within its intersection pattern, [49]. Therefore we need only consider $\mathcal{P}_\ell$ in $\mathcal{L}_{(d,a)}$.

2.1 Introduction

To describe the structure of the intersection pattern, $\mathcal{P}_\ell$, of the line $\ell : y = \frac{a}{d}x$, we will present a definition of the structure of the hierarchy of runs which constitutes the intersection pattern. We extend the definition of a run to include a recursive definition of a run of order i starting at the pixel or cell level. A pixel or a cell of the lattice is defined as a run of order 0. A run in the traditional sense is defined as a run of order 1. Hence, a run of order 1 is a collection of runs of order 0. A run of order i is therefore defined as a collection of runs of order $i - 1$. Hence the order of a run defines the level within the hierarchy of runs at which the run is defined and will be denoted for each of the parameters to be described, by the superscript of $[i]$.

To complete the definition of an ordered run, the properties of the run at each order will be defined. A run of order 0 has only one property, that of position. This property is inherent at each order and we will use the term $(x^{[i]}, y^{[i]})$ to describe the position of a run of order i.

A run of order 1 is defined as a contiguous set of cells or pixels with the same y coordinate. Therefore it is characterised by its position, defined as the leftmost pixel, $(x^{[1]}, y^{[1]})$ and by the additional property of length measured in runs of order 0 or pixels. From results presented by Reggiori [107] and Rosenfeld [111] we know two important facts about the length of runs of order 1 that occur in the intersection pattern. Firstly, there are only two possible lengths of runs of order 1 in the intersection pattern, $\mathcal{P}_\ell$, and these lengths are consecutive integers. Secondly, that if the intersection pattern, $\mathcal{P}_\ell$, is to represent a line, between any two of one of the possible run lengths, there must be at least one run of the other length.

Higher order runs, runs of order $i > 1$, also possess the attributes of position and length where the length is measured in runs of order $i - 1$. Rosenfeld extended his results on the possible run lengths of order 1 in the intersection pattern and stated without proof that higher order runs also have only possible lengths which are consecutive integers, $r^{[i]}$ and $r^{[i]} + 1$, and that singular occurrences of one of these lengths must punctuate sequences of the other. While we will formalise these statements later in the chapter we will initially assume they are correct to describe how runs of higher orders will be defined.

It will be useful to define some terminology to aid in the discussion of run length.

Definition 2.1.1 *For order $i \geq 1$ a run of length $r^{[i]}$ is defined to be short and a run of length $r^{[i]}+1$ is defined to be long. A short run shall be denoted by the code s, and a long run by l. A sequence of one or more short or long runs shall be denoted as s^+ and l^+ respectively.*

As well as position and length, higher-order runs have the additional property of shape of which four shapes are possible shape 0: sl^+, shape 1: ls^+, shape 2: s^+l and shape 3: l^+s. Therefore higher order runs are defined by the boundary of the singularly occurring run length. This definition has two immediate results. The length of each run must be at least two and the number of runs of order $i > 1$ is the number of least occurring run length of order $i-1$.

Having defined the recursive structure of a run of order $i \geq 0$, we will also need to consider the structure of these runs within the intersection pattern, $\mathcal{P}_\ell$. To do this we will be defining a number of run parameters. The complete list of parameters that define the structure of runs within the intersection pattern are

$r^{[i]}$ the length of a run of order i,

$s^{[i]}$ the shape of a run of order i which determines how the long and short runs of order $i-1$ are placed in the run of order i,

$p^{[i]}$ the total number of runs of order i in the intersection pattern,

$t^{[i]}$ the type of a run of order i which is used to calculate the shape and determines which run parameter $n^{[i]}$ or $m^{[i]}$ describes the number of long or short runs of order i in the intersection pattern,

$m^{[i]}$ if the type is $t^{[i]} = 0$, $m^{[i]}$ is the number of long runs of order i in the intersection pattern and if $t^{[i]} = 1$, $m^{[i]}$ is the number of short runs,

$n^{[i]}$ if the type is $t^{[i]} = 0$, $n^{[i]}$ is the number of short runs of order i in the intersection pattern and if $t^{[i]} = 1$, $n^{[i]}$ is the number of long runs.

In the remainder of this chapter we will define the relationships and calculation of the run parameters. We will also show that run length and type are necessary and sufficient to show the similarity of two intersection patterns. Having defined the structure of the intersection pattern we will then describe its generation from the run parameters. The technique we present is a conditional style algorithm which can be used at any level of the hierarchy to determine the length of each run in the intersection pattern.

2.2 The Structure of the Intersection Pattern

2.2.1 Extending the Definition of a Run

We start with a formal definition of an ordered run to include pixels, runs and the recursive definition of runs of runs.

Definition 2.2.1 *A run in the 8–connected intersection pattern $\mathcal{P}_\ell$ is defined by the following.*

- *A pixel or lattice cell is a run of order* 0 *and is defined by its position* $(x^{[0]}, y^{[0]})$.
- *A run of order* 1 *is a contiguous set of runs of order* 0 *with the same* y *coordinate. A run of order* 1 *is defined by its position* $(x^{[1]}, y^{[1]})$ *and its length* $r_j^{[1]}$ *measured in the number of order* 0 *runs that compose it.*
- *A run of order* $i \geq 2$ *is a set of runs of order* $i-1$ *such that the shape of the run has one of four possible shapes shape 0:* sl^+, *shape 1:* ls^+, *shape 2:* s^+l *or shape 3:* l^+s. *A run of order 2 has the special condition that it can have only one of two shapes,* ls^+, l^+s. *As well as shape, a run of order* i *is defined by its position* $(x^{[i]}, y^{[i]})$ *and its length* $r_j^{[i]}$ *measured in the number of order* $i-1$ *runs that compose it.*

The special case regarding run shape applied to runs of order 2 occurs because the first order 1 run in the line with zero intercept is always long as can be shown by geometry. Consider Figure 2.1. Due to the restriction that $0 < a < d$, we have

$$r_0^{[1]} = \lceil r \rceil = \left\lceil \frac{d}{a} \right\rceil = \left\lfloor \frac{d}{a} \right\rfloor + 1 = r^{[1]} + 1 \tag{2.1}$$

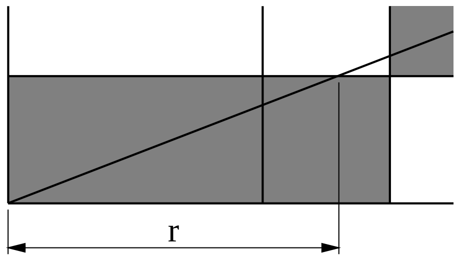


Figure 2.1: The length of the first order 1 run in the intersection pattern.

To show that the extended definition of a run is complete, we need to formally extend the results of Rosenfeld [111] regarding possible run length lengths that occur in the intersection pattern. Rosenfeld defined and used the chord property to prove the following theorem for runs of order 1.

Theorem 2.2.1 *For the intersection pattern, $\mathcal{P}_\ell$, of the line $\ell : y = \frac{a}{d}x$ and the lattice $\mathcal{L}_{(d,a)}$ where $0 \leq a \leq d$ and $a, d \in \mathcal{N}$, there are only two possible lengths of order 1 runs and these lengths are consecutive integers. For one of the two run lengths, between any two runs of that length there must be at least one run of the other length.*

Rosenfeld also stated that these properties are applicable to "runs of runs" without providing a proof. However due to the importance of the definition of higher order runs and the application of these results on run length to higher orders, we now provide a formal proof of Rosenfeld's conjecture regarding the length of higher-order runs. To use a common nomenclature, we redefine the chord property.

Definition 2.2.2 *A set of pixels contained in $\mathcal{L}_{(d,a)}$ between the points $P_0 : (0,0)$ and $P_1 : (d,a)$ is said to have the chord property if the line $\overline{P_0P_1}$ lies near $\mathcal{P}_\ell$. We say that $\overline{P_0P_1}$ lies near $\mathcal{P}_\ell$ if, for any (real) point (x, y) of $\overline{P_0P_1}$, there exists a lattice point (i, j) of $\mathcal{P}_\ell$ such that* $\max\{|i - x|, |j - y|\} < 1$.

We extend Rosenfeld's first assertion regarding the possible run lengths to include higher-order runs.

Theorem 2.2.2 *A run of order $i \geq 1$ can have only two lengths, measured in runs of order $i - 1$, which are consecutive integers, $r^{[i]}$ and $r^{[i]} + 1$.*

Proof : For order i let the j-th run in the intersection pattern of the line have length $r_j^{[i]}$ measured in runs of order $i-1$. Suppose the k-th run has a length $r_k^{[i]} \geq r_j^{[i]} + 2$. Assume that no closer pair of runs differ in length of more than 2 runs of order $i-1$. Then between them say we have $h \geq 0$ runs of length $r_j^{[i]} + 1$.

Over the length of this segment the intersection pattern rises by $(h+2)$ in a horizontal distance of $(r_j^{[i]} + 2) + h(r_j^{[i]} + 1) + r_j^{[i]} = (h+2)(r_j^{[i]} + 1)$ measured in runs of order $i-1$. The slope of this line segment is therefore $\alpha^{[i-1]} = \frac{1}{r_j^{[i]}+1}$ again measured in runs of order $i-1$.

Let $k < j$. At the point on the line segment where the y coordinate corresponds to the end of the first run of order $i-1$, $y_k^{[i-1]} = 1$, and

$$x_k^{[i-1]} = \frac{y_k^{[i-1]}}{\alpha^{[i-1]}} = r_k^{[i]} - 1 \tag{2.2}$$

Therefore the line passes above the last run of order $i-1$ in the k-th run in the intersection pattern. The chord property is violated.

Let $j < k$. At the point on the line segment $y_j^{[i-1]} = 1$, $x_j^{[i-1]} = r_j^{[i]} + 1$ and the line passes to the right of the first run of order $i-1$ in the $k+1$-th run in the intersection pattern. The chord property is violated. □

Rosenfeld's second assertion stated that for one of the two run lengths, between any two runs of that length there must be at least one of the other. Therefore if there are more short runs than long, the short runs occur in sequences punctuated by single long runs and vice versa. We extend the theorem to include higher-order runs completing the extension to Rosenfeld's theorem on run-length within the intersection pattern.

Theorem 2.2.3 *For order i, for one of the two possible run lengths, $r^{[i]}$ or $r^{[i]} + 1$, between any two runs of that length, there must be at least one run of the other length.*

Proof : For order i, suppose the j-th and $j+1$-th runs have length $r^{[i]} + 1$ and the k-th and $k+1$-th runs have length $r^{[i]}$. The height measured in runs of order

i of this collection of runs is 4. The width measured in runs of order $i-1$ is $2(r^{[i]}+1)+2r^{[i]} = 2(2r^{[i]}+1)$. Therefore the slope of the line segment is $\frac{2}{2r^{[i]}+1}$. The run length of order i, $r^{[i]} > 1$.

Let $k < j$. At the point on the line segment $y^{[i]} = 2$, the value of $x^{[i]} = 2r^{[i]} + 1$. Therefore the line segment passes to the right of the final order $i-1$ run in the $k+1$-th run and the chord property is violated.

Let $j < k$. At the point on the line segment $y^{[i]} = 2$, the value of $x^{[i]} = 2(r^{[i]}+1)-1$. In this case the line segment passes to the left of the final order $i-1$ run in the $j+1$-th run and the chord property is violated. □

While we have shown that only two run lengths of order i are possible in the intersection pattern, $\mathcal{P}_\ell$, we have not yet provided a way of calculating the possible run lengths. Where $p^{[i]}$ is the total number of runs of order i in $\mathcal{P}_\ell$ we have the following theorem.

Theorem 2.2.4 *The length of a short run of order $i > 0$ is,*

$$r^{[i]} = \left\lfloor \frac{p^{[i-1]}}{p^{[i]}} \right\rfloor \tag{2.3}$$

The length of a long run of order $i > 0$ is therefore $r^{[i]} + 1$.

Proof : By Theorem 2.2.2 the length of a long run must be $r^{[i]} + 1$. Let $u^{[i]}$ be the number of long runs of order i in the line ℓ. Therefore $p^{[i-1]} = r^{[i]}p^{[i]} + u^{[i]}$. We know that $0 \le u^{[i]} < p^{[i]}$ and that $r^{[i]}$ is an integer > 0. Hence

$$r^{[i]} = \frac{p^{[i-1]}}{p^{[i]}} - \frac{u}{p^{[i]}} = \left\lfloor \frac{p^{[i-1]}}{p^{[i]}} \right\rfloor$$

□

One last result that we need to show is that the shape of any particular order is invariant throughout the intersection pattern.

Lemma 2.2.5 *The shape of each run of order i is consistent for the intersection pattern $\mathcal{P}_\ell$.*

Proof : This lemma is apparent from the definition of the shape of the run, definition 2.2.1, and Theorem 2.2.3 regarding the sequence of run lengths. $\square$

2.2.2 The Structural Run Parameters

Having extended the definition of a run we now need to provide structural information regarding the composition of runs within the intersection pattern, $\mathcal{P}_\ell$, for any order. The information we need to provide for any order is the total number of runs for each order in $\mathcal{P}_\ell$, the number of long and short runs and the shape of the runs. For higher-order runs, we define an extra parameter, type, which will be used to define the shape of the run.

Definition 2.2.3 *For order $i \geq 2$, we define the type to be 0 ($t^{[i]} = 0$) when $n^{[i-1]} < m^{[i-1]}$ and the type to be 1 ($t^{[i]} = 1$) when $m^{[i-1]} < n^{[i-1]}$.*

There are two things to note about the definition of type. Firstly, the type parameter is only defined for order $i \geq 2$. The notion of type for orders zero and one is vacuous. Secondly, the case of $m^{[i-1]} = n^{[i-1]}$ is not included in the definition. It can only occur just before termination of the recursive definition of the run parameters and only when $m^{[i-1]} = n^{[i-1]} = 1$ and no type $t^{[i]}$ is defined. We will discuss this case later in this section.

Definition 2.2.4 *For the intersection pattern $\mathcal{P}_\ell$ of the line $\ell : y = \frac{a}{d}x$ and the lattice $\mathcal{L}_{(d,a)}$,*

1. *let $p^{[i]}$, be the total number of runs of order i*
2. *for order 1, let*
   - *$m^{[1]}$ be the number of short runs*
   - *$n^{[1]}$ be the number of long runs*
3. *for order $i > 1$ and*

- *type* $t^{[i]} = 0$, *let*

  $m^{[i]}$ *be the number of long runs*

  $n^{[i]}$ *be the number of short runs*

- *type* $t^{[i]} = 1$, *let*

  $m^{[i]}$ *be the number of short runs*

  $n^{[i]}$ *be the number of long runs*

By this definition $p^{[i]}$, $m^{[i]}$ and $n^{[i]}$ are positive integers.

Theorem 2.2.6 *The relationship between the structural parameters* $p^{[i]}$, $m^{[i]}$ *and* $n^{[i]}$ *is*

$$p^{[i]} = m^{[i]} + n^{[i]} \tag{2.4}$$

The total number of runs of order i, $p^{[i]}$, *we can determined by the relationship*

$$\begin{aligned} p^{[0]} &= d && (2.5)\\ p^{[1]} &= a && (2.6)\\ p^{[i]} &= \min\{m^{[i-1]}, n^{[i-1]}\} \text{ for } i \geq 2 && (2.7) \end{aligned}$$

For order 1 we have

$$\begin{aligned} n^{[1]} &\equiv p^{[0]} \pmod{p^{[1]}} && (2.8)\\ m^{[1]} &= p^{[1]} - n^{[1]} && (2.9) \end{aligned}$$

For order $i \geq 2$, *and type* $t^{[i]} = 0$ ($p^{[i]} = n^{[i-1]} < m^{[i-1]}$)

$$\begin{aligned} m^{[i]} &\equiv p^{[i-1]} \pmod{p^{[i]}} && (2.10)\\ n^{[i]} &= p^{[i]} - m^{[i]} && (2.11) \end{aligned}$$

For order $i \geq 2$, *and type* $t^{[i]} = 1$ ($p^{[i]} = m^{[i-1]} < n^{[i-1]}$)

$$\begin{aligned} n^{[i]} &\equiv p^{[i-1]} \pmod{p^{[i]}} && (2.12)\\ m^{[i]} &= p^{[i]} - n^{[i]} && (2.13) \end{aligned}$$

Proof : Equation 2.4 is apparent from Theorem 2.2.2 and Definition 2.2.1.

In the line $y = \frac{a}{d}x$ we have $p^{[0]} = d$ runs of order 0 (cells or pixels) and $p^{[1]} = a$ runs of order 1.

Therefore the $p^{[0]} = d$ runs of order 0 are divided into $p^{[1]} = a$ runs of order 1 whose lengths can differ only by one. Therefore the number of long runs must be $n^{[1]} \equiv d \bmod a \equiv p^{[0]} \bmod p^{[1]}$. The number of short runs is therefore $m^{[i]} = p^{[1]} - n^{[1]}$. The theorem is shown for order 1.

Assume that $p^{[i-1]}$ is the number of runs of order $i-1$ in the line ℓ. By Definition 2.2.1 and Theorem 2.2.3, the number of runs of order i is equal to the minimum number of long or short runs of order $i-1$. Therefore $p^{[i]} = \min\{n^{[i-1]}, m^{[i-1]}\}$ is the number of runs of order i in the line ℓ.

The $p^{[i-1]}$ runs of order $i-1$ must be divided into $p^{[i]} = \min\{n^{[i-1]}, m^{[i-1]}\}$ runs of order i whose lengths can differ only by one. Therefore the number of long runs of order i is $u^{[i]} \equiv p^{[i-1]} \bmod p^{[i]}$. The number of short runs is therefore $v^{[i]} = p^{[i]} - n^{[i]}$.

From definition 2.2.1 for order $i > 1$ and type $t^{[i]} = 0$, $m^{[i]}$ is the number of long runs and $n^{[i]}$ is the number of short runs. Therefore $m^{[i]} = u^{[i]} \equiv p^{[i-1]} \bmod p^{[i]}$ and $n^{[i]} = v^{[i]} = p^{[i]} - m^{[i]}$.

For order $i > 1$ and type $t^{[i]} = 1$, $n^{[i]}$ is the number of long runs and $m^{[i]}$ is the number of short runs. Therefore $n^{[i]} = u^{[i]} \equiv p^{[i-1]} \bmod p^{[i]}$ and $m^{[i]} = v^{[i]} = p^{[i]} - n^{[i]}$. $\square$

The definition of the run length also provides an algebraic method of calculating the run parameters $n^{[i]}$ and $m^{[i]}$ as $u^{[i]} \equiv p^{[i-1]} \bmod p^{[i]} = p^{[i-1]} - \left\lfloor p^{[i-1]} \right\rfloor p^{[1]} p^{[i]}$ and $p^{[i]} = m^{[i]} + n^{[i]}$.

Corollary 2.2.7 *For order 1 we have*

$$n^{[1]} = p^{[0]} - r^{[1]} p^{[1]} \tag{2.14}$$

$$m^{[1]} = (r^{[1]} + 1) p^{[1]} - p^{[0]} \tag{2.15}$$

For order $i \geq 2$ and type $t^{[i]} = 0$

$$m^{[i]} = p^{[i-1]} - r^{[i]} p^{[i]} \tag{2.16}$$

$$n^{[i]} = (r^{[i]}+1)p^{[i]} - p^{[i-1]} \tag{2.17}$$

For order $i \geq 2$ and type $t^{[i]} = 1$

$$n^{[i]} = p^{[i-1]} - r^{[i]}p^{[i]} \tag{2.18}$$

$$m^{[i]} = (r^{[i]}+1)p^{[i]} - p^{[i-1]} \tag{2.19}$$

Termination of the definition of the run parameters $p^{[i_{max}]}, n^{[i_{max}]}$ and $m^{[i_{max}]}$ occurs when $p^{[i_{max}]} = 1$ where i_{max} is called the maximum order of $\mathcal{P}_\ell$. At this order, there is only one run which constitutes the intersection pattern. The length of this run can only be short, $r^{[i]} = p^{[i-1]}$.

We have noted that in definition 2.2.3 the case of $m^{[i]} = n^{[i]}$ was not included.

Lemma 2.2.8 *The parameters $n^{[i]}$ and $m^{[i]}$ cannot have equal value except when $p^{[i]} = 2$.*

Proof : If $\gcd(d, a) = 1$, then by Lemma 4.6.1 $\gcd(m^{[i]}, n^{[i]}) = 1$. Therefore $m^{[i]} = n^{[i]} = 1$. □

Corollary 2.2.9 *If $m^{[i]} = n^{[i]}$, then $i = i_{max} - 1$.*

Therefore if $m^{[i]} = n^{[i]}$, $p^{[i+1]} = 1$ and the definition terminates. Therefore when $m^{[i]} = n^{[i]}$, the type $t^{[i+1]}$ is not defined.

2.2.3 Type and Shape

The type of the runs of order i plays an important role. One of the attributes of type is that it can be used to define the shape of the run of any order.

Theorem 2.2.10 *The shape of a run of order i depends on the type of the run and the type of the runs of order $i-1$ that compose it, in the following manner*

- *If $t^{[i]} = 0$ and $t^{[i-1]} = 0$ the shape of the run is sl^+.*

- *If* $t^{[i]} = 0$ *and* $t^{[i-1]} = 1$ *the shape of the run is* ls^+.

- *If* $t^{[i]} = 1$ *and* $t^{[i-1]} = 0$ *the shape of the run is* s^+l.

- *If* $t^{[i]} = 1$ *and* $t^{[i-1]} = 1$ *the shape of the run is* l^+s.

Proof : By definition 2.2.4 we have,

- if $t^{[i-1]} = 0$ there are $m^{[i-1]}$ long runs of order $i-1$ in the intersection pattern,

- if $t^{[i-1]} = 1$ there are $n^{[i-1]}$ long runs of order $i-1$ in the intersection pattern.

For the case of $t^{[i]} = 0$ we have $n^{[i-1]} < m^{[i-1]}$. If $t^{[i-1]} = 0$ there are more long runs of order $i-1$ in the intersection pattern, therefore the possible shapes are sl^+ and l^+s. If $t^{[i-1]} = 1$ there are more short runs of order $i-1$ in the intersection pattern, therefore the possible shapes are s^+l and ls^+.

For the case of $t^{[i]} = 1$ we have $m^{[i-1]} < n^{[i-1]}$. If $t^{[i-1]} = 0$ there are more short runs of order $i-1$ in the intersection pattern, therefore the possible shapes are s^+l and ls^+. If $t^{[i-1]} = 1$ there are more long runs of order $i-1$ in the intersection pattern, therefore the possible shapes are sl^+ and l^+s.

Assume that if $t^{[i]} = t^{[i-1]} = 0$, the shape is ls^+. Hence the first run of order $i-1$ is long and contains the runs $ls^{r^{[i-1]}}$ of order $i-2$. The slope of this line segment is $\frac{r^{[i-1]}+1}{(r^{[i-2]}+1)+r^{[i-1]}r^{[i-2]}} = \frac{1}{r^{[i-2]}+1}$.

We know that there is more than one run of order $i-2$ in the intersection pattern as $i-2 < i_{max}$. Therefore the slope, in terms of runs of order $i-2$, cannot be $\frac{1}{r^{[i-2]}}$ or $\frac{1}{r^{[i-2]}+1}$ and is bounded by $\frac{1}{r^{[i-2]}+1} < \alpha^{[i-2]} < \frac{1}{r^{[i-2]}}$. Therefore by contradiction we know that if $t^{[i]} = t^{[i-1]} = 0$ then the shape is s^+l and hence, if $t^{[i]} = t^{[i-1]} = 1$, the shape is ls^+. A similar argument follows for the other two cases. $\square$

Corollary 2.2.11 *If type* $t^{[i]} = 0$, *the first run of order* i *is short, If type* $t^{[i]} = 1$, *the first run of order* i *is long.*

It should be noted that the type of order i_{max} is not defined when $r^{[i_{max}]} = 2$. In this case the only shapes possible are ls and sl. Therefore if the type $t^{[i_{max}-1]} = 0$, the shape is sl. If $t^{[i_{max}-1]} = 1$, the shape is ls.

Due to the strength of the relationship between type and shape it will useful to define a numeric definition of shape based on type.

Definition 2.2.5 *The number of the shape is constructed via the binary concatenation of $t^{[i]}$ and $t^{[i-1]}$. Therefore, for*

- *shape sl^+, $s^{[i]} = 00_2 = 0$,*
- *shape ls^+, $s^{[i]} = 01_2 = 1$,*
- *shape s^+l, $s^{[i]} = 10_2 = 2$ and*
- *shape l^+s, $s^{[i]} = 11_2 = 3$.*

There is also a simple correspondence between the length and shape of a run of order i in terms of runs of order $i-1$ and $i-2$.

Definition 2.2.6 *We define two shapes to be of similar composition if number of short and long runs of order $i-1$ is similar in both. The composition $c^{[i]}$ of a run of order i is*

$$c^{[i]} = t^{[i]} \oplus t^{[i-1]} \tag{2.20}$$

Shapes 0 (sl^+) and 3 (l^+s) have composition 0 and shapes 1 (s^+l) and 2 (ls^+) have composition 1.

2.3 The Necessity and Sufficiency of the Run Parameters

The run parameters we have discussed describe the structure of the hierarchy of runs in the intersection pattern of a line and the lattice. What we will now show is that the similarity of run length and type throughout the hierarcies of two digitised lines is a necessary and sufficient condition for the intersection pattern of one line to be contained within the other. That is if the run length $r^{[i]}$ or type $t^{[i]}$ of two lines differ for any order i, the lines generate different intersection patterns over the length of the shorter line and if the run lengths and type of the two lines are equivalent they will generate the same intersection pattern. The only condition we will apply to this result is that the maximum order of the shorter line is greater than one.

Definition 2.3.1 *The intersection pattern $\mathcal{P}_\ell$ of the line $\ell : y = \frac{a}{d}x$ is said to be shorter than the intersection pattern $\mathcal{P}_{\ell'}$ of the line $\ell' : y = \frac{a'}{d'}x$ if $d < d'$. It is said to be longer if $d > d'$ and of equal length if $d = d'$.*

Definition 2.3.2 *The intersection pattern $\mathcal{P}_\ell$ of the line $\ell : y = \frac{a}{d}x$ is said to be contained within the the intersection pattern $\mathcal{P}_{\ell'}$ of the line $\ell' : y = \frac{a'}{d'}x$ if $d \leq d'$, $\mathcal{P}_\ell$ is shorter than or of equal length to $\mathcal{P}_{\ell'}$, and both lines intersect equivalent sets of lattice cells between cells $(0,0)$ and (a,d).*

An observation can be made about the length of an intersection pattern. If $\mathcal{P}_\ell = \mathcal{P}_{\ell'}$ within the frame $\mathcal{L}_{(d,a)}$ where $d < d'$ then $p^{[i]} < p'^{[i]}$ for all orders $i < i_{max}$. Therefore $a < a'$.

Theorem 2.3.1 *Consider the intersection patterns, $\mathcal{P}_{\ell'}$, of the line, $\ell' : y = \frac{a'}{d'}x$, and $\mathcal{P}_\ell$, of the line, $\ell : y = \frac{a}{d}x$ where $d' \leq d$, a and d are coprime and a' and d' are coprime. For $\mathcal{P}_{\ell'}$ to be contained in $\mathcal{P}_\ell$, the run lengths, $r^{[i]}$ and $r'^{[i]}$, and the types, $t^{[i]}$ and $t'^{[i]}$ must be equal for all orders $i < i'_{max}$, the maximum order of the ℓ', where $i'_{max} > 1$. If $i'_{max} = 1$, $\mathcal{P}_{\ell'}$ will be contained in $\mathcal{P}_\ell$ iff $r'^{[i]} \leq r^{[i]} + 1$.*

Proof : If $i'_{max} = 1$, the intersection pattern $\mathcal{P}_{\ell'}$ consists of one run or length $r'^{[i]}$. The first run of order 1 in $\mathcal{P}_\ell$ is long. Therefore if $\mathcal{P}_{\ell'}$ is to be similar to $\mathcal{P}_\ell$ over its length we must have the condition, $r'^{[i]} \leq r^{[i]} + 1$. If $i'_{max} > 1$, the proof is apparent from Lemmas 2.3.2, 2.3.3, 2.3.4 and 2.3.5 which will be provided in the remainder of this section. $\square$

To show the necessity and sufficiency of the run parameters we will consider the intersection patterns of each line to be a set of pixels or runs of order 0, and show under what conditions $\mathcal{P}_{\ell'}$ is a subset of $\mathcal{P}_\ell$. As we are considering the 8–connected intersection pattern, the size of the set is $p^{[0]}$.

We will be using a variety of notation and properties of sets.

- If $\mathcal{P}_{\ell'} = \mathcal{P}_\ell$ both intersection patterns contain identical sets of pixels.
- If $\mathcal{P}_{\ell'} \subset \mathcal{P}_\ell$, the intersection patterns are equivalent over the length of $\mathcal{P}_{\ell'}$.
- If $\mathcal{P}_{\ell'} \subseteq \mathcal{P}_\ell$, then $\mathcal{P}_{\ell'} \subset \mathcal{P}_\ell$ or $\mathcal{P}_{\ell'} = \mathcal{P}_\ell$.
- Where $\emptyset$, is the empty intersection pattern, $\emptyset \subset \mathcal{P}_\ell$.
- If $\mathcal{P}_\ell$ and $\mathcal{P}_{\ell'}$ have the same length, $\mathcal{P}_\ell \subset \mathcal{P}_{\ell'}$ implies $\mathcal{P}_{\ell'} \subset \mathcal{P}_\ell$ and therefore $\mathcal{P}_\ell = \mathcal{P}_{\ell'}$.
- If $\mathcal{P}_\ell$ and $\mathcal{P}_{\ell'}$ have the same length, $\mathcal{P}_\ell \not\subset \mathcal{P}_{\ell'}$ implies $\mathcal{P}_{\ell'} \not\subset \mathcal{P}_\ell$.
- If $\mathcal{P}_\ell \subset \mathcal{P}_{\ell'}$ and $\mathcal{P}_{\ell'} \subset \mathcal{P}_{\ell''}$ then $\mathcal{P}_\ell \subset \mathcal{P}_{\ell''}$.
- If $\mathcal{P}_\ell \not\subset \mathcal{P}_{\ell'}$ and $\mathcal{P}_{\ell'} \subset \mathcal{P}_{\ell''}$ then $\mathcal{P}_\ell \not\subset \mathcal{P}_{\ell''}$.
- If $\mathcal{P}_\ell \subset \mathcal{P}_{\ell'}$ and $\mathcal{P}_\ell \not\subset \mathcal{P}_{\ell''}$ then $\mathcal{P}_{\ell'} \not\subset \mathcal{P}_{\ell''}$.

To abbreviate the description of the elements of a set which defines an intersection pattern, we will replace subsets of pixels by the runs that represent them. For example, for the line $\ell : y = \frac{17}{41}x$, we will say $\mathcal{P}_\ell = \{\ lslslsslslsslslss\ \}^{[[1]]} = \{\ sslslsl\ \}^{[[2]]} = \{\ lss\ \}^{[[3]]} = \{\ s\ \}^{[[4]]}$. The notation $\{\ \ldots\ \}^{[[i]]}$ represents the structure of runs of order i in $\mathcal{P}_\ell$. We will denote the existence of runs by "...". For example, $\{\ ls^r \ldots\ \}^{[[i]]}$, indicates that there are more runs following the initial set of runs $\{\ ls^r\ \}^{[[i]]}$.

If the run sequences, $R_0^{[i]}$ and $R_1^{[i]}$, start with a similar set of runs, the problem can be reduced to the question of containment of the ordered subset of runs from each that starts with the first run that differs. For example if $R_0^{[i]} = \{\ ls^r ls^{r+1} ls^r\ \}^{[[i]]}$ and $R_1^{[i]} = \{\ ls^r ls^r ls^{r+1}\ \}^{[[i]]}$, both sequences start with $\{\ ls^r ls^r\ \}^{[[i]]}$. The problem can be reduced to the question of containment of the sequences $R_0'^{[i]} = \{\ sls^r\ \}^{[[i]]}$ and $R_1'^{[i]} = \{\ ls^{r+1}\ \}^{[[i]]}$.

In the notation we have also defined the order of the runs in the sequences. This is because to prove containment of run sequences of orders above 1 we will be forming inductive proofs. For example let $R_0^{[i]} = \{\ sls\ \}^{[[i]]}$ and $R_1^{[i]} = \{\ ssl\ \}^{[[i]]}$. If the shape of the runs at order i is ls^+ and the length of a short run of order i is $r^{[i]}$ the question of containment can be asked of the run sequences $R_0^{[i-1]} = \{\ ls^{r^{[i]}}ls^{r^{[i]}+1}ls^{r^{[i]}}\ \}^{[[i-1]]}$ and $R_1^{[i-1]} = \{\ ls^{r^{[i]}}ls^{r^{[i]}}ls^{r^{[i]}+1}\ \}^{[[i-1]]}$. This in turn reduces to the question of containment of the subsequences $R_0'^{[i-1]} = \{\ sls^{r^{[i]}}\ \}^{[[i-1]]}$ and $R_1'^{[i-1]} = \{\ ls^{r^{[i]}+1}\ \}^{[[i-1]]}$.

To show the containment of sequences of runs, six properties are sufficient. We derive these properties in Appendix A.

1. $\{\ s\ \}^{[[i]]} \subset \{\ l\ldots\ \}^{[[i]]}$
2. $\{\ l\ \}^{[[i]]} \subset \{\ s\ldots\ \}^{[[i]]}$ iff type $= 0$
3. $\{\ s\ldots\ \}^{[[i]]} \not\subset \{\ l\ldots\ \}^{[[i]]}$
4. $\{\ l\ldots\ \}^{[[i]]} \not\subset \{\ s\ldots\ \}^{[[i]]}$
5. $\{\ sR_0^{[i]}\ \}^{[[i]]} \subset \{\ sR_1^{[i]}\ \}^{[[i]]}$ reduces to $\{\ R_0^{[i]}\ \}^{[[i]]} \subset \{\ R_1^{[i]}\ \}^{[[i]]}$
6. $\{\ lR_0^{[i]}\ \}^{[[i]]} \subset \{\ lR_1^{[i]}\ \}^{[[i]]}$ reduces to $\{\ R_0^{[i]}\ \}^{[[i]]} \subset \{\ R_1^{[i]}\ \}^{[[i]]}$

The first result we will present is that, where the run lengths of the two intersection patterns are equal for all orders, if the types of the two for any order differ the intersection patterns differ.

Lemma 2.3.2 *Let $\mathcal{P}_\ell$ be the intersection pattern of the line $\ell : y = \frac{a}{d}x$ with maximum order i_{max} within the frame, $\mathcal{L}_{(d,a)}$ and let $\mathcal{P}_{\ell'}$ be the intersection pattern of the line $\ell' : y = \frac{a'}{d'}x$ with maximum order $i'_{max} > 1$ within the frame, $\mathcal{L}_{(d',a')}$ where $d' \leq d$. Assume that $r^{[i]} = r'^{[i]}$ for all orders i, $1 \leq i \leq i'_{max}$. If $t^{[i]} = 0$ and $t'^{[i]} = 1$ or $t^{[i]} = 1$ and $t'^{[i]} = 0$ for any order i where $2 < i \leq i'_{max}$ then $\mathcal{P}_{\ell'} \not\subset \mathcal{P}_\ell$.*

Proof : Note that if $p'^{[i'_{max}]} = 1$, $t'^{[i_{max}]}$ is not defined. Therefore if $t'^{[i_{max}]} = 0$ or 1, more than one run of order i'_{max} must exist.

Consider the case of order $i = 2$ or $i > 2$, where $t^{[i-1]} = t'^{[i-1]} = 1$. If $t^{[i]} = 0$ we know the shape is $1 : ls^+$ and the first run of order i in $\mathcal{P}_\ell$ is short. For $t'^{[i]} = 1$, we have shape 3: l^+s and the first run of order i in $\mathcal{P}_{\ell'}$ is long.

If $\mathcal{P}_{\ell'} \subset \mathcal{P}_\ell$ then	$\{\ s \ldots\ \}^{[i]} \subset \{\ l' \ldots\ \}^{[i]}$
which implies	$\{\ ls^{r^{[i]}-1} \ldots\ \}^{[i-1]} \subset \{\ l^{r^{[i]}} s \ldots\ \}^{[i-1]}$
and reduces to	$\{\ s^{r^{[i]}-1} l \ldots\ \}^{[i-1]} \subset \{\ l^{r^{[i]}-1} s \ldots\ \}^{[i-1]}$

As $r^{[i]} \geq 2$, if $r^{[i]} = 2$ this implies that $\{\ sl\ \}^{[i-1]} \subset \{\ ls \ldots\ \}^{[i-1]}$ which contradicts containment property 3. If $r^{[i]} > 2$, it implies that $\{\ ss\ \}^{[i-1]} \subset \{\ ll \ldots\ \}^{[i-1]}$ which also contradicts containment property 3.

If $t^{[i]} = 1$, we have shape 3: l^+s and the first run of order i in $\mathcal{P}_\ell$ is long. For $t'^{[i]} = 0$, we know the shape is $1 : ls^+$ and the first run of order i in $\mathcal{P}_{\ell'}$ is short,

If $\mathcal{P}_{\ell'} \subset \mathcal{P}_\ell$ then	$\{\ l \ldots\ \}^{[i]} \subset \{\ s' \ldots\ \}^{[i]}$
which implies	$\{\ l^{r^{[i]}} s \ldots\ \}^{[i-1]} \subset \{\ ls^{r^{[i]}-1} \ldots\ \}^{[i-1]}$
and reduces to	$\{\ l^{r^{[i]}-1} s \ldots\ \}^{[i-1]} \subset \{\ s^{r^{[i]}-1} l \ldots\ \}^{[i-1]}$

If $r^{[i]} = 2$ this implies that $\{\ ls\ \}^{[i-1]} \subset \{\ sl \ldots\ \}^{[i-1]}$. If $r^{[i]} > 2$, it implies that $\{\ ll\ \}^{[i-1]} \subset \{\ sss \ldots\ \}^{[i-1]}$. Both cases contradict containment property 4.

For order $i > 2$, consider the case where $t^{[i-1]} = t'^{[i-1]} = 0$. If $t^{[i]} = 0$, we know the shape is $0 : sl^+$ and the first run of order i in $\mathcal{P}_\ell$ is short. For $t'^{[i]} = 1$, we have shape 2: s^+l and the first run of order i in $\mathcal{P}_{\ell'}$ is long.

If $\mathcal{P}_{\ell'} \subset \mathcal{P}_\ell$ then	$\{\ s \ldots\ \}^{[i]} \subset \{\ l' \ldots\ \}^{[i]}$
which implies	$\{\ sl^{r^{[i]}-1} \ldots\ \}^{[i-1]} \subset \{\ s^{r^{[i]}} l \ldots\ \}^{[i-1]}$
and reduces to	$\{\ l^{r^{[i]}-1} s \ldots\ \}^{[i-1]} \subset \{\ s^{r^{[i]}-1} l \ldots\ \}^{[i-1]}$

If $r^{[i]} = 2$ this implies that $\{\ sl\ \}^{[i-1]} \subset \{\ ls \ldots\ \}^{[i-1]}$. If $r^{[i]} > 2$, it implies that $\{\ ll\ \}^{[i-1]} \subset \{\ sls \ldots\ \}^{[i-1]}$. Both cases are not possible by properties 3 and 4.

If $t^{[i]} = 1$, we have shape 2: s^+l and the first run of order i in $\mathcal{P}_\ell$ is long. For $t'^{[i]} = 0$, we know the shape is $0 : sl^+$ and the first run of order i in $\mathcal{P}_{\ell'}$ is short,

If $\mathcal{P}_{\ell'} \subset \mathcal{P}_\ell$ then	$\{\ l \ldots\ \}^{[i]} \subset \{\ s' \ldots '\ \}^{[i]}$
which implies	$\{\ s^{r^{[i]}} l \ldots\ \}^{[i-1]} \subset \{\ s l^{r^{[i]}-1} \ldots '\ \}^{[i-1]}$
and reduces to	$\{\ s^{r^{[i]}-1} l \ldots\ \}^{[i-1]} \subset \{\ l^{r^{[i]}-1} \ldots '\ \}^{[i-1]}$

If $r^{[i]} = 2$ this implies that $\{\ sl\ \}^{[i-1]} \subset \{\ ls \ldots\ \}^{[i-1]}$. If $r^{[i]} > 2$, it implies that $\{\ ss\ \}^{[i-1]} \subset \{\ ll \ldots\ \}^{[i-1]}$. Both cases are not possible by property 3.

The lemma is shown. $\square$

Our second result shows that where the types of the two intersection patterns are equal for all orders, if the run lengths differ between the two intersection patterns, the intersection patterns differ.

Lemma 2.3.3 *Let $\mathcal{P}_\ell$ be the intersection pattern of the line $\ell : y = \frac{a}{d}x$ with maximum order i_{max} within the frame, $\mathcal{L}_{(d,a)}$ and $\mathcal{P}_{\ell'}$ be the intersection pattern of the line $\ell' : y = \frac{a'}{d'}x$ with maximum order $i'_{max} > 1$ within the frame, $\mathcal{L}_{(d',a')}$ where $d' \leq d$. Assume that $t^{[i]} = t'^{[i]}$ for all orders i, $2 \leq i \leq i'_{max}$. If $r^{[i]} \neq r'^{[i]}$ for any $i < i'_{max}$ then $\mathcal{P}_{\ell'} \not\subset \mathcal{P}_\ell$.*

Proof :

For shape 0: ls^+ we know that $t^{[i]} = 0$ and the first run in both intersection patterns must be short. Consider the case where $r^{[i]} > r'^{[i]}$.

For shape 0: sl^+	$\{\ s \ldots\ \}^{[i]} \subset \{\ s' \ldots\ \}^{[i]}$
which implies	$\{\ sl^{r^{[i]}-1} sl^{r^{[i]}-1} \ldots\ \}^{[i-1]} \subset \{\ sl^{r'^{[i]}-1} sl^{r'^{[i]}-1} \ldots\ \}^{[i-1]}$
and reduces to	$\{\ l^{r^{[i]}-r'^{[i]}} sl^{r^{[i]}-1} \ldots\ \}^{[i-1]} \subset \{\ sl^{r'^{[i]}-1} \ldots\ \}^{[i-1]}$

If $r^{[i]} - r'^{[i]} = 1$, then the problem reduces to $\{\ lsl^{r^{[i]}-1} \ldots\ \}^{[i-1]} \subset \{\ sl^{r'^{[i]}-1} \ldots\ \}^{[i-1]}$ which implies $\{\ ls\ \}^{[i-1]} \subset \{\ sl\ \}^{[i-1]}$. If $r^{[i]} - r'^{[i]} > 1$, then the problem reduces to $\{\ ll\ \}^{[i-1]} \subset \{\ sll\ \}^{[i-1]}$. Both cases contradict containment property 4. A similar argument follows for $r^{[i]} < r'^{[i]}$.

For shape 1: ls^+ we know that $t^{[i]} = 0$ and the first run in both intersection patterns must be short. Consider the case where $r^{[i]} > r'^{[i]}$.

For shape 1: ls^+ $\quad \{\ s \ldots\ \}^{[i]} \subset \{\ s' \ldots\ \}^{[i]}$

which implies $\quad \{\ ls^{r^{[i]}-1} ls^{r^{[i]}-1} \ldots\ \}^{[i-1]} \subset \{\ ls^{r'^{[i]}-1} ls^{r'^{[i]}-1} \ldots\ \}^{[i-1]}$

and reduces to $\quad \{\ s^{r^{[i]}-r'^{[i]}} ls^{r^{[i]}-1} \ldots\ \}^{[i-1]} \subset \{\ ls^{r'^{[i]}-1} \ldots\ \}^{[i-1]}$

If $r^{[i]} - r'^{[i]} = 1$, then the problem reduces to $\{\ sls^{r^{[i]}-1} \ldots\ \}^{[i-1]} \subset \{\ ls^{r'^{[i]}-1} \ldots\ \}^{[i-1]}$ which implies that $\{\ sl\ \}^{[i-1]} \subset \{\ ls\ \}^{[i-1]}$ If $r^{[i]} - r'^{[i]} > 1$, then the problem reduces to $\{\ ss\ \}^{[i-1]} \subset \{\ ls\ \}^{[i-1]}$ $(ss \subset ls)^{[i-1]}$. Both cases contradict property 3. A similar case can be shown for $r^{[i]} < r'^{[i]}$.

For shape 2: s^+l we know that $t^{[i]} = 1$ and the first run in both intersection patterns must be long. Consider the case where $r^{[i]} > r'^{[i]}$.

For shape 2: s^+l $\quad \{\ l \ldots\ \}^{[i]} \subset \{\ l' \ldots\ \}^{[i]}$

which implies $\quad \{\ s^{r^{[i]}} ls^{r^{[i]}-1} \ldots\ \}^{[i-1]} \subset \{\ s^{r'^{[i]}} ls^{r'^{[i]}-1} \ldots\ \}^{[i-1]}$

and reduces to $\quad \{\ s^{r^{[i]}-r'^{[i]}} ls^{r^{[i]}-1} \ldots\ \}^{[i-1]} \subset \{\ ls^{r'^{[i]}-1} \ldots\ \}^{[i-1]}$

If $r^{[i]} - r'^{[i]} = 1$, then the problem reduces to $\{\ sls^{r^{[i]}-1} \ldots\ \}^{[i-1]} \subset \{\ ls^{r'^{[i]}-1} \ldots\ \}^{[i-1]}$ which implies that $\{\ sl\ \}^{[i-1]} \subset \{\ ls\ \}^{[i-1]}$ If $r^{[i]} - r'^{[i]} > 1$, then the problem reduces to $\{\ ss\ \}^{[i-1]} \subset \{\ ls\ \}^{[i-1]}$. Both of which are not possible. A similar case can be shown for $r^{[i]} < r'^{[i]}$.

For shape 3: l^+s we know that $t^{[i]} = 0$ and the first run in both intersection patterns must be long. Consider the case where $r^{[i]} > r'^{[i]}$.

For shape 3: l^+s $\quad \{\ s \ldots\ \}^{[i]} \subset \{\ s' \ldots\ \}^{[i]}$

which implies $\quad \{\ l^{r^{[i]}} sl^{r^{[i]}-1} \ldots\ \}^{[i-1]} \subset \{\ l^{r'^{[i]}} sl^{r'^{[i]}-1} \ldots\ \}^{[i-1]}$

and reduces to $\quad \{\ l^{r^{[i]}-r'^{[i]}} sl^{r^{[i]}-1} \ldots\ \}^{[i-1]} \subset \{\ sl^{r'^{[i]}-1} \ldots\ \}^{[i-1]}$

If $r^{[i]} - r'^{[i]} = 1$, then the problem reduces to $\{\ lsl^{r^{[i]}-1} \ldots\ \}^{[i-1]} \subset \{\ sl^{r'^{[i]}-1} \ldots\ \}^{[i-1]}$ which implies that $\{\ ls\ \}^{[i-1]} \subset \{\ sl\ \}^{[i-1]}$. If $r^{[i]} - r'^{[i]} > 1$, then the problem reduces to $\{\ ll\ \}^{[i-1]} \subset \{\ sll\ \}^{[i-1]}$. Both of these cases are not possible. A similar argument can be shown for $r^{[i]} < r'^{[i]}$. The lemma is shown. $\square$

For completeness we also need to show that if the run lengths and types differ for some order within the two intersection patterns, so will the patterns.

Lemma 2.3.4 *Let $\mathcal{P}_\ell$ be the intersection pattern of the line $\ell : y = \frac{a}{d}x$ with max-*

imum order i_{max} within the frame, $\mathcal{L}_{(d,a)}$ and $\mathcal{P}_{\ell'}$ be the intersection pattern of the line $\ell' : y = \frac{a'}{d'}x$ with maximum order $i'_{max} > 1$ within the frame, $\mathcal{L}_{(d',a')}$ where $d' \leq d$. If $r^{[i]} \neq r'^{[i]}$ for any $i < i'_{max}$ and if $t^{[i]} = 0$ and $t'^{[i]} = 1$ or $t^{[i]} = 1$ and $t'^{[i]} = 0$ for any order i where $2 < i < i'_{max}$ then $\mathcal{P}_{\ell'} \not\subset \mathcal{P}_{\ell}$.

Proof : Assume that $r^{[j]} = r'^{[j]}$ and $t^{[j]} = t'^{[j]}$ for all orders $1 \leq j < i$.

If $r^{[i]} \neq r'^{[i]}$ and $t^{[i]} \neq t'^{[i]}$.

If $t^{[i]} \neq t'^{[i]}$ this implies that $t^{[i]} = 0$ and $t'^{[i]} = 1$ or $t^{[i]} = 1$ and $t'^{[i]} = 0$. We know that if $t^{[i]} = 0$ the first run in $\mathcal{P}_{\ell}$ must be short and the first run in $\mathcal{P}_{\ell'}$ must be long as $t'^{[i]} = 1$. Therefore if $\mathcal{P}_{\ell'} \subset \mathcal{P}_{\ell}$ then $\{\ l' \ldots\ \}^{[i]} \subset \{\ s \ldots\ \}^{[i]}$.

If $t^{[i-1]} = 0$, $s^{[i]} = 0$ (shape sl^{+}) and $s'^{[i]} = 2$ (shape $s^{+}l$)
this implies that $\{\ s^{r'^{[i]}} l \ldots\ \}^{[i-1]} \subset \{\ s l^{r^{[i]}} \ldots\ \}^{[i-1]}$
which implies $\{\ s^{r'^{[i]}-1} l \ldots\ \}^{[i-1]} \subset \{\ l^{r^{[i]}} \ldots\ \}^{[i-1]}$
and reduces to $\{\ s \ldots\ \}^{[i-1]} \subset \{\ l \ldots\ \}^{[i-1]}$
which is not possible.

If $t^{[i-1]} = 1$, $s^{[i]} = 1$ (shape ls^{+}) and $s'^{[i]} = 3$ (shape $l^{+}s$)
this implies that $\{\ l s^{r'^{[i]}} \ldots\ \}^{[i-1]} \subset \{\ l^{r^{[i]}} s \ldots\ \}^{[i-1]}$
which implies $\{\ s^{r'^{[i]}} \ldots\ \}^{[i-1]} \subset \{\ l^{r^{[i]}-1} \ldots\ \}^{[i-1]}$
and reduces to $\{\ s \ldots\ \}^{[i-1]} \subset \{\ l \ldots\ \}^{[i-1]}$
which is also not possible.

If $t^{[i]} = 1$ the first run in $\mathcal{P}_{\ell}$ must be long and the first run in $\mathcal{P}_{\ell'}$ must be short. Hence if $\mathcal{P}_{\ell'} \subset \mathcal{P}_{\ell}$ then $\{\ s' \ldots\ \}^{[i]} \subset \{\ l \ldots\ \}^{[i]}$.

If $t^{[i-1]} = 0$, $s^{[i]} = 2$ (shape $s^{+}l$) and $s'^{[i]} = 0$ (shape sl^{+})
this implies that $\{\ s l^{r'^{[i]}} \ldots\ \}^{[i-1]} \subset \{\ s^{r^{[i]}} l \ldots\ \}^{[i-1]}$
which implies $\{\ l^{r'^{[i]}} \ldots\ \}^{[i-1]} \subset \{\ s^{r^{[i]}-1} l \ldots\ \}^{[i-1]}$
and reduces to $\{\ l \ldots\ \}^{[i-1]} \subset \{\ s \ldots\ \}^{[i-1]}$
which is not possible.

If $t^{[i-1]} = 1$, $s^{[i]} = 3$ (shape l^+s) and $s'^{[i]} = 1$ (shape ls^+)

this implies that $\{ \, l^{r'^{[i]}} s \ldots \, \}^{[i-1]} \subset \{ \, l s^{r^{[i]}} \ldots \, \}^{[i-1]}$

which implies $\{ \, l^{r'^{[i]}-1} \ldots \, \}^{[i-1]} \subset \{ \, s^{r^{[i]}} \ldots \, \}^{[i-1]}$

and reduces to $\{ \, l \ldots \, \}^{[i-1]} \subset \{ \, s \ldots \, \}^{[i-1]}$

which is also not possible. □

Our final step is to show that if the run lengths and type are similar for all orders, the intersection patterns are the same over the length of the shorter.

Lemma 2.3.5 *Let $\mathcal{P}_\ell$ be the intersection pattern of the line $\ell : y = \frac{a}{d}x$ with maximum order i_{max} within the frame, $\mathcal{L}_{(d,a)}$ and $\mathcal{P}_{\ell'}$ be the intersection pattern of the line $\ell' : y = \frac{a'}{d'}x$ with maximum order $i'_{max} > 1$ within the frame, $\mathcal{L}_{(d',a')}$ where $d' \leq d$.*

If $r^{[i]} = r'^{[i]}$ for all orders $i < i'_{max}$ and if $t^{[i]} = t'^{[i]}$ for any $i < i'_{max}$ then $\mathcal{P}_{\ell'} \not\subset \mathcal{P}_\ell$.

Proof : From the definition of the run parameters, Definition 2.2.4, we know that if the shapes of order i of the two intersection patterns are the same $s^{[i]} = s'^{[i]}$ and the run lengths are the same $r^{[i]} = r'^{[i]}$ the sequence of order $i-1$ runs must be the same. At order i'_{max}, $\mathcal{P}_{\ell'} = \{ \, s \, \}^{[i'_{max}]}$ which is a subset of any definition of $\mathcal{P}_\ell$.

The only complication that may occur is when the type $t'^{[i'_{max}]}$ is not defined. In this case $p'^{[i'_{max}-1]} = 2$ and $\mathcal{P}_{\ell'} = \{ \, sl \, \}^{[i'_{max}-1]}$ or $\{ \, ls \, \}^{[i'_{max}-1]}$. If $t^{[i'_{max}-1]} = t'^{[i'_{max}-1]} = 0$ and $t^{[i'_{max}]} = 0$ then the shape $s^{[i]} = 0 : sl^+$ and the shape $s'^{[i]} = 0$ or $2 : sl$. It is obvious that $\{ \, sl \, \}^{[i'_{max}]} \subset \{ \, sl^+ \ldots \, \}^{[i'_{max}]}$. A similar argument can be given for the remaining cases of shape. □

This final result with the previous three, shows that the similarity of run length and type is a necessary and sufficient condition for two intersection patterns to be equivalent over the length of the shorter.

We can also define another sufficient but not necessary condition for coincidence of intersection pattern sections for two lines with distinct rational slopes.

Theorem 2.3.6 *Consider the lines $\ell : y = \frac{a}{d}x$ and $\ell' : y = \frac{a'}{d'}x$ such that $\frac{a}{d} \neq \frac{a'}{d'}$, a and d are coprime, and $d < d'$. If $|ad' - a'd| < \frac{d'}{d-1}$, then $\mathcal{P}_\ell = \mathcal{P}_{\ell'}$ in $\mathcal{L}_{(d,a)}$.*

Proof : Let h be an integer satisfying $0 \leq h < d$. Since a and d are coprime, we have $ah/d = n + q/d$ such that q is an integer satisfying $0 < q < d$. From $|ad' - a'd| < \frac{d'}{d-1}$ we have

$$ah/d - \frac{h}{(d-1)d} < a'h/d' = ah/d - \frac{(ad'-a'd)h}{dd'} < ah/d + \frac{h}{(d-1)d}$$

Since $0 \leq h \leq d-1$, we get the theorem. □

The converse of Theorem 2.3.6 does not hold.

2.4 Generating the Intersection Pattern

Having defined the structure of the hierarchy of runs of the intersection pattern, $\mathcal{P}_\ell$, we now present a technique for it's generation using a conditional style algorithm at any level within the hierarchy. Thus bridging the disparity which has existed between conditional and structural definitions of the digitised line. The technique describes the length of each individual run of any order in $\mathcal{P}_\ell$ and is based on the values of a sequence of integers, the numerator sequence.

To formally define the numerator sequence we will initally define two other sequences, the run sequence and the intercept sequence. Each of these three sequences are defined for any order within the hierarchy of runs. The run sequence is the sequence of run lengths which exist in an intersection pattern. From the run sequence it is straightforward to define the intercept sequence which contains the fractional component of the intercept of the line and the x coordinate at the end (or beginning) of a run. As we are considering a line with rational slope, in this case the intercept at the end of each run will also be a rational number with a denominator similar to the slope. Therefore the numerator sequence is the rational variant of the intercept sequence where only the numerators of the entries are considered. Let $(x_j^{[i]}, y_j^{[i]})$ be the (x, y) coordinate at the end of the j-th run of order i.

Definition 2.4.1 *The run sequence of the intersection pattern $\mathcal{P}_\ell$ is a sequence of integers $x_0^{[i]} < x_1^{[i]} < \ldots$, such that $x_0^{[i]} = 0$, and for $j \geq 1$,*

$$\lfloor \alpha x_j^{[i]} + \beta \rfloor - \lfloor \alpha x_{j-1}^{[i]} + \beta \rfloor = y_j^{[i]} - y_{j-1}^{[i]} \qquad (2.21)$$

The intercept sequence $\beta_0^{[i]}, \beta_1^{[i]}, \ldots$ *of* ℓ *is defined from the run sequence by*

$$\beta_j^{[i]} = \alpha x_j^{[i]} + \beta - \lfloor \alpha x_j^{[i]} + \beta \rfloor \tag{2.22}$$

The numerator sequence $b_0^{[i]}, b_1^{[i]}, \ldots$ *is defined from the intercept sequence by letting* $\beta_j^{[i]} = b_j^{[i]}/d$.

We will first show how to generate the numerator sequence and then describe how to use the numerator sequence to generate the intersection pattern.

2.4.1 Generating the Numerator Sequence

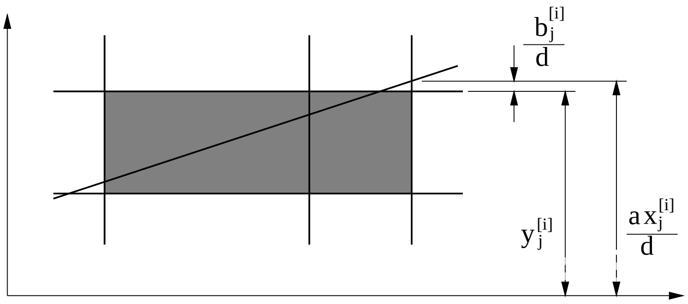


Figure 2.2: The intercept geometry of a run of order i.

Consider the final run of order 1 of any order run described in Figure 2.2. From Definition 2.4.1 we have the following relationship

$$\frac{b_j^{[i]}}{d} = \frac{a x_j^{[i]}}{d} - y_j^{[i]} \tag{2.23}$$

therefore

$$\begin{aligned} b_j^{[i]} &= a x_j^{[i]} - d y_j^{[i]} \\ &= p^{[1]} x_j^{[i]} - p^{[0]} y_j^{[i]} \end{aligned} \tag{2.24}$$

and

$$b_j^{[i]} - b_{j-1}^{[i]} = p^{[1]}(x_j^{[i]} - x_{j-1}^{[i]}) - p^{[0]}(y_j^{[i]} - y_{j-1}^{[i]}) \tag{2.25}$$

where the difference between the two consecutive entries in the numerator sequence in the numerator sequence is measured in runs of order 0 (lattice cells). However what is more useful, is a method of measuring the difference between successive values in the numerator sequence of order i in terms of the structural run parameters and run lengths of order $i-1$. Therefore we provide the following result.

Lemma 2.4.1 *For order 1*

$$b^{[1]}_{j+1} - b^{[1]}_{j} = p^{[1]} r^{[1]}_{j} - p^{[0]} \tag{2.26}$$

For order $i > 1$ and type 0

$$b^{[i+1]}_{j+1} - b^{[i+1]}_{j} = p^{[i]} - r^{[i+1]}_{j} p^{[i+1]} \tag{2.27}$$

For order $i > 1$ and type 1

$$b^{[i+1]}_{j+1} - b^{[i+1]}_{j} = r^{[i+1]}_{j} p^{[i+1]} - p^{[i]} \tag{2.28}$$

Proof : For order 1, the length of a run in pixels is simply the run length and the height of the run is the height of one order 0 run (pixel).

$$\begin{aligned} x^{[1]}_{j} - x^{[1]}_{j-1} &= r^{[1]}_{j} \\ y^{[1]}_{j} - y^{[1]}_{j-1} &= 1 \end{aligned}$$

From Equation 2.25

$$\begin{aligned} b^{[1]}_{j+1} - b^{[1]}_{j} &= p^{[1]}(x^{[1]}_{j+1} - x^{[1]}_{j}) - p^{[0]}(y^{[1]}_{j+1} - y^{[1]}_{j}) \\ &= p^{[1]} r^{[1]}_{j} - p^{[0]} \end{aligned}$$

For order 2 we know that $y^{[2]}_{j+1} - y^{[2]}_{j} = r^{[2]}_{j}$. Therefore for order 2 and type $t^{[2]} = 0$ we have shape 1 (ls^{+}), $p^{[2]} = n^{[1]}$ and

$$x^{[2]}_{j+1} - x^{[2]}_{j} = r^{[2]}_{j} r^{[1]} + 1$$

therefore

$$\begin{aligned} b^{[2]}_{j+1} - b^{[2]}_{j} &= p^{[1]}(r^{[2]}_{j} r^{[1]} + 1) - p^{[0]} r^{[2]}_{j} \\ &= p^{[1]} + r^{[2]}_{j}(p^{[1]} r^{[1]} - p^{[0]}) \\ &= p^{[1]} + r^{[2]}_{j}(-n^{[1]}) \\ &= p^{[1]} - r^{[2]}_{j} p^{[2]} \end{aligned}$$

A similar argument can be constructed for type 1.

Assume for order i and type $t^{[i]} = 0$ that $b^{[i]}_{k+1} - b^{[i]}_{k} = p^{[i-1]} - r^{[i]}_{k} p^{[i]}$.

We are seeking the $r^{[i+1]}_{j}$-th next entry in the order i numerator sequence to correspond to a run of order $i+1$ and length $r^{[i+1]}_{j}$.

$$b^{[i+1]}_{j+1} - b^{[i+1]}_{j} = r^{[i+1]}_{j}(p^{[i-1]} - r^{[i]}_{k} p^{[i]})$$

If the type of order $i+1$ is 0 we have shape sl^{+}, $p^{[i+1]} = n^{[i]}$ and $r^{[i+1]}_{j} r^{[i]}_{k} = r^{[i+1]}_{j}(r^{[i]}+1) - 1$ therefore

$$\begin{aligned} b^{[i+1]}_{j+1} - b^{[i+1]}_{j} &= r^{[i+1]}_{j}(p^{[i-1]} - p^{[i]}(r^{[i]}+1)) + p^{[i]} \\ &= r^{[i+1]}_{j}(p^{[i-1]} \bmod p^{[i]} - p^{[i]}) + p^{[i]} \\ &= r^{[i+1]}_{j}(-n^{[i]}) + p^{[i]} \\ &= p^{[i]} - r^{[i+1]}_{j} p^{[i+1]} \end{aligned}$$

A similar argument will show the theorem for the remaining three shapes.

□

With these relationships we will develop two relationships for generating the numerator sequence based on modular arithmetic. From these a sequential and a parallel algorithm can be designed for calculating the numerator sequence and a number of important properties of the numerator sequence can be described. However in the applications we will consider in later chapters, neither of these techniques are used. For practical applications it is more efficent to calculate the next numerator value based on the length of the current run. We will show in the next section that the difference between two consecutive numerator values is always $m^{[i]}$ or $n^{[i]}$ depending on the type and whether the current is long or short. For now we will concentrate of the construction of the numerator sequence based on modular arithmetic which provides a number of insights into the properties of the numerator sequence.

Theorem 2.4.2 *The numerator sequence of order i can be calculated by the following relation*

$$b^{[i]}_{j} \equiv b^{[i]}_{j-1} + m^{[i]} \pmod{p^{[i]}} \tag{2.29}$$

or by the equivalent relation

$$b_j^{[i]} \equiv b_{j-1}^{[i]} - n^{[i]} \pmod{p^{[i]}} \tag{2.30}$$

Proof : Firstly, from Equation 2.4, $p^{[i]} = m^{[i]} + n^{[i]}$, it is apparent that equations 2.29 and 2.30 are equivalent.

$$\begin{aligned} b_j^{[i]} &\equiv b_{j-1}^{[i]} + m^{[i]} \pmod{p^{[i]}} \\ &\equiv b_{j-1}^{[i]} + p^{[i]} - n^{[i]} \pmod{p^{[i]}} \\ &\equiv b_{j-1}^{[i]} - n^{[i]} \pmod{p^{[i]}} \end{aligned}$$

Consider the order 1 numerator sequence. Lemma 2.4.2 gives us the equation

$$b_j^{[1]} - b_{j-1}^{[1]} = p^{[1]} r_{j-1}^{[1]} - p^{[0]}$$

Therefore

$$\begin{aligned} b_j^{[1]} &= b_{j-1}^{[1]} + p^{[1]} r_{j-1}^{[1]} - p^{[0]} \\ &\equiv b_{j-1}^{[1]} - p^{[0]} \pmod{p^{[1]}} \\ &\equiv b_{j-1}^{[1]} - n^{[1]} \pmod{p^{[1]}} \\ &\equiv b_{j-1}^{[1]} + m^{[1]} \pmod{p^{[1]}} \end{aligned}$$

For order $i > 1$ and type 0 from Lemma 2.4.2 we have

$$\begin{aligned} b_j^{[i]} &= b_{j-1}^{[i]} + p^{[i-1]} - r_{j-1}^{[i]} p^{[i]} \\ &\equiv b_{j-1}^{[i]} + p^{[i-1]} \pmod{p^{[i]}} \end{aligned}$$

From Equation 2.10 from Definition 2.2.4, $m^{[i]} \equiv p^{[i-1]} \pmod{p^{[i]}}$ for type 0,

$$b_j^{[i]} \equiv b_{j-1}^{[i]} + m^{[i]} \pmod{p^{[i]}}$$

For order $i > 1$ and type 1 from Lemma 2.4.1 we have

$$\begin{aligned} b_j^{[i]} &= b_{j-1}^{[i]} + r_{j-1}^{[i]} p^{[i]} - p^{[i-1]} \\ &\equiv b_{j-1}^{[i]} - p^{[i-1]} \pmod{p^{[i]}} \end{aligned}$$

and Equation 2.12 from Definition 2.2.4 $n^{[i]} \equiv p^{[i-1]} \pmod{p^{[i]}}$ for type 1,

$$\begin{aligned} b_j^{[i]} &\equiv b_{j-1}^{[i]} - n^{[i]} \pmod{p^{[i]}} \\ &\equiv b_{j-1}^{[i]} + m^{[i]} \pmod{p^{[i]}} \end{aligned}$$

□

From this result, the second technique for calculating the numerator sequence based on modular arithmetic immediately follows.

Corollary 2.4.3 *The numerator sequence of order i can be calculated by the following relation*

$$b_j^{[i]} \equiv b_0^{[i]} + jm^{[i]} \pmod{p^{[i]}} \tag{2.31}$$

or by the equivalent relation

$$b_j^{[i]} \equiv b_0^{[i]} - jn^{[i]} \pmod{p^{[i]}} \tag{2.32}$$

From these results we can also define two important properties of the numerator sequence. The first is that each value in the numerator sequence of order i, $b_j^{[i]} < p^{[i]}$, where $j = 0, \ldots, p^{[i]} - 1$.

Corollary 2.4.4 *Every term in the numerator sequence of order i, $b_j^{[i]} \in \{0, \ldots, p^{[i]} - 1\}$.*

Corollary 2.4.5 *The $p^{[i]}$ entries of the numerator sequence of order i of the line $y = \frac{a}{d}x$ where a and d are coprime, each have a unique value from the range $b = 0, \ldots, p^{[i]} - 1$.*

Proof : Assume that $b_j^{[i]} = b_{j+k}^{[i]}$. From Corollary 2.4.3 $\gcd(k, p^{[i]}) = p^{[i]}$ since $b_j^{[i]} \equiv jm^{[i]} \equiv (j+k)m^{[i]} \pmod{p^{[i]}}$. There are $p^{[i]}$ values in the numerator sequence of order i hence the corollary is shown. □

It should be noted that in the development of this work we have placed the restriction on the slope of the line, $\frac{a}{d}$, that a and d are coprime (ie. $\gcd(a, d) = 1$) because the intersection pattern of the line $y = \frac{ga}{gd}x$ where $g \in \mathcal{N}$ contains $\mathcal{P}_\ell$ repeated g times [49]. In practise, this restriction can be relaxed as the numerator sequence will also be repeated g times within the frame $\mathcal{L}_{(gd,ga)}$ and each value within of the numerator sequence will have the common divisor of g. This fact is mirrored in the proof to Corollary 2.4.5.

Therefore we have two techniques for calculating the numerator sequence which can be used in sequential applications. The second is also useful for parallel applications. There is a third technique which we will now present which uses only additive integer arithmetic. It is this technique that we will use in the applications we present, to calculate the next value of the numerator sequence based on the length of the current run.

Theorem 2.4.6 *Two consecutive values in the numerator sequence have the following relationships*

- *for order $i = 1$,*
  - *if the j-th run is short, $b^{[1]}_{j+1} = b^{[1]}_j - n^{[1]}$*
  - *if the j-th run is long, $b^{[1]}_{j+1} = b^{[1]}_j + m^{[1]}$*
- *for order $i > 1$, type $t^{[i]} = 0$,*
  - *if the j-th run is short, $b^{[i]}_{j+1} = b^{[i]}_j + m^{[i]}$*
  - *if the j-th run is long, $b^{[i]}_{j+1} = b^{[i]}_j - n^{[i]}$*
- *for order $i > 1$, type $t^{[i]} = 1$,*
  - *if the j-th run is short, $b^{[i]}_{j+1} = b^{[i]}_j - n^{[i]}$*
  - *if the j-th run is long, $b^{[i]}_{j+1} = b^{[i]}_j + m^{[i]}$*

Proof : For order $i = 1$ and order $i > 1$, type $t^{[i]} = 1$ we have from Lemma 2.4.1 $b^{[i]}_{j+1} = b^{[i]}_j + p^{[i]}r^{[i]}_j - p^{[i-1]}$. If the j-th run is short, $r^{[i]}_j = r^{[i]}$, $b^{[i]}_{j+1} = b^{[i]}_j - n^{[i]}$. If the j-th run is long, $r^{[i]}_j = r^{[i]} + 1$, $b^{[i]}_{j+1} = b^{[i]}_j + m^{[i]}$.

For order $i > 1$, type $t^{[i]} = 0$ we have from Lemma 2.4.1 $b^{[i]}_{j+1} = b^{[i]}_j - p^{[i]}r^{[i]}_j + p^{[i-1]}$. If the j-th run is short, $r^{[i]}_j = r^{[i]}$, $b^{[i]}_{j+1} = b^{[i]}_j + m^{[i]}$. If the j-th run is long, $r^{[i]}_j = r^{[i]} + 1$, $b^{[i]}_{j+1} = b^{[i]}_j - n^{[i]}$. □

2.4.2 Generating the Intersection Pattern from the Numerator Sequence

The numerator sequence can be used to generate the intersection pattern, $\mathcal{P}_\ell$, by determining whether the length of each run of any order in $\mathcal{P}_\ell$ is long or short based on the relative values of two consecutive entries of the sequence or the value of a single numerator value relative to the run parameters $m^{[i]}$ and $n^{[i]}$. We will therefore present three techniques for generating the intersection pattern from the numerator sequence. For each of these techniques, if the j-th run of order i is short, $r^{[1]}_j = r^{[1]}$ and if it is long $r^{[1]}_j = r^{[1]} + 1$.

Theorem 2.4.7 *The length of the j-th run can be calculated from the values of the j-th and $j+1$-th numerator values, $b^{[i]}_j$ and $b^{[i]}_{j+1}$, by the following cases.*

- *For order 1,*
  - *the j-th run is short, if $b^{[1]}_j > b^{[1]}_{j+1}$*
  - *the j-th run is long, if $b^{[1]}_j < b^{[1]}_{j+1}$*
- *For order $i \geq 2$ and type 0 ($m^{[i-1]} > n^{[i-1]}$):*
  - *the j-th run is short, if $b^{[i]}_j < b^{[i]}_{j+1}$*
  - *the j-th run is long, if $b^{[i]}_j > b^{[i]}_{j+1}$*
- *For order $i \geq 2$ and type 1 ($m^{[i-1]} < n^{[i-1]}$):*
  - *the j-th run is short, if $b^{[i]}_j > b^{[i]}_{j+1}$*
  - *the j-th run is long, if $b^{[i]}_j < b^{[i]}_{j+1}$*

Proof : For order $i > 1$, type $t^{[i]} = 0$, we have $p^{[i]} = n^{[i-1]} < n^{[i-1]}$ and from Equation 2.27

$$b_{j+1}^{[i]} - b_j^{[i]} = p^{[i-1]} - r_j^{[i]} p^{[i]}$$

Therefore

$$r_j^{[i]} = \frac{p^{[i-1]}}{p^{[i]}} - \frac{b_{j+1}^{[i]} - b_j^{[i]}}{p^{[i]}}$$

Write

$$\frac{p^{[i-1]}}{p^{[i]}} = r_j^{[i]} + u$$

We know $r_j^{[i]} = r^{[i]}$ or $r^{[i]} + 1$ where $r^{[i]} = \left\lfloor \frac{p^{[i-1]}}{p^{[i]}} \right\rfloor$. Therefore $u < 1$ hence

$$\left| \frac{b_{j+1}^{[i]} - b_j^{[i]}}{p^{[i]}} \right| < 1$$

this implies that

$$-1 < \frac{b_{j+1}^{[i]} - b_j^{[i]}}{p^{[i]}} + u < 2$$

which implies that either

$$\begin{array}{lll} r_j^{[i]} = \left\lfloor \frac{p^{[i]}}{p^{[i-1]}} \right\rfloor & \text{when} & b_{j+1}^{[i]} < b_j^{[i]} \quad \text{or} \\ r_j^{[i]} = \left\lfloor \frac{p^{[i]}}{p^{[i-1]}} \right\rfloor + 1 & \text{when} & b_{j+1}^{[i]} > b_j^{[i]} \end{array}$$

Therefore the first claim for type 0 is shown. A similar argument extends to type 1 and order 1. □

From this theorem we provide the alternative techniques for determining the length of the runs in $\mathcal{P}_\ell$ based on a single numerator value.

Theorem 2.4.8 *The length of the j-th run can be calculated from the value of the j-th numerator value, $b_j^{[i]}$, by the following cases.*

- *For order 1,*
  - *the j-th run is short if $b_j^{[1]} \geq n^{[1]}$,*
  - *the j-th run is long if $b_j^{[1]} < n^{[1]}$,*
- *For order $i \geq 2$ and type 0*

- *the j-th run is long if* $b_j^{[i]} \geq n^{[i]}$,
- *the j-th run is short if* $b_j^{[i]} < n^{[i]}$,

- *For order* $i \geq 2$ *and type 1*
  - *the j-th run is short if* $b_j^{[i]} \geq n^{[i]}$,
  - *the j-th run is long if* $b_j^{[i]} < n^{[i]}$,

Proof : We have from Theorem 2.4.6 for order $i = 1$ and order $i > 1$, type $t^{[i]} = 1$ if the j-th run is short, $b_{j+1}^{[i]} = b_j^{[i]} - n^{[i]}$ and if the j-th run is long, $b_{j+1}^{[i]} = b_j^{[i]} + m^{[i]}$. Let $r_j^{[i]}$ be long and $b_j^{[i]} \geq n^{[i]}$. Therefore $b_{j+1}^{[i]} = b_j^{[i]} + m^{[i]} \geq p^{[i]}$ which contradicts Corollary 2.4.4. All other cases can be shown in a similar manner. □

Corollary 2.4.9 *The length of the j-th run can be calculated from the value of the j-th numerator value,* $b_j^{[i]}$ *by the following cases.*

- *For order 1,*
  - *the j-th run is short, if* $b_{j+1}^{[1]} < m^{[1]}$,
  - *the j-th run is long, if* $b_{j+1}^{[1]} \geq m^{[1]}$,
- *For order* $i \geq 2$ *and type 0*
  - *the j-th run is long, if* $b_{j+1}^{[i]} < m^{[i]}$,
  - *the j-th run is short, if* $b_{j+1}^{[i]} \geq m^{[i]}$,
- *For order* $i \geq 2$ *and type 1*
  - *the j-th run is short, if* $b_{j+1}^{[i]} < m^{[i]}$,
  - *the j-th run is long, if* $b_{j+1}^{[i]} \geq m^{[i]}$,

We can therefore very efficiently calculate the length of a run based on a single value of the numerator sequence. It is these two techniques, primarily Theorem 2.4.8 which we shall use in our discussion of the application of the run–based hierarchy of the digitised line to techniques in computer graphics.

2.5 The Calculation of the Run Parameters and Numerator Sequence

We now present algorithms to calculate the run parameters and the numerator sequence for any order.

2.5.1 Calculating the Run Parameters

The structural run parameters, $p^{[i]}, m^{[i]}, n^{[i]}$ and type $t^{[i]}$ can be calculated using Theorem 2.2.6 and the run length, $r^{[i]}$, can be calculated by Theorem 2.2.4.

Algorithm 2.1 Calculating the run parameters for all orders of the line $y = \frac{a}{d}x$.

{Run parameters for order 0}
if $d = 1$ **return**
$p^{[0]} = d$

{Run parameters for order 1}
if $a = 1$ **return**
$p^{[1]} = a$
$r^{[1]} = \lfloor \frac{p^{[0]}}{p^{[1]}} \rfloor$
$n^{[1]} \equiv p^{[0]} \pmod{p^{[1]}}$
$m^{[1]} = p^{[1]} - n^{[1]}$

{Run parameters for order $i > 2$}
$i = 2$
while $p^{[i-1]} > 1$
    if $m^{[i-1]} > n^{[i-1]}$
        $t^{[i]} = 0$
        $p^{[i]} = n^{[i-1]}$
        $m^{[i]} \equiv p^{[i-1]} \pmod{p^{[i]}}$
        $n^{[i]} = p^{[i]} - m^{[i]}$
    else
        $t^{[i]} = 1$
        $p^{[i]} = m^{[i-1]}$
        $n^{[i]} \equiv p^{[i-1]} \pmod{p^{[i]}}$
        $m^{[i]} = p^{[i]} - n^{[i]}$
    $r^{[i]} = \lfloor \frac{p^{[i-1]}}{p^{[i]}} \rfloor$

Theorem 2.2.6 depends heavily on integer modular arithmetic. The modulus operator is difficult to implement in hardware and on current systems performs slowly when compared with other operators. Therefore we present a second technique based on the algebraic relations derived in Corollary 2.2.7 in which only integer addition and multiplication are required to calculate the run parameters $m^{[i]}$ and $n^{[i]}$.

Algorithm 2.2 Calculating the run parameters for all orders of the line $y = \frac{a}{d}x$.

$\{$Run parameters for order 0$\}$
if $d = 1$ **return**
$p^{[0]} = d$

$\{$Run parameters for order 1$\}$
if $a = 1$ **return**
$p^{[1]} = a$
$r^{[1]} = \lfloor \frac{p^{[0]}}{p^{[1]}} \rfloor$
$n^{[1]} = p^{[0]} - r^{[1]} p^{[1]}$
$m^{[1]} = p^{[1]} - n^{[1]}$

$\{$Run parameters for order $i > 2\}$
$i = 2$
while $p^{[i-1]} > 1$
    if $m^{[i-1]} > n^{[i-1]}$
        $t^{[i]} = 0$
        $p^{[i]} = n^{[i-1]}$
        $r^{[i]} = \lfloor \frac{p^{[i-1]}}{p^{[i]}} \rfloor$
        $m^{[i]} = p^{[i-1]} - r^{[i]} p^{[i]}$
        $n^{[i]} = p^{[i]} - m^{[i]}$
    else
        $t^{[i]} = 1$
        $p^{[i]} = m^{[i-1]}$
        $r^{[i]} = \lfloor \frac{p^{[i-1]}}{p^{[i]}} \rfloor$
        $n^{[i]} = p^{[i-1]} - r^{[i]} p^{[i]}$
        $m^{[i]} = p^{[i]} - n^{[i]}$

2.5.2 Calculating the Numerator Sequence and Intersection Pattern

The following algorithms describe the calculation of the numerator sequence and the intersection pattern, $\mathcal{P}_\ell$, of the line $\ell : y = \frac{a}{d}x$. The first algorithm is based on

Theorems 2.4.2 and 2.4.7.

Algorithm 2.3 Calculating the numerator sequence for all orders and the intersection pattern $\mathcal{P}_\ell$ of the line $y = \frac{a}{d}x$.

```
{Numerator sequence for order 1}
b_0^{[1]} = 0
do
    b_{j+1}^{[1]} ≡ b_j^{[1]} + m^{[1]}  (mod p^{[1]})
    if b_j^{[1]} < b_{j+1}^{[1]}
        r_j^{[1]} = r^{[1]} + 1
    else
        r_j^{[1]} = r^{[1]}
    while b_{j+1}^{[1]} > 0

{Numerator sequence for higher order runs}
i = 2
b_0^{[i]} = 0
do
    b_{j+1}^{[i]} ≡ b_j^{[i]} + m^{[i]}  (mod p^{[i]})
    if t^{[i]} = 0
        if b_j^{[i]} < b_{j+1}^{[i]}
            r_j^{[i]} = r^{[i]}
        else
            r_j^{[i]} = r^{[i]} + 1

    else t^{[i]} = 1
        if b_j^{[i]} < b_{j+1}^{[i]}
            r_j^{[i]} = r^{[i]} + 1
        else
            r_j^{[i]} = r^{[i]}
        while b_{j+1}^{[i]} > 0
i = i + 1
```

It is however more prudent to use Theorem 2.4.8 which decides the length of a run based on a single numerator value. The values of the numerator sequence can also be altered without changing their relative values so the test can be performed against zero. The decision of run length can therefore be implemented as a simple sign check. The calculation of the next numerator value can be made using Theorem 2.4.6.

Algorithm 2.4 Calculating the numerator sequence for all orders and the intersection pattern $\mathcal{P}_\ell$ of the line $y = \frac{a}{d}x$.

```
{Numerator sequence for order 1}
b_0^{[1]} = -n^{[1]}
do
    if b_j^{[1]} < 0
        b_{j+1}^{[1]} = b_j^{[1]} + m^{[1]}
        r_j^{[1]} = r^{[1]} + 1
    else
        b_{j+1}^{[1]} = b_j^{[1]} - n^{[1]}
        r_j^{[1]} = r^{[1]}
    while b_{j+1}^{[1]} > -n^{[1]}

{Numerator sequence for higher order runs}
i = 2
b_0^{[i]} = -n^{[i]}
do
    if t^{[i]} = 0
        if b_j^{[i]} < 0
            b_{j+1}^{[i]} = b_j^{[i]} + m^{[i]}
            r_j^{[i]} = r^{[i]}
        else
            b_{j+1}^{[i]} = b_j^{[i]} - n^{[i]}
            r_j^{[i]} = r^{[i]} + 1

    else t^{[i]} = 1
        if b_j^{[i]} < 0
            b_{j+1}^{[i]} = b_j^{[i]} + m^{[i]}
            r_j^{[i]} = r^{[i]} + 1
        else
            b_{j+1}^{[i]} = b_j^{[i]} - n^{[i]}
            r_j^{[i]} = r^{[i]}
        while b_{j+1}^{[i]} > -n^{[i]}
i = i + 1
```

2.6 Examples

We provide two examples of the intersection patterns of the lines $y = \frac{17}{41}x$ and $y = \frac{29}{41}x$. Within the examples the relationship between the numerator sequences and the individual run lengths can be seen. The run parameters for all orders less than the maximum order are also presented.

2.6.1 The Example of $y = \frac{17}{41}x$.

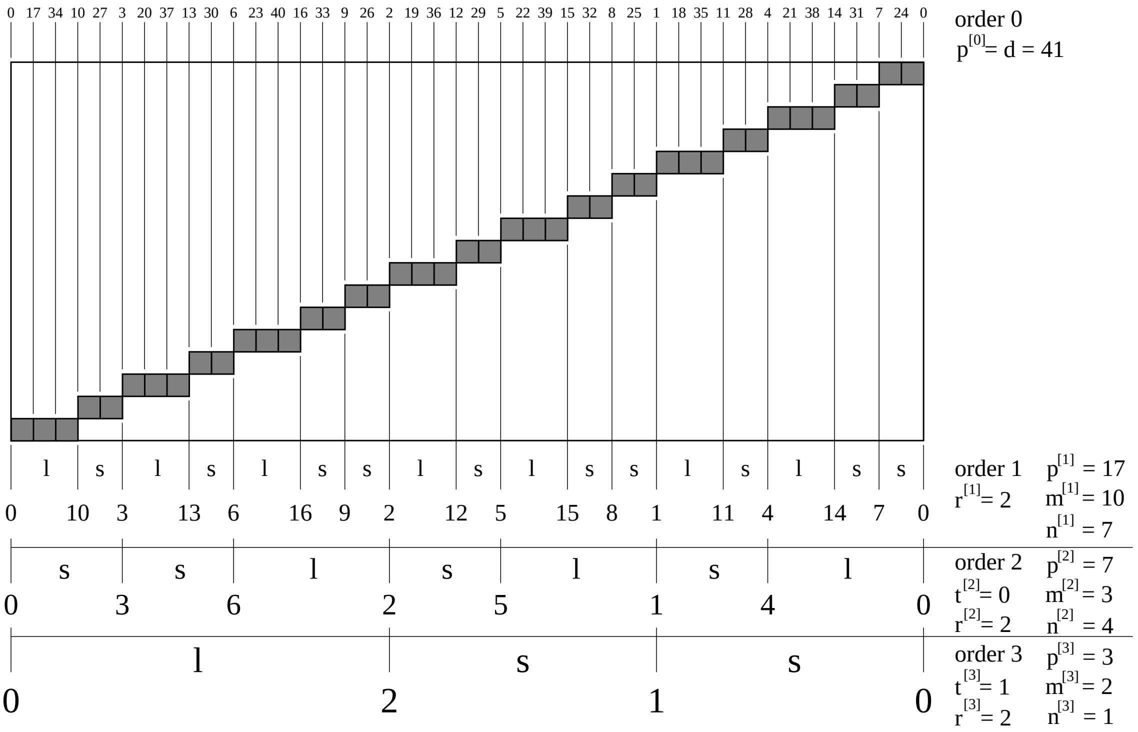


Figure 2.3: The construction of runs and the numerator sequence of all orders in the line $y = \frac{17}{41}x$.

Consider the run parameters and numerator sequence of order 2. From Definition 2.2.4, there are $p^{[2]} = 7$ runs of order 2 in the intersection pattern providing a sequence of run lengths $sslslsl$. The type of this order is $t^{[i]} = 0$ therefore there are $m^{[2]} = 3$ long runs and $n^{[2]} = 4$ short runs. The value of $m^{[2]} < n^{[2]}$ therefore the numerator sequence comprises subsequences of increasing values with a difference of $m^{[2]}$ which correspond the sequences of short runs.

2.6.2 The Example of $y = \frac{29}{41}x$.

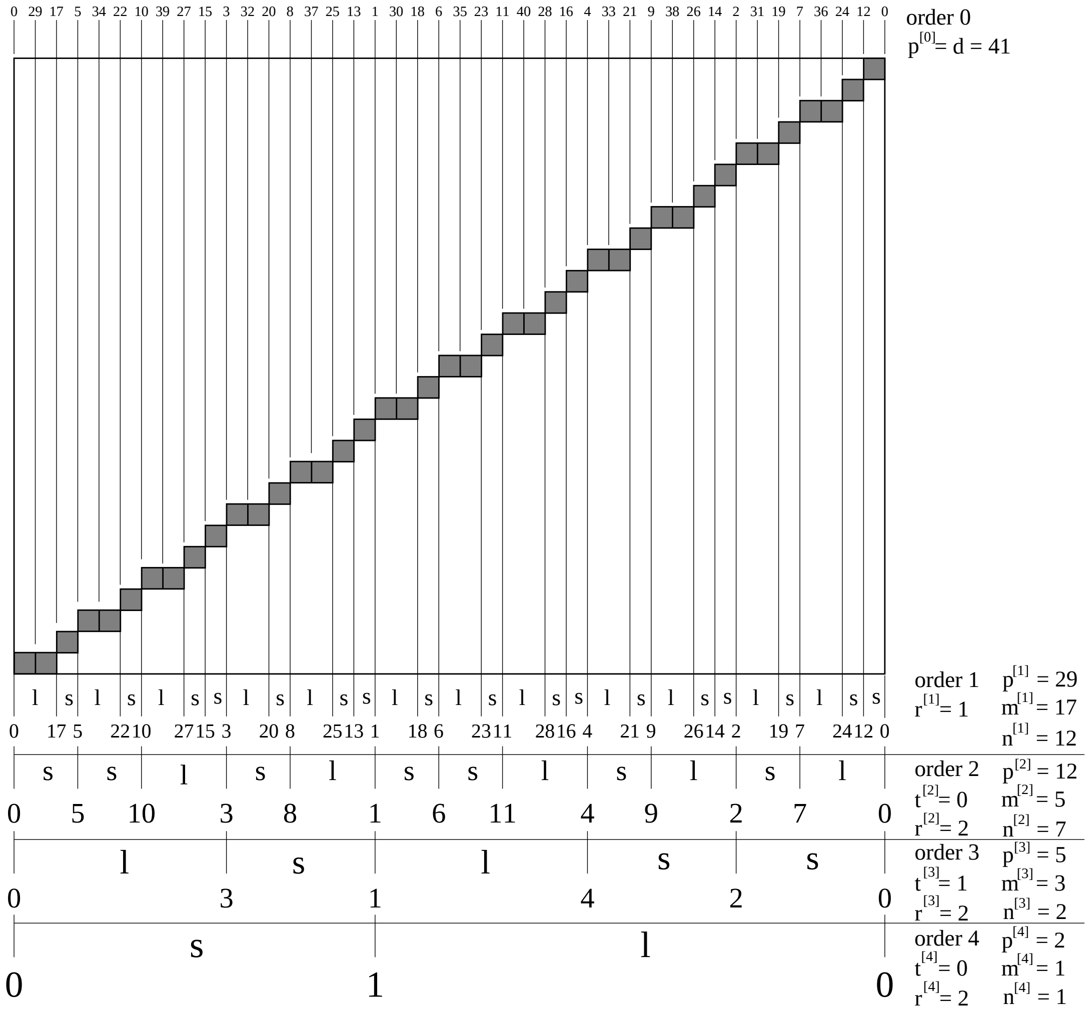


Figure 2.4: The construction of runs and the numerator sequence of all orders in the line $y = \frac{29}{41}x$.

In this example note the $m^{[4]} = n^{[4]} = 1$ and no type would be defined for order 5, the maximum order of the intersection pattern.

2.7 Conclusions

By extending the definition of a run to include pixels, runs and runs of runs, we have formally defined the hierarchical structure of runs within the intersection pattern of a line and the lattice. We have also formally extended results presented by Rosenfeld

[111] on the nature of the run lengths of every order in the intersection pattern. This was necessary as the work presented by Dorst and Duin [45] which confirms Rosenfeld's observation will be shown in Chapter 4 to have a set of degenerate cases. We have provided a set of parameters to describe the hierarchy of runs which are necessary and sufficient.

To generate the intersection pattern, we have provided a technique based on the sequence of numerators of the intercept values defined at the beginning of each run of order i. The definition of the numerator sequence allows for the length of each run to be calculated. A conditional style algorithm is presented to generate the intersection pattern based on runs of any order. This result effectively bridges the gap which has existed between structurally–based and conditionally–based descriptions of the intersection pattern [22].

Chapter 3

The Intersection Pattern of a Line with a Real Slope and Nonzero Intercept

3.1 Introduction

In the previous chapter we described the structure of the hierarchy of runs in the intersection pattern, $\mathcal{P}_{\ell'}$ of the line with rational slope and zero intercept, $\ell' : y = \frac{a}{d}$ where $(a, d) \in \mathcal{N}$, $0 < a < d$ and a and d are coprime and the lattice. We have also provided a technique to generate the intersection pattern based on the numerator sequence. In this chapter we will extend these ideas for the line with a real slope and non-zero real intercept, $\ell : y = \alpha x + \beta$.

To complete this aim, we will be considering three cases. Firstly, we will describe the effect of a non-zero intercept on a line with rational slope therefore completing the treatment of this case. Our second step will be to show that within a bounded frame or region of the lattice, a line with irrational slope and zero intercept can be approximated by a line with rational slope such that both lines produce identical intersection patterns within the frame. We can then use our results from the treatment of a line with rational slope and nonzero intercept to describe the effect of introducing an intercept to a line of irrational slope within the frame. We will have then provided a complete description of the structure of runs within the intersection pattern of the line and the lattice.

3.2 The Effect of a Non–Zero Intercept on a Line with Rational Slope

Introducing a non-zero intercept to a line with rational slope presents a number of complications. Consider the line $\ell : y = \frac{a}{d}x + \beta$ where a and d are coprime. Within the frame of the lattice $\mathcal{L}_{(d,a)}$ the line ℓ forms an intersection pattern that is repeated throughout the infinite intersection pattern defined upon the unbounded lattice, [49]. Therefore we will only consider the intersection pattern, $\mathcal{P}_\ell$, within the frame $\mathcal{L}_{(d,a)}$.

Let us consider first the line $\ell : y = \frac{a}{d}x + \beta$ where $\beta = 0$. Within the frame $\mathcal{L}_{(d,a)}$, the vertical distance the line ℓ is above any lattice point within the intersection pattern is $\frac{b_j^{[0]}}{d}$ where $b_j^{[0]} = 0, \ldots, d-1$ where the values $b_j^{[0]}$ are those values of the order 0 numerator sequence. Therefore the line is at least $\frac{1}{d}$ from any lattice point. If the value of β is raised slowly, the intersection pattern will remain unchanged until the line crosses or intersects a lattice point. Therefore the first change will occur when $\beta = \frac{1}{d}$. Let the lattice point intersected be (x_h, y_h).

As the line $\ell' : y = \frac{a}{d}x + \frac{1}{d}$ intersects the point (x_h, y_h) and the line $\ell : y = \frac{a}{d}x$ intersects the origin, the intersection pattern $\mathcal{P}_{\ell'}$ of the line ℓ' will be identical to the intersection pattern $\mathcal{P}_\ell$ of the line ℓ translated by (x_h, y_h).

Therefore the introduction of an intercept to a line with rational slope will, more often than not, cause a shifting of the intersection pattern of the line. The key values of the intercept at which this translation of the intersection pattern occurs are $\beta = \frac{b}{d}$ where $b = 0, \ldots, d-1$. A similar result was presented by Dorst and Duin [45]. The first result we present formalises the argument we have presented.

Theorem 3.2.1 *The intersection pattern, $\mathcal{P}_{\ell'}$, of the line $\ell' : y = \frac{a}{d}x + \frac{b}{d}$ is identical to the intersection pattern, $\mathcal{P}_{\ell}$, of the line $\ell : y = \frac{a}{d}x + \beta$ where $\beta \in \mathcal{R}$, if $0 \le \beta - \frac{b}{d} < \frac{1}{d}$.*

Proof : Define $\beta' = \frac{b}{d}$ by $\beta = \beta' + \rho$, such that $0 \le \rho < \frac{1}{d}$. Note that if $\beta_0, \dots$ and $\beta'_0, \dots$ are the intercept sequences of ℓ and ℓ', then clearly $\beta_i - \beta_{i-1} = \beta'_i - \beta'_{i-1}$. Therefore ℓ and ℓ' will have the same 8–connected intersection pattern. $\Box$

Therefore the line $\ell : y = \frac{a}{d}x + \beta$ and the line $\ell' : y = \frac{a}{d}x + \frac{b}{d}$, have the same intersection pattern where

$$b = \lfloor \beta d \rfloor \tag{3.1}$$

The question of the existence of corner intersections is important to applications such as ray tracing. Being able to predict and calculate their occurrence is extremely useful in many cases to improve the efficiency of the technique. For a line with rational slope, a corner intersection can only occur when the slope and intercept have the same denominator and when the associated numerator sequence value $b_j^{[i]} = 0$.

Lemma 3.2.2 *The intersection pattern, $\mathcal{P}_{\ell}$, of the line $\ell : y = \frac{a}{d}x + \beta$ will contain a corner intersection for those values of $\beta = \frac{b}{d}$ where $b = 0, \dots, d-1$. The corner intersection, if it occurs, occurs at the point of the lattice associated with the numerator sequence value of $b_j^{[i]} = 0$.*

Proof : Define $\beta' = \frac{b}{d}$ by $\beta = \beta' + \rho$, such that $0 \le \rho < \frac{1}{d}$. For a corner intersection to occur $y = \frac{a}{d}x + \frac{b}{d} + \rho$ must be an integer for some value of x. This can only occur if $\rho = 0$.

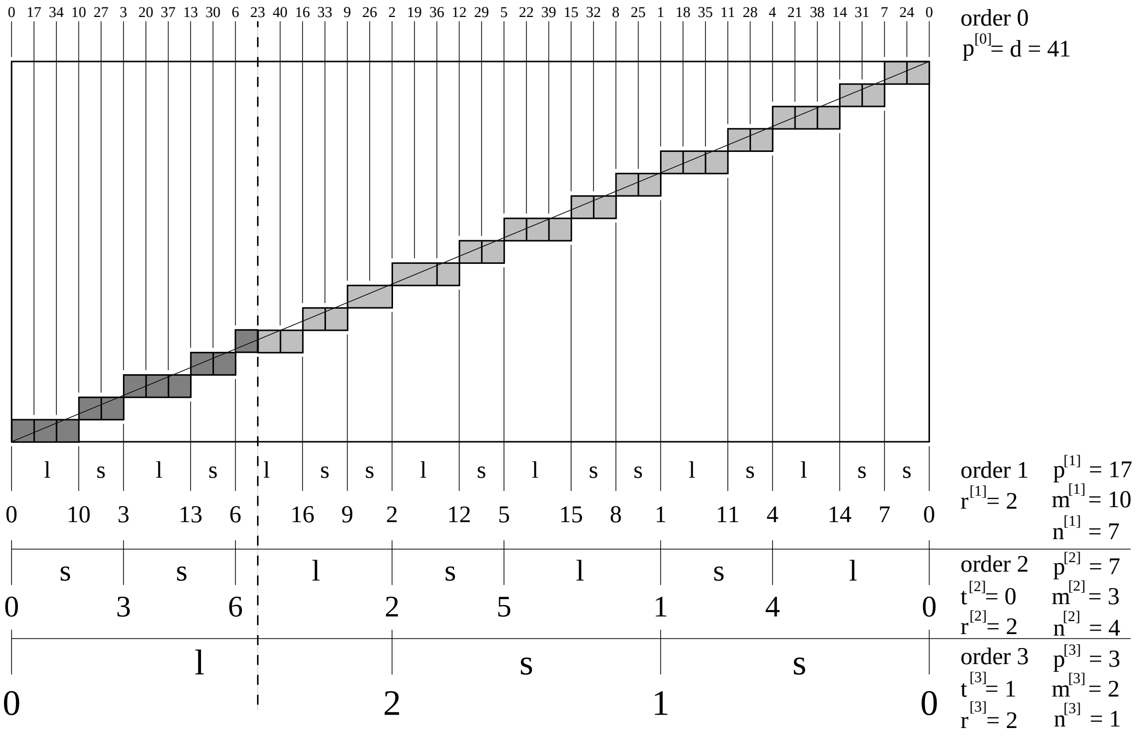
By the definition of the numerator sequence, Definition 2.4.1, a corner intersection can occur only at the value in the numerator sequence $b_j^{[i]} = 0$. By Theorem 2.4.5 we know that each value of the numerator sequence of order i has a unique value from the range $k = 0, \dots, p^{[i]} - 1$. Therefore a value of $b_j^{[i]} = 0$ must occur with the frame $\mathcal{L}_{(d,a)}$. $\Box$

Consider the example of introducing a non-zero intercept to a line $\ell : y = \frac{17}{41}x$. Figure 3.1(a) displays the intersection pattern of the line ℓ. Figure 3.1(b) displays the effect on the intersection pattern of introducing an intercept value of $\frac{23}{41}$.

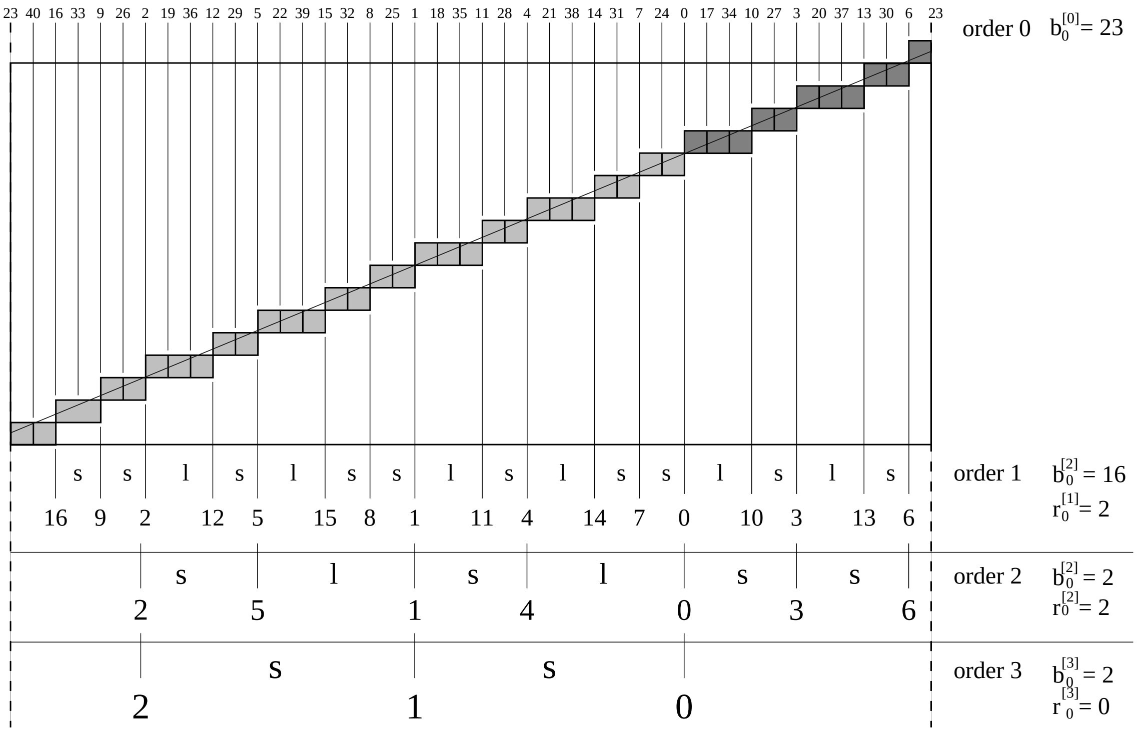
Given that the numerator of the intercept is $b = 23$, we have demarcated the pixels before and after this value in the order 0 numerator sequence in Figure 3.1(a) by a dashed line. The pixels to the right of this value are denoted by light grey, the pixels to the left by dark grey. In Figure 3.1(b) we can see that the dark grey pixels are shifted to the beginning of the intersection pattern, $\mathcal{P}_\ell$. The light grey pixels form the end of the pattern $\mathcal{P}_\ell$. Coinciding with the shift in pixels, the values of the numerator sequence of order 0 are also shifted and now start with a value of $b_0^{[0]} = b = 23$. The shifting of the runs of all orders in the intersection pattern with the introduction of a non-zero intercept is mimicked by the shift in the values of the numerator sequence.

The shifting of the intersection pattern, due to the introduction of a non-zero intercept, has a number of side effects. The initial and final run of any order may be truncated. This occurs when the numerator of the intercept is not in the numerator sequence of that order. A run order is split and forms the initial and final partial run. If the numerator sequence is to be calculated for this order, the initial numerator value will have to be calculated.

For example in Figure 3.1(b), the numerator value of the intercept is $b = 23$. We know that all of the numerator values of order 1 are less than $p^{[1]} = a = 17$ from Corollary 2.4.4. Therefore the initial value of the order 0 numerator sequence will not be the same as the initial value of the order 1 numerator sequence and the initial and final runs of order 1 will be truncated. To calculate the initial value of the order 1 numerator sequence we seek the first value of the order 0 numerator sequence that is less than $p^{[1]}$. From the example, $b_0^{[1]} = b_2^{[0]} = 16$. The number of entries skipped is the length of the initial partial run, $r_0^{[1]} = 2$. From Theorem 2.4.9 we know that the preceding run which was split was long as $b_0^{[1]} = 16 > m^{[1]} = 10$. The length of a short run is $r^{[1]} = 2$. Hence the length of the final partial run, $r_p^{[1]} = 1$.



(a) The intersection pattern of $\ell : y = \frac{17}{41}x$.



(b) The intersection pattern of $\ell' : y = \frac{17}{41}x + \frac{23}{41}$.

Figure 3.1: The effect of a non-zero intercept, $\beta = \frac{23}{41}$, on the intersection pattern of a line with rational slope, $\ell : y = \frac{17}{41}x$.

Definition 3.2.1 *The initial partial run length of order i, $r_0^{[i]}$, is the number of full runs of order $i-1$ before the first full run of order i in the intersection pattern $\mathcal{P}_\ell$.*

Similarly the final partial run length of order i, $r_p^{[i]}$ is the number of full runs of order $i-1$ after the last full run of order i in the intersection pattern $\mathcal{P}_\ell$.

The initial numerator value of order i, $b_0^{[i]}$ is the first numerator value in the numerator sequence of order $i-1$ that is less than $p^{[i]}$.

To describe how to calculate the length of the initial and final partial run lengths and the initial numerator value of order $i \geq 1$, consider the construction of the numerator sequence.

For order i and type $t^{[i]} = 1$ we have $p^{[i]} = m^{[i-1]} < n^{[i]}$. Therefore as $m^{[i-1]} < \frac{p^{[i-1]}}{2}$ the numerator sequence of order $i-1$ will form subsequences of increasing numerator values each differing by $m^{[i-1]}$. The values at the beginning of these sequences correspond to the numerator values in the numerator sequence of order i. Therefore the order i numerator sequence for type $t^{[i]} = 1$ is represented by

$$b_0^{[i-1]}, \ldots, b_j^{[i]}, b_j^{[i]} + m^{[i-1]}, b_j^{[i]} + 2m^{[i-1]}, \ldots, b_j^{[i]} + r_{j-1}^{[i]} m^{[i-1]} - p^{[i-1]} = b_{j+1}^{[i]}, \ldots, b_0^{[i-1]}$$

Where $r_j^{[i]}$ is the j-th run length of order i: the number of order $i-1$ runs in the order i run.

For example of the line $\ell : y = \frac{17}{41}x + \frac{23}{41}$ from Figure 3.1(a), the type of the order 3 runs is $t^{[3]} = 1$. The order 2 numerator sequence can be represented as

$$\begin{aligned}
b_{k=0\to 7}^{[2]} = \{ & b_0^{[3]}, \\
& b_0^{[3]} + m^{[2]}, \\
& b_0^{[3]} + 2m^{[2]}, \\
& b_1^{[3]} = b_0^{[3]} + r_0^{[3]} m^{[2]} - p^{[2]}, \\
& b_1^{[3]} + m^{[2]}, \\
& b_2^{[3]} = b_1^{[3]} + r_1^{[3]} m^{[2]} - p^{[2]}, \\
& b_2^{[3]} + m^{[2]}, \\
& b_3^{[3]} = b_2^{[3]} + r_2^{[3]} m^{[2]} - p^{[2]} \}
\end{aligned}$$

We have $m^{[2]} = 3$, $p^{[2]} = 7$ and $r^{[2]}_{j=0\to 2} = \{3, 2, 2\}$ therefore

$$\begin{aligned} b^{[2]}_{k=0\to 7} = \{\ 0 &= 0, \\ 3 &= 0+3, \\ 6 &= 0+2\times 3, \\ 2 &= 0+3\times 3-7, \\ 5 &= 2+3, \\ 1 &= 2+2\times 3-7, \\ 4 &= 1+3, \\ 0 &= 1+2\times 3-7\ \} \end{aligned}$$

Similarly for order i and type $t^{[i]} = 0$ we have $p^{[i]} = n^{[i-1]} < m^{[i]}$. Therefore as $n^{[i-1]} < \frac{p^{[i-1]}}{2}$ the numerator sequence of order $i-1$ will form subsequences of decreasing numerator values each differing by $n^{[i-1]}$. The values at the beginning of these sequences correspond to the numerator values in the numerator sequence of order i. Therefore the order i numerator sequence for type $t^{[i]} = 0$ represented by

$$b^{[i-1]}_0, \ldots, b^{[i]}_j, b^{[i]}_j - n^{[i-1]} + p^{[i-1]}, b^{[i]}_j - 2n^{[i-1]} + p^{[i-1]}, \ldots, b^{[i]}_j - r^{[i]}_{j-1} n^{[i-1]} + p^{[i-1]} = b^{[i]}_{j+1}, \ldots, b^{[i-1]}_0$$

An example of this construction of the numerator sequence can be seen in the order 2 numerator sequence of Figure 3.1(a).

3.2.1 The Initial Partial Run Length

To calculate the initial partial run length of order i based on the initial numerator value of order $i-1$, $b^{[i-1]}_0$, we need to calculate the number of runs or the number of numerator values in the numerator sequence of order $i-1$ before the first value less than $p^{[i]}$. We have that $b^{[0]}_0 = b$ from the definition of the line $\ell : y = \frac{a}{d}x + \frac{b}{d}$.

Theorem 3.2.3 *For order $i > 0$, if $b^{[i-1]}_0 < p^{[i]}$, $r^{[i]}_0 = 0$.*

For order 1, *where* $b^{[0]}_0 = b$ *the initial partial run length is*

$$r^{[1]}_0 = \left\lceil \frac{p^{[0]} - b^{[0]}_0}{p^{[1]}} \right\rceil = \left\lfloor \frac{p^{[0]} - b^{[0]}_0}{p^{[1]}} \right\rfloor + 1 \tag{3.2}$$

For order $i > 1$ and type $t^{[i]} = 0$ the initial partial run length is

$$r_0^{[i]} = \left\lfloor \frac{b_0^{[i-1]}}{p^{[i]}} \right\rfloor \tag{3.3}$$

For order $i > 1$ and type $t^{[i]} = 1$ the initial partial run length is

$$r_0^{[i]} = \left\lceil \frac{p^{[i-1]} - b_0^{[i-1]}}{p^{[i]}} \right\rceil = \left\lfloor \frac{p^{[i-1]} - b_0^{[i-1]}}{p^{[i]}} \right\rfloor + 1 \tag{3.4}$$

Proof : By definition, if $b_0^{[i-1]} < p^{[i]}$, $r_0^{[i]} = 0$.

For order 1, we have from the definition of ℓ

$$1 + \frac{b_0^{[1]}}{d} = \frac{a}{d} r_0^{[1]} + \frac{b_0^{[0]}}{d}$$

Therefore

$$r_0^{[1]} = \frac{d + b_0^{[0]} - b_0^{[1]}}{a} = \frac{p^{[0]} + b_0^{[0]} - b_0^{[1]}}{p^{[1]}}$$

From Corollary 2.4.4 $b_0^{[1]} < p^{[1]}$

$$r_0^{[1]} = \left\lceil \frac{p^{[0]} + b_0^{[0]}}{p^{[1]}} \right\rceil = \left\lfloor \frac{p^{[0]} + b_0^{[0]}}{p^{[1]}} \right\rfloor + 1$$

For order $i > 1$ and type $t^{[i]} = 0$ we have $p^{[i]} = n^{[i-1]} < m^{[i-1]}$. Therefore as $n^{[i-1]} < \frac{p^{[i-1]}}{2}$ the numerator sequence of order $i-1$ has the construction

$$b_0^{[i-1]}, b_0^{[i-1]} - p^{[i]}, b_0^{[i-1]} - 2p^{[i]}, \ldots, b_0^{[i-1]} - r_0^{[i]} p^{[i]} = b_0^{[i]}, \ldots$$

Therefore $b_0^{[i]} = b_0^{[i-1]} - r_0^{[i]} p^{[i]}$ and

$$r_0^{[i]} = \frac{b_0^{[i-1]} - b_0^{[i]}}{p^{[i]}}$$

As $b_0^{[i]} < p^{[i]}$

$$r_0^{[i]} = \left\lfloor \frac{b_0^{[i-1]}}{p^{[i]}} \right\rfloor$$

For order $i > 1$ and type $t^{[i]} = 1$ we have $p^{[i]} = m^{[i-1]} < n^{[i-1]}$. Therefore as $m^{[i-1]} < \frac{p^{[i-1]}}{2}$ the numerator sequence of order $i-1$ has the construction

$$b_0^{[i-1]}, b_0^{[i-1]} + p^{[i]}, b_0^{[i-1]} + 2p^{[i]}, \ldots, b_0^{[i-1]} + r_0^{[i]} p^{[i]} - p^{[i-1]} = b_0^{[i]}, \ldots$$

Therefore $b_0^{[i]} = b_0^{[i-1]} + r_0^{[i]} p^{[i]} - p^{[i-1]}$ and

$$r_0^{[i]} = \frac{p^{[i-1]} - b_0^{[i-1]} + b_0^{[i]}}{p^{[i]}}$$

We know $b_0^{[i]} < p^{[i]}$ and therefore $\frac{p^{[i-1]} - b_0^{[i-1]}}{p^{[i]}}$ is not an integer hence

$$r_0^{[i]} = \left\lceil \frac{p^{[i-1]} - b_0^{[i-1]}}{p^{[i]}} \right\rceil = \left\lfloor \frac{p^{[i-1]}}{p^{[i]}} - \frac{b_0^{[i-1]}}{p^{[i]}} \right\rfloor + 1$$

□

3.2.2 The Initial Numerator Value

To calculate the initial numerator value of order i, $b_0^{[i]}$, based on the initial numerator value of order $i-1$, $b_0^{[i-1]}$, and the initial partial run length, $r_0^{[i]}$, we provide the following theorem.

Theorem 3.2.4 *For the line $\ell : y = \frac{a}{d}x + \frac{b}{d}$, where $r_0^{[i]}$ is the length of the first partial run of order i and $b_0^{[0]} = b$, the first value of the numerator sequence of order i can be calculated by the following relationships. For order 1,*

$$b_0^{[1]} \equiv b_0^{[0]} + p^{[1]} r_0^{[1]} \pmod{p^{[0]}} \tag{3.5}$$

For order $i > 1$,

$$b_0^{[i]} \equiv b_0^{[i-1]} + m^{[i-1]} r_0^{[i]} \pmod{p^{[i-1]}} \tag{3.6}$$

$$b_0^{[i]} \equiv b_0^{[i-1]} - n^{[i-1]} r_0^{[i]} \pmod{p^{[i-1]}} \tag{3.7}$$

Proof : We have from Corollary 2.4.3 the equivalent relations to calculate the numerator sequence of order i.

$$\begin{aligned} b_j^{[i-1]} &\equiv b_0^{[i-1]} + jm^{[i-1]} \pmod{p^{[i-1]}} \\ b_j^{[i-1]} &\equiv b_0^{[i-1]} - jn^{[i-1]} \pmod{p^{[i-1]}} \end{aligned}$$

From Definition 3.2.1, the initial partial run length of order i is defined as the number of full runs of order $i-1$ before the first full run of order i. Hence set $j = r_0^{[i]}$ and therefore $b_j^{[i-1]} = b_0^{[i]}$. Therefore we have

$$\begin{aligned} b_0^{[i]} &\equiv b_0^{[i-1]} + r_0^{[i]} m^{[i-1]} \pmod{p^{[i-1]}} \\ b_0^{[i]} &\equiv b_0^{[i-1]} - r_0^{[i]} n^{[i-1]} \pmod{p^{[i-1]}} \end{aligned}$$

□

For the example of the line $\ell : y = \frac{17}{41}x + \frac{23}{41}$ described in Figure 3.1, we have

$$\begin{aligned} b_0^{[0]} &= b = 23 \\ r_0^{[1]} &= 2 \\ b_0^{[1]} &= 23 + 17 \times 2 \bmod 41 = 16 \\ r_0^{[2]} &= 2 \\ b_0^{[2]} &= 16 + 10 \times 2 \bmod 17 = 2 \\ r_0^{[3]} &= 0 \\ b_0^{[3]} &= 2 + 7 \times 0 \bmod 7 = 2 \end{aligned}$$

We therefore have a numeric technique for calculating the initial numerator value of any order. From these we can derive an equivalent algebraic technique.

Theorem 3.2.5 *Where $b_0^{[0]} = b$, if $b_0^{[i-1]} < p^{[i]}$, $b_0^{[i]} = b_0^{[i-1]}$.*

If $\frac{a}{d} \geq \frac{1}{2}$,

$$b_0^{[1]} = b_0^{[0]} - p^{[1]} \tag{3.8}$$

If $\frac{a}{d} < \frac{1}{2}$

$$b_0^{[1]} = b_0^{[0]} + r_0^{[1]} p^{[1]} - p^{[0]} \tag{3.9}$$

If order $i > 1$ and type $t^{[i]} = 0$

$$b_0^{[i]} = b_0^{[i-1]} - r_0^{[i]} p^{[i]} \tag{3.10}$$

If order $i > 1$ and type $t^{[i]} = 1$

$$b_0^{[i]} = b_0^{[i-1]} + r_0^{[i]} p^{[i]} - p^{[i-1]} \tag{3.11}$$

Proof : For order 1, if $\frac{a}{d} \geq \frac{1}{2}$ the run length $r^{[1]} = 1$. Therefore $r_0^{[1]} = 0$ or 1. For a value of $r_0^{[1]} = 1$, $b_0^{[0]} \geq p^{[1]}$.

For order 1, $p^{[1]} = a$. If $a > d - a$

$$b_0^{[0]}, b_0^{[0]} - a, b_0^{[0]} - r_0^{[1]} a = b_0^{[1]}, \ldots$$

If $a < d - a$

$$b_0^{[0]}, b_0^{[0]} + a, b_0^{[0]} + 2a, \ldots, b_0^{[0]} + r_0^{[i]} a - d = b_0^{[1]}, \ldots$$

If $a = d - a$ the two cases are equivalent.

For order $i > 1$ and type $t^{[i]} = 0$ we have $p^{[i]} = n^{[i-1]} < m^{[i]}$. Therefore as $n^{[i-1]} < \frac{p^{[i-1]}}{2}$ the numerator sequence of order $i - 1$ has the construction

$$b_0^{[i-1]}, b_0^{[i-1]} - p^{[i]}, b_0^{[i-1]} - 2p^{[i]}, \ldots, b_0^{[i-1]} - r_0^{[i]} p^{[i]} = b_0^{[i]}, \ldots$$

For order $i > 1$ and type $t^{[i]} = 1$ we have $p^{[i]} = m^{[i-1]} < n^{[i]}$. Therefore as $m^{[i-1]} < \frac{p^{[i-1]}}{2}$ the numerator sequence of order $i - 1$ has the construction

$$b_0^{[i-1]}, b_0^{[i-1]} + p^{[i]}, b_0^{[i-1]} + 2p^{[i]}, \ldots, b_0^{[i-1]} + r_0^{[i]} p^{[i]} - p^{[i-1]} = b_0^{[i]}, \ldots$$

□

Take the example of the line $\ell : y = \frac{17}{41}x + \frac{23}{41}$ described in Figure 3.1. For order 1, we have $b_0^{[0]} = b = 23$ and $r_0^{[1]} = 2$, therefore $b_0^{[1]} = 23 + 17 \times 2 - 41 = 16$. For order 2, we have $b_0^{[1]} = 16$, $r_0^{[2]} = 2$ and $t^{[2]} = 0$, therefore $b_0^{[2]} = 16 - 7 \times 2 = 2$. For order 3, we have $b_0^{[2]} = 2$, $r_0^{[3]} = 0$ and $t^{[3]} = 1$, therefore $b_0^{[3]} = 2 + 7 \times 0 - 3 = 2$

3.2.3 The Final Partial Run Length

The final partial run length of order i, denoted by $r_p^{[i]}$ as it is the $p^{[i]}$-th run in the intersection pattern, is the number of full runs of order $i-1$ after the last full run of order i in the intersection pattern $\mathcal{P}_\ell$.

Theorem 3.2.6 *For any order i, if $b_0^{[i]} = b_0^{[0]}$, $r_p^{[i]} = 0$. If $b_0^{[i]} \neq b_0^{[0]}$ we have the following cases.*

- *For order 1,*
  - *if* $b_0^{[1]} < m^{[1]}$, $r_p^{[1]} = r^{[1]} - r_0^{[1]} - 1$
  - *if* $b_0^{[1]} \geq m^{[1]}$, $r_p^{[1]} = r^{[1]} - r_0^{[1]}$
- *For order $i \geq 2$ and type 0*
  - *if* $b_0^{[i]} < m^{[i]}$, $r_p^{[i]} = r^{[i]} - r_0^{[i]}$
  - *if* $b_0^{[i]} \geq m^{[i]}$, $r_p^{[i]} = r^{[i]} - r_0^{[i]} - 1$
- *For order $i \geq 2$ and type 1*
  - *if* $b_0^{[i]} < m^{[i]}$, $r_p^{[i]} = r^{[i]} - r_0^{[i]} - 1$
  - *if* $b_0^{[i]} \geq m^{[i]}$, $r_p^{[i]} = r^{[i]} - r_0^{[i]}$

Proof : It is evident that if $b_0^{[i]} = b_0^{[0]}$ there does not exist any run of order $j \leq i$ that is truncated.

If a run of order i is truncated to produce the initial and final partial runs, the length of the truncated run, $r'^{[i]}$, can be determined by Theorem 2.4.8. Given that the length of the initial partial run of order i is $r_0^{[i]}$ and one run of order $i-1$ is also truncated, we have

$$r_p^{[i]} = r'^{[i]} - r_0^{[i]} - 1 \tag{3.12}$$

□

3.3 Lines with Irrational Slope

Having completed the treatment of a line with rational slope and real intercept, we will now consider the case of an irrational slope.

We will be working towards a technique by which we can approximate the intersection pattern $\mathcal{P}_\ell$ by a line with rational slope and intercept within a bounded frame of the lattice, $\mathcal{L}_{(f_x,f_y)}$. Initially we will consider a line with irrational slope and zero intercept. We will then describe the effect of adding a non-zero intercept.

3.3.1 Zero Intercept

We begin with a basic fact about the intersection pattern. Lines given by $\ell : y = \alpha x + \beta$ and $\ell' : y = \alpha' x + \beta'$ respectively will have distinct intersection patterns if and only if there is some integer h such that

$$||\alpha h + \beta| - \lfloor \alpha' h + \beta' \rfloor| \geq 1 \tag{3.13}$$

In particular, if n is an integer such that $|\alpha - \alpha'| > 1/n$, and k is an integer such that $k > |\beta - \beta'| + 1$, then it is clear that equation 3.13 will hold when $h = kn$.

The first step we will take is to show that a line of irrational slope, α, and zero intercept can be approximated by the line of rational slope, $\frac{a}{d}$, within the frame $\mathcal{L}_{(d,a)}$ where $f_x < d$ and $\alpha - \frac{a}{d} < \frac{1}{d^2}$. This final condition is chosen because there are infinitely many rational numbers $\frac{a}{d}$ such that $0 < \alpha - \frac{a}{d} < 1/d^2$ where $\frac{a}{d}$ is the n-th partial convergent of the continued fraction expansion of α. We know that the n-th partial convergent provides the best rational approximation of the irrational slope. Therefore due to the connection between continued fractions and the run structure of the intersection pattern defined by a line of rational slope the run structure of the intersection pattern $\mathcal{P}_\ell$ within the frame $\mathcal{L}_{(f_x,f_y)}$ is defined by this result.

Theorem 3.3.1 *The intersection patterns $\mathcal{P}_\ell$ and $\mathcal{P}_{\ell'}$ of the lines $\ell : y = \alpha x$ and $\ell' : y = \frac{a}{d}x$, are identical in the frame $\mathcal{L}_{(d,a)}$ if $0 < \alpha - a/d < 1/d^2$, where $a, d \in \mathcal{N}$ and α is irrational. If a and d are coprime, then the condition $|\alpha - \frac{a}{d}| < \frac{1}{d^2}$ suffices.*

Proof : Let x_h be a positive integer. Note that $\alpha x_h > \lfloor \alpha x_h \rfloor$ since α is irrational. If ax_h/d is not an integer then it is a distance of at least $1/d$ from the nearest integer. These two facts and equation 3.13 imply that if P_ℓ and $P_{\ell'}$ have different cells involving x coordinate x_h then $\alpha x_h - ax_h/d > 1/d$ but this is impossible if $x_h \le d$ because

$$0 < \alpha x_h - ax_h/d < x_h/d^2 < 1/d$$

If a and d are coprime and $0 \le x_h < d$ then ax_h/d is never an integer. The case to consider is $0 < a/d - \alpha < 1/d^2$. Now ax_h/d is at least $1/d$ above an integer and we can repeat the above argument. $\square$

It should be noted about Theorem 3.3.1 that if $0 < a/d - \alpha < 1/d^2$ but a and d are not coprime, it can happen that $P_\ell \ne P_{\ell'}$ in $\mathcal{L}_{(d,a)}$. Inequality will occur when ah/d is an integer. In this case, $\lfloor \alpha h \rfloor = ah/d - 1$.

As stated, for any irrational α there are infinitely many rational numbers a/d such that $0 < \alpha - a/d < 1/d^2$ and a and d are coprime which follows from the continued fraction expansion of an irrational. Therefore we can approximate the line ℓ with the line $\ell' : y = \frac{P_n}{Q_n}x$ where $\frac{P_n}{Q_n}$ is the n-th partial convergent of the continued fraction expansion of α. We reiterate that, due to the connection between continued fractions and the hierarchy of the run structure of a line with rational slope, the following result defines the structure of hierarchy of runs within the intersection pattern of a line of irrational slope within the frame $\mathcal{L}_{(f_x,f_y)}$.

Theorem 3.3.2 *The intersection patterns $\mathcal{P}_\ell$ and $\mathcal{P}_{\ell'}$ of the lines $\ell : y = \alpha x$ and $\ell' : y = \frac{P_n}{Q_n}x$ are identical in the frame $\mathcal{L}_{(f_x,f_y)}$ where α is irrational, P_n/Q_n is the n-th partial convergent of the regular continued fraction expansion of α and $f_x \le Q_{n-1}$.*

Proof : We have from the literature [71], $\left|\alpha - \frac{P_n}{Q_n}\right| < \frac{1}{Q_nQ_{n-1}} < \frac{1}{f_x^2}$. By Theorem 3.3.1 we have the proof. $\square$

In Chapter 4 we shall refine this result by showing that the termination condition of $f_x \le Q_{n-1}$ can be reduced to $f_x \le Q_n$.

3.3.2 Non–Zero Intercept

Our next step is to describe the effect of adding a non-zero intercept to the line with irrational slope. We wish to approximate the line $\ell : y = \alpha x + \beta$ with the line $\ell' : y = \frac{a}{d}x + \frac{b}{d}$ within the frame $\mathcal{L}_{(f_x, f_y)}$ such that $0 < \alpha - \frac{a}{d} < \frac{1}{d^2}$ and $0 < \beta - \frac{b}{d} < \frac{1}{d}$. Therefore we can then use the knowledge we have about the hierarchy of the run structure of the intersection pattern $\mathcal{P}_{\ell'}$ to describe the run structure of $\mathcal{P}_\ell$. The problem is that no matter how close the rational approximation is to the irrational slope the intersection patterns may differ. The problem is that we cannot guarantee that the distance between the approximate line ℓ' and the true line ℓ is less than $\frac{1}{d}$.

If the distance between the original intercept and the approximate intercept is $\frac{1}{d} - \frac{f_x}{d^2} < \beta - \frac{b}{d} < \frac{1}{d}$, at the point (x_h, y_h) where $x_h < f_x$ and $\frac{a}{d}x_h + \frac{b}{d} = y_h - \frac{1}{d}$ the intersection patterns can differ. This situation, described in Figure 3.2, occurs because the difference in the original line and the approximate line has the bounds, $\frac{1}{d} < \left(\alpha - \frac{a}{d}\right) x_h + \beta - \frac{b}{d} < \frac{1}{d^2}x_h + \frac{1}{d}$.

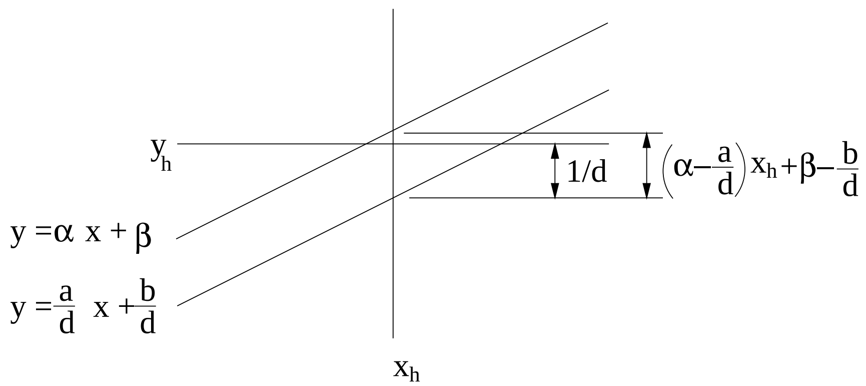


Figure 3.2: The geometry of a near corner intersection of a line with irrational slope and intercept.

However all is not lost. This situation can only occur at lattice position (x_h, y_h) where $y_h - \frac{a}{d}x_h = \frac{1}{d}$. That is at the beginning of the pixel associated with the order 0 numerator sequence value of $d - 1$ of the line ℓ'.

What we will show is that, if the lattice position (x_h, y_h) where $\frac{a}{d}x_h - (y_h - 1) = d - 1$ is not in the frame or $\left(\alpha - \frac{a}{d}\right) x_h + \beta - \frac{b}{d} < \frac{1}{d}$ at this point, the approximate line $\ell' y = \frac{a}{d}x + \frac{b}{d}$ has the same intersection pattern as the line ℓ. If the lattice position (x_h, y_h) is in the frame $\mathcal{L}_{(f_x,f_y)}$ and at this point we have the condition $\frac{1}{d} < \left(\alpha - \frac{a}{d}\right) x_h + \beta - \frac{b}{d} < \frac{1}{d^2}x_h + \frac{1}{d}$, the intersection pattern $\mathcal{P}_\ell$ of the line ℓ is identical to the intersection pattern produced by the line $\ell'' : y = \frac{a}{d}x + \frac{b+1}{d}$.

We will start our theoretical arguments by showing some intermediary results. Our aim is to separate the cases where a difference can occur between the intersection patterns $\mathcal{P}_\ell$ and $\mathcal{P}_{\ell'}$. A difference occurs when the value of the intercept β is greater than $\frac{1}{d} - \frac{f_x}{d^2}$. We shall then initially consider the case where $0 < \beta - \frac{b}{d} < \frac{1}{d} - \frac{f_x}{d^2}$. If $0 < \alpha - \frac{a}{d} < \frac{1}{d^2}$ and $f_x \leq d$ then the line $\ell' : y = \frac{a}{d}x + \frac{b}{d}$ is a good approximation for $\ell : y = \alpha x + \beta$.

We will then show the intermediary result that the lines ℓ' and ℓ'' have identical intersection patterns up to where the ℓ'' intersects its first lattice position. That is the position (x_h, y_h), where $\left\lfloor \frac{a}{d}x_h + \frac{b+1}{d} \right\rfloor = 0$. We will complete the treatment by considering the case where $\frac{1}{d} - \frac{f_x}{d^2} < \beta - \frac{b}{d} < \frac{1}{d}$ and showing which line $\ell' : y = \frac{a}{d}x + \frac{b}{d}$ or $\ell'' : y = \frac{a}{d}x + \frac{b+1}{d}$ is the better approximation to the line $\ell : y = \alpha x + \beta$.

Consider the case where $0 < \beta - \frac{b}{d} < \frac{1}{d} - \frac{f_x}{d^2}$.

Lemma 3.3.3 *The intersection patterns $\mathcal{P}_\ell$ and $\mathcal{P}_{\ell'}$ of the lines $\ell : y = \alpha x + \beta$ and $\ell' : y = \frac{a}{d}x + \frac{b}{d}$, are identical in the frame $\mathcal{L}_{(f_x,f_y)}$ where α is irrational, $0 < \alpha - \frac{a}{d} < \frac{1}{d^2}$, $0 < \beta - \frac{b}{d} < \frac{1}{d} - \frac{f_x}{d^2}$ and $f_x \leq d$.*

Proof : Let x_h be a positive integer. If $\frac{a}{d}x_h$ is not an integer, then it is a distance of at least $\frac{1}{d}$ from the nearest integer. This fact and equation 3.13 imply that if P_ℓ and $P_{\ell'}$ have different cells involving x coordinate x_h, then $(\alpha x_h + \beta) - (\frac{a}{d}x_h - \frac{b}{d}) > 1/d$, but this is impossible if $x_h \leq f_x < d$ because

$$\left(\alpha - \frac{a}{d}\right) x_h + \beta - \frac{b}{d} < \frac{1}{d} - \frac{f_x - x_h}{d^2} < \frac{1}{d}$$

□

Now for the second intermediate result that the lines ℓ' and ℓ'' have identical intersection patterns up to the point where the ℓ'' intersects its first lattice position.

Lemma 3.3.4 *The 4– or 8–connected intersection pattern $\mathcal{P}'_\ell$ of the line $\ell' : y = \frac{a}{d}x + \frac{b}{d}$, and the intersection pattern $\mathcal{P}'_\ell$ of the line $\ell'' : y = \frac{a}{d}x + \frac{b+1}{d}$ are identical up to but not including the first lattice point intersected by ℓ''.*

Proof : Let (x_h, y_h) be the first lattice point intersected by ℓ''. The slopes of ℓ' and ℓ'' are identical and hence the two lines produce identical run parameters of all orders. For $x = 0, \ldots, x_h - 1$ we have $b_j''^{[i]} = b_j'^{[i]} + 1$. Therefore by Theorem 2.4.7 the intersection pattern will be identical. $\square$

We can now approach the case where $\frac{1}{d} - \frac{f_x}{d^2} \leq \beta - \frac{b}{d} < \frac{1}{d}$ to complete our argument. We show under which conditions will the line $\ell' : y = \frac{a}{d}x + \frac{b}{d}$ or the line $\ell' : y = \frac{a}{d}x + \frac{b+1}{d}$ will be a better approximate for the line $\ell : y = \alpha x + \beta$.

Theorem 3.3.5 *Given that we have an approximation of the line $\ell : y = \alpha x + \beta$ by the line $\ell' : y = \frac{a}{d}x + \frac{b}{d}$ within the frame $\mathcal{L}_{(f_x, f_y)}$, where $0 < \alpha - \frac{a}{d} < \frac{1}{d^2}$, $\frac{1}{d} - \frac{f_x}{d^2} \leq \beta - \frac{b}{d} < \frac{1}{d}$ and $f_x \leq d$.*

The intersection pattern $\mathcal{P}_\ell$ of the line ℓ is identical to the intersection pattern $\mathcal{P}_{\ell''}$ of the line $\ell'' : y = \frac{a}{d}x + \frac{b+1}{d}$ if there exists a point (x_h, y_h) within the frame $\mathcal{L}_{(f_x, f_y)}$ where $\frac{a}{d}x_h + \frac{b+1}{d} = y_h$ and $\lfloor \alpha x_h + \beta \rfloor = y_h$.

If this point does not exist within the frame or at this point we have the condition that $\lfloor \alpha x_h + \beta \rfloor = y_h - 1$, the intersection pattern $\mathcal{P}_{\ell'}$ of the line ℓ' is identical $\mathcal{P}_\ell$.

Proof : Let x_j be a positive integer where $x_j < x_h$. If $y = \frac{a}{d}x_j + \frac{b}{d}$ is not an integer then it is a distance of at least $\frac{2}{d}$ from the nearest integer.

This fact and equation 3.13 imply that if $\mathcal{P}_\ell$ and $\mathcal{P}_{\ell'}$ have different cells involving x_j, then $\alpha x_j + \beta - \frac{a}{d}x_j - \frac{b}{d} > 2/d$. This is impossible if $x_j \leq f_x < d$ because

$$0 < \left(\alpha - \frac{a}{d}\right) x_j + \beta - \frac{b}{d} < \frac{1}{d^2}x_h + \frac{1}{d} < \frac{2}{d} \tag{3.14}$$

If y is an integer, it is straightforward to show that $\lfloor \alpha x_j + \beta \rfloor - \left\lfloor \frac{a}{d}x_j - \frac{b}{d} \right\rfloor = 0$ from equation 3.14.

From Lemma 3.3.4 we have that the intersection patterns $\mathcal{P}_\ell$, $\mathcal{P}_{\ell'}$ and $\mathcal{P}_{\ell''}$ are identical up to but not including the point (x_h, y_h) in the frame $\mathcal{L}_{(f_x,f_y)}$. Therefore if the point is not in the frame, $x_h \geq f_x$, either line ℓ' or ℓ'' can be used to approximate the intersection pattern $\mathcal{P}_\ell$.

If the point (x_h, y_h) is within the frame and we have the condition $\lfloor \alpha x_h + \beta \rfloor = y_h$ it is apparent that equation 3.13 will only hold where $\frac{a}{d}x_h + \frac{b+1}{d} = y_h$. In this case for values of x_j where $x_h < x_j < f_x$ the line ℓ is equivalent to the line $y = y_h + \alpha x + \beta'$ where $\beta' = \alpha x_h + \beta - y_h < \frac{1}{d}$. Therefore by Lemma 3.3.4, the rest of the intersection pattern $\mathcal{P}_\ell$ is identical to $\mathcal{P}_{\ell''}$.

If the point (x_h, y_h) is within the frame and we have the condition $\lfloor \alpha x_h + \beta \rfloor = y_h - 1$, it is apparent that equation 3.13 will only hold where $\frac{a}{d}x_h + \frac{b}{d} = y_h$. In this case for values of x_j, where $x_h < x_j < f_x$, by Corollary 2.4.5 the situation that $\frac{a}{d}x_h + \frac{b}{d} = y_h - \frac{1}{d}$ cannot occur again within the frame. Therefore we use the same argument as for the case of $x_j < x_h$. $\square$

By these results we have described the structure of the hierarchy of runs within the intersection pattern of the line with irrational slope and real intercept. What we have not dealt with so far is the occurrence of corner intersections.

3.3.3 Corner Intersections

If the slope is irrational, there can only be at most one corner intersection. If there is more than one corner intersection, say at lattice points $P_0(x_0, y_0)$ and $P_1(x_1, y_1)$, the slope of the line is $\alpha = \left|\frac{y_1 - y_0}{x_1 - x_0}\right|$, a rational number.

If the slope is irrational, and the intercept is rational, there can be no corner intersections. For a rational intercept, a point of the line at $x = 0$ is $P_0(0, \frac{b}{d})$. If there was a corner intersection at point $P_1(x_1, y_1)$ the slope of the line is $\alpha = \frac{dy_1 - b}{x_1}$, again a rational number. Therefore we can concern ourselves solely with the situation where the intercept is irrational.

If the slope is irrational, there can be only one corner intersection at most. Therefore the question is, can the one possible corner intersection occur within the bounds of the frame $\mathcal{L}_{(f_x,f_y)}$? From our discussions in the last subsection, the corner intersection can only occur at the position (x_h, y_h) described in Theorem 3.3.5. If corner intersections need to be detected, at position $x_h < f_x$ the corner of the cell will be intersected if

$$\alpha x_h + \beta - y_h = 0 \tag{3.15}$$

3.4 Conclusions

In this chapter we have extended our description of the hierarchy of runs in the intersection pattern of a line of rational slope to a line of real slope and intercept. We have provided the necessary conditions under which the run structure of the line with real slope and intercept can be approximated by a line of rational slope by using a continued fractions approximation. We have also provided techniques to calculate the initial and final partial run lengths and the initial numerator sequences value such that the intersection pattern can be generated by the technique described in the previous chapter. The application we will apply these results to is ray tracing a uniform spatial subdivision. Therefore we have included a treatment of corner intersections.

3.4.1 The Sensitivity of the Digital Line to Changes in Slope and Intercept

The question of how much a slope or intercept can change is a natural progression from the work preented here. Dorst and Duin [45] introduced a geometric representation, the spirograph theory, which they used to describe several properties of the line.

The spirograph is a circle of unit circumference onto which a number of points are inscribed. If the intersection pattern of the line $\ell : y = \alpha x + \beta$ where $0 < \alpha < 1$ is being investigated within a bounded frame of the lattice $\mathcal{L}_{(d,a)}$, d points are inscribed each separated by an arc length of α. While Dorst *et al.* consider a line with real slope and intercept we will give an example of their work using line with rational slope and real intercept. From Figure 3.3, consider the intersection pattern defined by the line $\ell : y = \frac{3}{10}x + \beta$ in the frame $\mathcal{L}_{(d,a)}$. On the spirograph we have 10 points, each separated from the previous by an arc length of $\frac{3}{10}$.

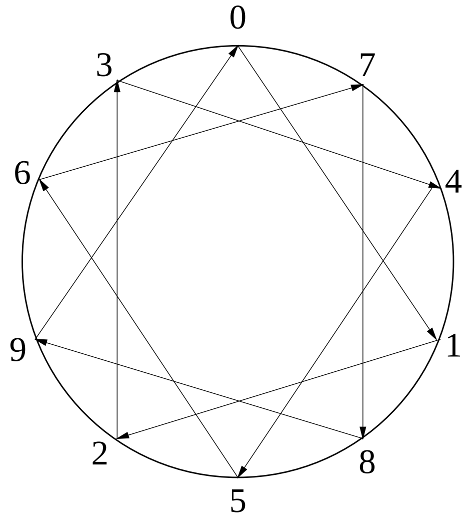


Figure 3.3: The spirograph corresponding to the line $\ell : y = \frac{3}{10}x + \beta$.

The distance $D(i, j)$ between two points i and j on the spirograph is the arc length between i and j defined in an anticlockwise direction where $0 \leq (i, j) < d$.

$$D(i, j) = (i - j)\alpha - \lfloor (i - j)\alpha \rfloor \tag{3.16}$$

Several properties of this distance parameter were provided,

- $0 \leq D(i, j) < 1$
- $D(i, j) + D(j, i) = 0$ if $(i - j)\alpha$ is an integer and 0 if not.
- $D(i, j) \equiv D(i, k) + D(k, j) \pmod 1$
- Every distance can be measured relative to position 0. If $i \geq j$, $D(i, j) = D(i - j, 0)$ and if $i \leq j$, $D(i, j) = D(0, j - i)$.

Two distances, and the points by which they were defined, were most important to Dorst. The point R defined by the smallest distance r to the right or clockwise of point 0, and the point L defined by the smallest distance l to the left or anti-clockwise. Where $k = 1, \ldots, d-1$,

$$r = min\{D(k,0)\} = \alpha R - \lfloor \alpha R \rfloor \quad (3.17)$$

$$l = min\{D(0,k)\} = \lceil \alpha L \rceil - \alpha L \quad (3.18)$$

For example, from Figure 3.3, we have $L = 3$, $l = \frac{3}{10}$, $R = 7$ and $r = \frac{3}{10}$.

Dorst defines that the strictest bounds on the slope of the line to maintain the same topology or order of the labels of the Spirograph are

$$\frac{\lfloor \alpha R \rfloor}{R} \leq \alpha < \frac{\lceil \alpha L \rceil}{L} \quad (3.19)$$

From our example we have $\frac{3}{7} \leq \alpha < \frac{1}{3}$ which are illustrated in Figure 3.4.

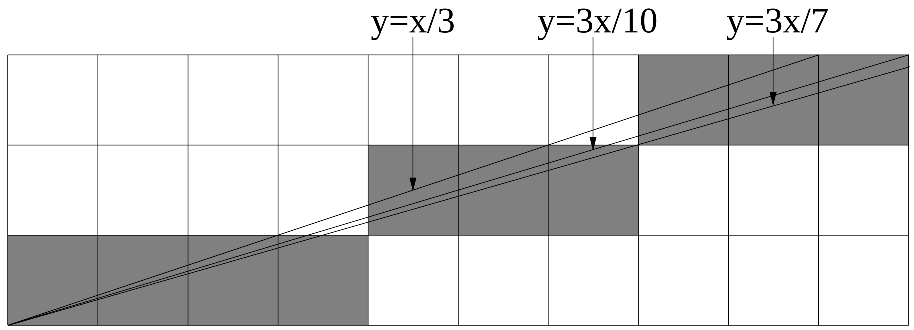


Figure 3.4: The bounds to changes of the slope of the line $\ell : y = \frac{3}{10}x$.

Dorst *et al.* also describes the effect of the introduction of an intercept has on the intersection pattern. The result which interests us is the maximum change of an intercept which does not alter the intersection pattern. This was shown to be $\frac{1}{d}$.

As well as the question of how much can one change the slope and intercept of a line before the chain code sequence or intersection pattern of the digital line changes, the question of which set of lines could produce a given chain code or intersection pattern is important. Dorst [43] answered the question, *given an intersection pattern, what is the set of lines $y = \alpha x + \beta$ that could have generated it?* Which is the reverse of the process of generating the intersection pattern of a continuous line.

He showed that the chain code of a line $\ell : y = \alpha x + \beta$ can be represented within

$n+1$ columns by the line $\ell' : y = \frac{a}{d}x + b$ where

$$c_i = \left\lfloor \frac{a}{d}(i-s) \right\rfloor - \left\lfloor \frac{a}{d}(i-s-1) \right\rfloor \text{ for } i = 1, \ldots, n \tag{3.20}$$

and c_i is the i-th chain code element in the string.

$$c_i = \lfloor y(i) \rfloor - \lfloor y(i-1) \rfloor \text{ for } i = 1, \ldots, n \tag{3.21}$$

$$y_i = \lfloor \alpha x + \beta \rfloor \tag{3.22}$$

The integer s is chosen to be

$$s = \lfloor \beta d \rfloor l - \left\lfloor \frac{\lfloor \beta d \rfloor l}{d} \right\rfloor d \tag{3.23}$$

where l is chosen such that $la \equiv d-1 \pmod{d}$ which he shows exists and is unique.

Chapter 4

The Greatest Common Divisor, Continued Fractions and the Run Parameters

One of the earliest, if not the earliest algorithm described, was *Euclid's algorithm* (*c.* 300 B.C.) [51, 82] which calculates the greatest common divisor (gcd) of two integers. The greatest common divisor of two numbers is the largest number which evenly divides both numbers. It is interesting to note that Euclid's algorithm which is still used widely today, was penned in a time when unity was not considered a number and the concept of zero did not exist. While some suggest Euclid's algorithm is a rendition of an earlier method by Eudoxus (*c.* 375 B.C.) [98] and systems for calculating the gcd can be dated back to the Sumerians (*c.* 2000 B.C.), it is suggested that these techniques did not fulfill the requirements of an algorithm.

Euclid's gcd algorithm has a very close connection to the continued fractions algorithm, which provides an infinite, integer–based, exact representation of real numbers. The order of approximation of the continued fraction convergents improves exponentially.

In this chapter, we provide a full and complete description of the connection between the hierarchical structure of runs in the intersection pattern, $\mathcal{P}_\ell$, the least remainder version of Euclid's algorithm and continued fractions.

The connection between these algorithms provide useful results for the analysis of the hierarchy of runs with the intersection pattern. Knuth [82] has provided a rigorous study of the behaviour of the gcd algorithm and these results can be extended to characterise the behaviour of the algorithm to calculate the structure of the hierarchy of runs within the intersection pattern. The connection also provides some interesting insights into the characterisation of the run structure of the intersection pattern.

One important use of continued fractions is to provide an integer based approximation of irrational numbers with infinite precision. The connection between the run structure of the intersection pattern and continued fractions allows for the development of a technique for approximating a line with irrational slope by a line of rational slope such that the intersection patterns produced are identical. If a suitable approximation can be found, which is always the case, a run–based description of the lattice cells intersected by a ray can be developed. This topic was discussed in Chapter 3.

4.1 The Link Between Number Theory and the Structure of Runs

Due to the discrete nature of the lattice, number theory has played an important role in characterising the digital line. We will concentrate on the connection between the chain code of or sequences of runs within the line and continued fractions [71, 78]. A number of algorithms for generating the chain code of a line have had a resemblance to the continued fractions algorithm. Brons [21, 22] developed an algorithm to generate the chain code of a line with rational slope. The basis of the technique was the continued fractions algorithm. Arcelli and Massarotti [5] modified the Brons' algorithm for line drawing. Cederberg [30] also produced a similar algorithm and used a shifting or permutation style method for altering the chain code to use for line drawing. Wu [141] produced a version of the algorithm similar to Brons for irrational slope, as did Dorst and Duin [45] whose algorithm is identical to the continued fractions algorithm. However the link between the run lengths within the intersection pattern and regular continued fractions has not be fully described.

Castle and Pitteway [103] noted a connection between continued fractions and the run lengths found in the intersection pattern of a line and a lattice. Although they did not supply any theoretical arguments and their description is incomplete, they illustrated the connection through the example of the line $y = \frac{16}{131}x + \frac{1}{2}$.

The emphasis of Castle's and Pitteway's work was geared towards line drawing hence the intercept of $\frac{1}{2}$. Ignoring the intercept which does not alter the argument, the intersection pattern for the line $\ell : y = \frac{16}{131}x$ using Freeman's chain code description is

$$(0^8 1\ (0^7 1)^4)^2\ 0^8 1\ (0^7 1)^5$$

Using the notation s for a short run and l for long, we can more easily describe the run structure of the pattern. We should note here that codes 0 and 1 can be used for short and long runs respectively and there appears to be some similar properties to Freeman's chain codes at higher orders. In this discussion s and l have a more apparent meaning.

To describe the first level or order of the run structure we need to identify the 0^+1 subgroups of the chain code. Each of which represent a run of order 1: a run of pixels. Within the chain code of ℓ we have two subgroups which occur, $0^7 1$ and $0^8 1$. The subgroup $0^7 1$ represents a short run of order 1 and has a length of $r^{[1]} = 8$. A long run of length $r^{[1]} + 1 = 9$ is represented by the subgroup $0^8 1$. Therefore the intersection pattern at order 1 can be described by

$$ls^4\ ls^4\ ls^5 \quad \text{or} \quad 10^4\ 10^4\ 10^5$$

and the slope of the line, $\alpha = \frac{16}{131}$, can be bounded by

$$\frac{1}{r^{[1]}} < \alpha < \frac{1}{r^{[1]} + 1} \tag{4.1}$$

which from our example is

$$\frac{1}{8} < \alpha < \frac{1}{9} \tag{4.2}$$

For order 2, the length of a short run is $r^{[2]} = 5$ and the length of a long run is $r^{[2]} + 1 = 6$, and the run structure of the intersection pattern at order 2 is

$$ssl \quad \text{or} \quad 001$$

The bounds of slope of the line can then be tightened by

$$\frac{1}{r^{[1]} + \frac{1}{r^{[2]}}} < \alpha < \frac{1}{r^{[1]} + \frac{1}{r^{[2]}+1}} \tag{4.3}$$

which for an example is

$$\frac{5}{141} = \frac{1}{8 + \frac{1}{5}} < \frac{16}{131} < \frac{1}{8 + \frac{1}{6}} = \frac{6}{49} \tag{4.4}$$

For order 3, the length of a short run is $r^{[3]} = 3$ and long runs are not possible: there is only one run of order 3 in the pattern. Pitteway notes that the lengths of the short runs of each order, 8, 5 and 3, correspond to the quotients of Euclid's algorithm. Due to the connection between Euclid's algorithm and continued fractions, the slope of the line can be represented by the regular continued fraction expansion where the partial denominators are the run lengths of each order.

$$\frac{16}{131} = \cfrac{1}{8 + \cfrac{1}{5 + \cfrac{1}{3}}} \tag{4.5}$$

However not all lines follow this example. Take the line $y = \frac{17}{131}x$. The chain code and run structures for all orders of the intersection pattern of this line are summarised by

Order 0	$(0^7 1)^3\ (0^6 1)\ (0^7 1)^2\ (0^6 1)\ (0^7 1)^3\ (0^6 1)\ (0^7 1)^2\ (0^6 1)\ (0^7 1)^2\ (0^6 1)$
Order 1	$l^3 s\ l^2 s\ l^3 s\ l^2 s\ l^2 s$
Order 2	$ls\ ls^2$
Order 3	sl

Therefore the run lengths are $r^{[1]} = 7$, $r^{[2]} = 3$, $r^{[3]} = 2$ and $r^{[4]} = 2$. These values however are not the partial denominators of the regular continued fraction expansion of $\frac{17}{131}$ which is

$$\frac{17}{131} = \cfrac{1}{7 + \cfrac{1}{1 + \cfrac{1}{2 + \cfrac{1}{2 + \cfrac{1}{2}}}}} \tag{4.6}$$

The run lengths coincide with the partial denominators of a general continued fraction expansion of $\frac{17}{131}$.

$$\frac{17}{131} = \cfrac{1}{7 + \cfrac{\frac{12}{5}}{3 + \cfrac{1}{2 + \cfrac{1}{2}}}} \tag{4.7}$$

Therefore it appears that the relationship between the run structure of the digital line and continued fractions is not as straightforward as it was hoped. This raises a question about the algorithms presented which describe the run hierarchy in the chain code of a digital line. The algorithm we will consider is that presented by Dorst and Duin [45] as they draw a similar conclusion between their algorithm based on the continued fractions algorithm and the calculation of run length.

The 4–connected chain code string, $\mathcal{P}_\ell$, of the line, $\ell : y = \alpha x$ where $0 < \alpha < 1$ is given by the following relations. The 8–connected intersection pattern can be calculated by setting $n_0 = \left\lfloor \frac{1}{\alpha} \right\rfloor - 1$.

$$\begin{array}{ll} \alpha_0 = \alpha & \alpha_{i+1} = \frac{1}{\alpha_i} - n_i \\ n_0 = \left\lfloor \frac{1}{\alpha_0} \right\rfloor & n_{i+1} = \left\lfloor \frac{1}{\alpha_{i+1}} \right\rfloor \\ \mathcal{U}_0 = 1 & \mathcal{U}_{i+1} = \mathcal{C}_i^{n_{2i}} \mathcal{U}_i \\ \mathcal{C}_0 = 0 & \mathcal{C}_{i+1} = \mathcal{C}_i \mathcal{U}_{i+1}^{n_{2i+1}} \end{array} \tag{4.8}$$

If the N elements of the chain code are required, the algorithm is terminated when the number of codes $q_i \geq N$ where $q_{-1} = q_0 = 1$ and $q_{i+1} = n_i q_i + q_{i-1}$. If α is irrational the algorithm generates an infinite string. If α is rational, the algorithm generates an endlessly repeating string. However for lines of rational slope the algorithm is incomplete. The calculation of α_i and n_i will terminate when $\alpha_{i+1} = 0$. In this case, $n_i = \left\lfloor \frac{1}{\alpha_i} \right\rfloor - 1$. We will also assume that the values of $n_{i+1} = n_0$, $n_{i+2} = n_1$ and so on.

For example, consider the 8–connected chain code of the line $\ell : y = \frac{16}{131}x$ within the bounded frame of the lattice $\mathcal{L}_{(131,16)}$.

i	α_i	n_i	$\mathcal{U}_i$	$\mathcal{C}_i$
0	$\frac{16}{131}$	7	1	0
1	$\frac{3}{16}$	5	$0^7 1$	$0(0^7 1)^5$
2	$\frac{1}{3}$	2	$(0(0^7 1)^5)^2(0^7 1)$	$(0(0^7 1)^5)((0(0^7 1)^5)^2(0^7 1)$

Table 4.1: Calculation of the chain code of the line $\ell : y = \frac{16}{131}x$.

Therefore $\mathcal{C}_2$ is the chain code of the line $\ell : y = \frac{16}{131}x$ within the bounded frame of the lattice $\mathcal{L}_{(131,16)}$. If the line is to be extended outside this frame, the chain code, $\mathcal{C}_2$, forms the basic pattern which is repeated throughout the line.

To provide an example where problems occur we will once again consider the line $\ell : y = \frac{17}{131}x$ within the bounded frame of the lattice $\mathcal{L}_{(131,17)}$. We have $n_i = 6, 1, 2, 1, \ldots$ and

$$\begin{array}{llllll}
\text{for } i = 0, & \mathcal{U}_0 & = & 1 & & \\
 & \mathcal{C}_0 & = & 0 & & \\
\text{for } i = 1, & \mathcal{U}_1 & = & \mathcal{C}_0^{n_0}\mathcal{U}_0 & = & 0^6 1 \\
 & \mathcal{C}_1 & = & \mathcal{C}_0\mathcal{U}_1^{n_1} & = & 0(0^6 1)^1 = 0^7 1 \\
\text{for } i = 2, & \mathcal{U}_2 & = & \mathcal{C}_1^{n_2}\mathcal{U}_1 & = & (0^7 1)^2(0^6 1) \\
 & \mathcal{C}_2 & = & \mathcal{C}_1\mathcal{U}_2^{n_3} & = & (0^7 1)((0^7 1)^2(0^6 1))^1 = (0^7 1)^3(0^6 1) \\
\text{for } i = 3, & \mathcal{U}_3 & = & \mathcal{C}_2^{n_4}\mathcal{U}_2 & = & ((0^7 1)^3(0^6 1))^{n_4}(0^7 1)^2(0^6 1) \\
 & \mathcal{C}_3 & = & \mathcal{C}_2\mathcal{U}_3^{n_5} & = & (0^7 1)^3(0^6 1)(((0^7 1)^3(0^6 1))^{n_4}(0^7 1)^2(0^6 1))^{n_5}
\end{array}$$

Therefore for whatever value of n_4 and n_5, the chain code will start with the sequence $(0^7 1)^3(0^6 1)(0^7 1)^{\mathbf{3}}(0^6 1)\ldots$ which differs from the actual chain code of the line $y = \frac{17}{131}x$, $(0^7 1)^3$ $(0^6 1)$ $(0^7 1)^{\mathbf{2}}$ $(0^6 1)$.

To describe the relationship between the chain code of a line and continued fractions we will present the arguments of McIlroy [93]. Freeman [55] stated that the chain code representing a digitised line consists of two code values and to assure minimum deviation from the true straight-line path, the two code values should alternate to the maximum extent. McIlroy states that continued fractions provides the details of the distribution of these codes.

To show the connection, we will be working through the example of $\alpha = \frac{17}{131}$. The regular continued fraction expansion of $\frac{17}{131}$ is given by equation 4.6 and has the partial denominators $x_i = \{7, 1, 2, 2\}$.

McIlroy uses three chain code transformations, the first two defined by Bruckstein [23], to manipulate the chain code

- X – exchange 0 and 1,
- S – replace 01 by 1, and
- R – reverse the chain code.

He assumes that the slope of the line $\alpha < \frac{1}{2}$ and therefore there are more 0's than 1's in the chain code. From Rosenfeld [111] we know that two run lengths of order 1 which are consecutive integers occur within the intersection pattern. Let these run lengths be $r+1$ and $r+2$. Therefore the chain code will consist of a sequence of the words 0^r1 and $0^{r+1}1$. When transform S is repeated r times, the words 0^r1 and $0^{r+1}1$ reduce into the words 1 and 01 respectively.

The 8–connected chain code of the line $\ell : y = \frac{17}{131}x$ within the bounded frame of the lattice $\mathcal{L}_{(131,17)}$ is $(0^71)^3\ (0^61)\ (0^71)^2\ (0^61)\ (0^71)^3\ (0^61)\ (0^71)^2\ (0^61)\ (0^71)^2\ (0^61)$. Table 4.1 gives the sequence of resultant chain codes as they as transformed by S^p, X and R.

However the connection between the chain code and continued fractions provided by McIlroy does not describe the relationship of continued fractions to the run lengths of each order found in the digitised line. For the remainder of this chapter we will provide a complete description of the connection between the gcd, continued

Transformation		Chain code
		$(0^7 1)^3$ $(0^6 1)$ $(0^7 1)^2$ $(0^6 1)$ $(0^7 1)^3$ $(0^6 1)$ $(0^7 1)^2$ $(0^6 1)$ $(0^7 1)^2$ $(0^6 1)$
S^6	$\rightarrow$	010101 1 0101 1 010101 1 0101 1 0101 1
X	$\rightarrow$	101010 0 1010 0 101010 0 1010 0 10100
R	$\rightarrow$	00101 00101 0010101 00101 0010101
S^1	$\rightarrow$	011 011 0111 011 0111
X	$\rightarrow$	100 100 1000 100 1000
R	$\rightarrow$	0001 001 0001 001 001
S^2	$\rightarrow$	01 1 01 1 1
X	$\rightarrow$	10 0 10 0 0
R	$\rightarrow$	0001 001
S^2	$\rightarrow$	01 1
X	$\rightarrow$	10 0
R	$\rightarrow$	001
S^2	$\rightarrow$	1
X	$\rightarrow$	0

Table 4.2: Transforming the chain code for slope 17/131.

fractions and the run structure of the intersection pattern of a line and a lattice. The relationship between these constructions proves extremely useful in describing the characteristics of the run structure of the intersection pattern. We will start by providing some mathematical background regarding continued fractions and the gcd algorithms based on Euclid's algorithm.

4.2 The Greatest Common Divisor (GCD)

To begin our discussion of the greatest common divisor algorithms, it is natural to commence with the notion of divisibility. Questions regarding the divisibility of one integer by another are central to the study of number theory.

Definition 4.2.1 *The integer y divides z, written $y|z$, if $y = kz$ for some integer k. Every integer divides 0. If $y|z$, y is a divisor of z.*

If the divisors of two separate integers are being considered, a standard question regards common divisors: are they divisors common to both integers?

Definition 4.2.2 *If $w|y$ and $w|z$, w is a common divisor of y and z.*

An important property of common divisors is that if w is a common divisor of y and z, for any integers u and v

$$w|(uy + vz) \tag{4.9}$$

The *greatest common divisor* is the largest of the common divisors of two numbers.

Definition 4.2.3 *If y and z are non-zero integers, their* greatest common divisor, *$\gcd(y, z)$, is the largest integer which evenly divides both y and z;* $\gcd(0, 0) = 0$.

The gcd function has a number of elementary properties

$$\gcd(y, 0) = |y| \tag{4.10}$$

$$\gcd(y, ky) = |y| \tag{4.11}$$

$$\gcd(y, z) = \gcd(z, y) \tag{4.12}$$

$$\gcd(y, z) = \gcd(y, |z|) \tag{4.13}$$

Algorithms which calculate the gcd of two parameters typically operate on the notion of replacing the existing values of the parameters with smaller values which have the same gcd until a termination condition is met. The main termination conditions used are the elementary properties $\gcd(y, 0) = |y|$, equation 4.10 and $\gcd(y, 1) = |y|$, equation 4.11.

4.2.1 Euclid's Recursive GCD Algorithm

A number of algorithms exist for calculating the gcd of two integers. The most widely known is Euclid's algorithm, a recursive algorithm based on the *gcd recursion theorem.*

Theorem 4.2.1 (GCD Recursion Theorem)
For any nonnegative integer y and any positive integer z

$$\gcd(y, z) = \gcd(y \bmod z, z) \tag{4.14}$$

Euclid's algorithm uses the gcd recursion theorem and equation 4.12, to replace the existing values of the parameters with smaller values with the same gcd.

Algorithm 4.1 Euclid's recursive gcd algorithm

```
integer GCD(y, z)
    if z = 0 then return y
   else return GCD(z, y mod z)
```

Let y_i and z_i be the values of the variables y and z at the i-th level of recursion, the transformation at each level of recursion is

$$\begin{aligned} y_{i+1} &= z_i \\ z_{i+1} &= y_i \bmod z_i \end{aligned} \tag{4.15}$$

Termination of the recursion occurs when $z = 0$, which according to equation 4.10 and the gcd of the original values is the final value of y. The number of modulus operations performed is $O(\log n)$ where $n = \max(y, z)$ [82].

4.2.2 Euclid's Subtractive GCD Algorithm

There is another common form of Euclid's algorithm which uses the following subtractive property of the gcd function

$$\gcd(y, z) = \gcd(y - z, z) \tag{4.16}$$

The subtractive property of the gcd function allows us to redesign Euclid's algorithm as such.

Algorithm 4.2 The subtractive gcd algorithm

```
integer GCD(y, z)
    while y ≠ z
        if y < z then z = z − y;
       else if y > z then y = y − z;
    return y
```

The subtractive algorithm terminates on the condition $\gcd(y, y) = y$, equation 4.11.

4.2.3 The Least Remainder GCD Algorithm

While the recursive and subtractive versions of Euclid's algorithm are the most widely known and used, another variant of Euclid's algorithm exists. The least remainder algorithm uses the GCD Recursion Theorem (theorem 4.2.1) and the subtractive property (equation 4.16) to produce an algorithm which has a faster convergence to the gcd of the two inputs than the recursive or subtractive forms of Euclid's algorithm.

Algorithm 4.3 The least remainder gcd algorithm

```
integer GCD(y, z)
    if z = 0 return y
   u = y mod z
   return GCD(z, min(u, z − u))
```

If $u < z - u$, the recursion takes the same form as Euclid's recursive algorithm. If $u > z - u$, the recursion combines a recursive step of Euclid's recursive algorithm and a subtractive step of Euclid's subtractive algorithm.

4.3 Continued Fractions

A continued fraction is a simple yet powerful construction which provides an integer–based representation of any real number. Continued fractions are closely related to Euclid's algorithm as we shall describe.

Definition 4.3.1 *A continued fraction is a development of the form*

$$\begin{aligned} & b_0 + \cfrac{a_1}{b_1 + \cfrac{a_2}{b_2 + \cfrac{a_3}{\ddots \cfrac{}{b_{n-1} + \cfrac{a_n}{b_n \ddots}}}}} \\ = \quad & b_0 + a_1/(b_1 + a_2/(b_2 + a_3/(\cdots/(b_{n-1} + a_n/b_n)\cdots))) \end{aligned} \tag{4.17}$$

Due to the unwieldy notation of continued fractions, a menagerie of alternative notations have been used for describing continued fractions. We shall base our notation on that described by Khovanskii [78].

$$\left[b_0;\ \frac{a_1}{b_1}, \frac{a_2}{b_2}, \ldots, \frac{a_n}{b_n}, \ldots\right] \tag{4.18}$$

$$= \left[\frac{a_1}{b_1}, \frac{a_2}{b_2}, \ldots, \frac{a_n}{b_n}, \ldots\right] \text{ when } b_0 = 0 \tag{4.19}$$

Continued fractions have the following structural definitions

- The fraction $\frac{a_n}{b_n}$ is called the *n-th partial quotient* of the continued fraction.
- a_n and b_n are the *coefficients* of the continued fraction.
- a_1, a_2 and a_3 are called the *partial numerators* of the continued fraction.
- b_1, b_2 and b_3 are called the *partial denominators* of the continued fraction.
- The terminating continued fraction

$$\frac{P_n}{Q_n} = \left[b_0;\ \frac{a_i}{b_i}, \ldots \frac{a_n}{b_n}\right] \tag{4.20}$$

  is called the *n-th convergent* of the continued fraction.

The n-th convergent can be recursively calculated from the values of the two previous convergents by defining $P_0 = 0$, $Q_0 = 1$, $P_{-1} = 1$ and $Q_{-1} = 0$ and the relationship

$$\frac{P_n}{Q_n} = \frac{b_n P_{n-1} + a_n P_{n-2}}{b_n Q_{n-1} + a_n Q_{n-2}} \tag{4.21}$$

4.3.1 Ordinary and Regular Continued Fractions

An *ordinary continued fraction* is any continued fraction where $a_n = 1$ for $n = 1, 2, 3 \ldots$. An ordinary continued fraction with positive integer partial denominators is called a *regular continued fraction* and has the form

$$x_0 + \cfrac{1}{x_1 + \cfrac{1}{x_2 + \cfrac{1}{\ddots \cfrac{}{x_{n-1} + \cfrac{1}{x_n \ddots}}}}} = x_0 + \left[1;\ \frac{1}{x_i}, \ldots \frac{1}{x_n}, \ldots\right] \tag{4.22}$$

Any continued fraction can be transformed into an ordinary continued fraction, where

$$x_n = a_n \prod_{k=1}^{n} b_k^{(-1)^{n+k-1}};\ n = 1, 2, \ldots \tag{4.23}$$

We will adapt our notation for continued fractions for ordinary continued fractions as such

$$[x_0;\ x_1, x_2, \ldots, x_n, \ldots] \tag{4.24}$$

$$= \quad [x_1, x_2, \ldots, x_n, \ldots] \text{ when } x_0 = 0 \tag{4.25}$$

The recurrence relation describing the correspondance between three consecutive convergents for ordinary continued fractions, equivalent to equation 4.21, is

$$\frac{P_n}{Q_n} = \frac{x_n P_{n-1} + P_{n-2}}{x_n Q_{n-1} + Q_{n-2}} \tag{4.26}$$

4.3.2 Calculating the Regular Continued Fraction Expansion of a Real Number

To generate the regular continued fraction expansion of a real number $0 \leq \alpha < 1$ the *continued fraction algorithm* is used.

$$\alpha = \left[\frac{1}{x_1}, \frac{1}{x_2}, \ldots, \frac{1}{x_n + \alpha_n}\right] \tag{4.27}$$

where

$$x_{n+1} = \left\lfloor \frac{1}{\alpha_n} \right\rfloor \tag{4.28}$$

$$\alpha_{n+1} = \frac{1}{\alpha_n} - x_{n+1} \tag{4.29}$$

It should be noted that if α is rational, the continued fraction representation is finite and exact. If α is irrational, the continued fraction representation is infinite and the convergents provide the best possible rational approximation in a certain sense. In both cases, the order of approximation improves exponentially.

4.3.3 Convergence

To summarise the convergence properties of continued fractions, let α be the number to be approximated. Therefore $\alpha = \lim_{n\to\infty} \frac{P_n}{Q_n}$.

The first property regards the continued fraction expansion of a rational number.

1. A positive rational number has a finite regular continued fraction expansion and conversely.

The next two properties concern the continued fraction approximation of an irrational number.

2. Every positive irrational number α has an infinite regular continued fraction expansion whose convergents converge to it.

3. The convergents of every infinite regular continued fraction, converge to an irrational number α.

The fourth property describes how the n-th convergent is the best rational approximation of the number being approximated (Theorem 182, [71]). That is, there does not exist an integer w such that $|\alpha - \frac{P_n}{Q_n}| > |\alpha - \frac{w}{Q_n}|$.

4. If $|\alpha - r/s| < |\alpha - \frac{P_n}{Q_n}|$, $r, s \in \mathcal{N}$, $r > 0$, $n > 1$, *then* $s > b_n$.

The fifth property provides a bound for the precision of the approximation.

5. $\left|\alpha - \frac{P_n}{Q_n}\right| < \frac{a_1 a_2 \ldots a_n}{Q_{n-1} Q_{n+1}}$

The last property shows how the even numbered convergents approach the number from below where odd numbered convergents approach from above the number being approximated.

6. $\frac{P_{2n-2}}{Q_{2n-2}} < \frac{P_{2n}}{Q_{2n}} < \alpha < \frac{P_{2n+1}}{Q_{2n+1}} < \frac{P_{2n-1}}{Q_{2n-1}}$.

4.3.4 Euclid's Recursive Algorithm and Regular Continued Fractions

When α is a rational number, the algorithm for calculating the regular continued fraction is closely related to the process of Euclid's algorithm. The recurrence relations for Euclid's algorithm from equation 4.15 are

$$\begin{aligned} y_{i+1} &\equiv z_i \pmod{y_i} \\ z_{i+1} &= y_i \end{aligned}$$

where y_i and z_i are the values of the y and z variables in the gcd algorithm at level i in the recursion. Set $\alpha_0 = \frac{y_0}{z_0}$ where $z_0 > y_0 \geq 0$. The continued fractions algorithm can then be transformed into the following equations

$$x_{n+1} = \left\lfloor \frac{1}{\alpha_n} \right\rfloor = \left\lfloor \frac{z_n}{y_n} \right\rfloor \tag{4.30}$$

$$\alpha_{n+1} = \frac{1}{\alpha_n} - x_{n+1} = \frac{z_n}{y_n} - \left\lfloor \frac{z_n}{y_n} \right\rfloor = \frac{z_n - y_n \left\lfloor \frac{z_n}{y_n} \right\rfloor}{y_n} = \frac{z_n \bmod y_n}{y_n} \tag{4.31}$$

Therefore as $\alpha_{n+1} = \frac{y_{n+1}}{z_{n+1}}$, we have

$$x_{n+1} = \frac{z_n}{y_n} \tag{4.32}$$

$$y_{n+1} \equiv z_n \pmod{y_n} \tag{4.33}$$

$$z_{n+1} = y_n \tag{4.34}$$

which correspond to the transformations made on the variables y and z in Euclid's algorithm, equation 4.15.

4.4 The Least Remainder GCD Algorithm and the Run Parameters

To describe the relationship between the structure of the hierarchy of runs in the intersection pattern, $\mathcal{P}_\ell$, of the line $\ell : y = \frac{a}{d}x$ and the lattice $\mathcal{L}_{(d,a)}$ and the least remainder gcd algorithm, the correlation between the parameters used within the algorithm and the structural run parameters $p^{[i]}$, $n^{[i]}$ and $m^{[i]}$ will be elaborated.

From definition 2.2.4 in Chapter 2 we have that the total number of runs of order i in the intersection pattern $\mathcal{P}_\ell$ is $p^{[i]}$. Let $u^{[i]}$ be the number of long runs of order i and $v^{[i]}$ the number of short runs. We have

$$\begin{aligned} p^{[0]} &= d \\ p^{[1]} &= a \end{aligned}$$

For $i > 1$

$$\begin{aligned} u^{[i]} &\equiv p^{[i]} \pmod{p^{[i]}} \\ v^{[i]} &= p^{[i]} - u^{[i]} \\ p^{[i+1]} &= \min(u^{[i]}, v^{[i]}) \end{aligned}$$

Therefore if y_i, z_i and u_i represent the values of the parameters y, z and u in the i-th level of recursion of the least remainder gcd algorithm, we have

$$\begin{aligned} y_i &= p^{[i-1]} \\ z_i &= p^{[i]} \\ u_i &= u^{[i]} \end{aligned}$$

The least remainder algorithm could therefore be rewritten as

Algorithm 4.4 The least remainder gcd algorithm in terms of the run parameters

$i = 1$
integer $GCD(p^{[i-1]}, p^{[i]})$
  if $p^{[i]} = 0$ **return** $p^{[i-1]}$
  $u^{[i]} = p^{[i]} \bmod p^{[i-1]}$
  $v^{[i]} = p^{[i-1]} - u^{[i]}$
  $p^{[i+1]} = \min(u^{[i]}, v^{[i]})$
  return $GCD(p^{[i]}, p^{[i+1]})$

The relationship between the number of long and short runs of order i and the least remainder gcd algorithm can be used to describe the relationship of the structural run parameters of order i, $p^{[i]}$, $m^{[i]}$ and $n^{[i]}$ and the least remainder gcd algorithm.

For order 1

$$\begin{aligned} n^{[i]} &= u^{[i]} = u_i \\ m^{[i]} &= v^{[i]} = z_i - u_i \end{aligned}$$

For order $i > 1$ type $t^{[i]} = 0$, $n^{[i]} < m^{[i]}$

$$\begin{aligned} m^{[i]} &\equiv p^{[i-1]} \bmod\ p^{[i]} = u^{[i]} = u_i \\ n^{[i]} &= p^{[i]} - m^{[i]} = v^{[i]} = z_i - u_i \end{aligned}$$

For order $i > 1$ type $t^{[i]} = 1$, $m^{[i]} < n^{[i]}$

$$\begin{aligned} n^{[i]} &\equiv p^{[i-1]} \bmod\ p^{[i]} = u^{[i]} = u_i \\ m^{[i]} &= p^{[i]} - n^{[i]} = v^{[i]} = z_i - u_i \end{aligned}$$

Therefore the relationship between the run parameters $p^{[i]}$, $m^{[i]}$ and $n^{[i]}$ and the least remainder gcd algorithm is shown.

4.5 Continued Fractions and the Run Parameters

What is evident from the run parameters algorithm is that the calculation of the run lengths is similar to the calculation of the partial denominators of continued fractions as noted by Pitteway [103]. However as we have shown in the discussion regarding Pitteway's suggestion, the connection is not with ordinary continued fractions.

We will show a simple alteration to the continued fractions algorithm for calculating the regular continued fraction expansion of a rational number $\frac{a}{d}$ such that the partial denominators of the continued fraction expansion generated are the run lengths of each order of the run hierarchy of the intersection pattern of the line $y = \frac{a}{d}x$. From this algorithm we will also show the relationship with the parameters used in the algorithm and the structural run parameters $p^{[i]}$, $n^{[i]}$ and $m^{[i]}$ and the type $t^{[i]}$ for all orders within the hierarchy of runs.

An important application of this result is when considering the intersection pattern of a line with an irrational slope. From theorem 3.3.2 we know that the n-th convergent of the regular continued fraction expansion of α provides a suitable approximation of α such that the lines $\ell : y = \alpha x$ and $\ell' : y = \frac{P_n}{Q_n}x$ produce identical intersection patterns in the frame $\mathcal{L}_{(P_n,Q_n)}$. Therefore understanding the relationship between continued fractions and the structure of the run hierarchy of the intersection pattern allows us to calculate the run parameters for all orders during the approximation process.

4.5.1 The Continued Fraction Expansion of a Rational Slope

Theorem 4.5.1 *The continued fractions algorithm can be used to generate a continued fraction expansion where the partial denominator at level i within the fraction is the run length of order i, $r^{[i]}$, within the intersection pattern, $\mathcal{P}_\ell$. If the partial denominator calculated at level j, $x_j > 1$, $r^{[i]} = x_j$. If the partial denominator calculated at level j, $x_j = 1$, $r^{[i]} = x_j + x_{j+1}$.*

Proof : To show the theorem we will form an inductive proof starting at order

1 which coincides with the first iteration of the continued fraction algorithm. We rewrite the continued fraction algorithm to calculate the continued fraction expansion of a rational number where $\alpha_j = \frac{y_j}{z_j}$ and $y_j < z_j$. We have from equations 4.30, 4.33 and 4.34

$$\begin{aligned} x_{j+1} &= \left\lfloor \frac{z_j}{y_j} \right\rfloor \\ y_{j+1} &\equiv z_j \pmod{y_j} \\ z_{j+1} &= y_j \end{aligned}$$

Order 1

Let $z_0 = p^{[0]} = d$ and $y_0 = p^{[1]} = a$. For the first iteration of the continued fraction algorithm we have

$$\begin{aligned} x_1 &= \left\lfloor \frac{z_0}{y_0} \right\rfloor = \left\lfloor \frac{p^{[0]}}{p^{[1]}} \right\rfloor = r^{[1]} \\ y_1 &\equiv z_0 \pmod{y_0} \equiv p^{[0]} \pmod{p^{[1]}} = n^{[1]} \\ z_1 &= y_0 = p^{[1]} \end{aligned}$$

Order 2

For the second iteration, if $n^{[1]} < m^{[1]}$ we have type $t^{[2]} = 0$ and $p^{[2]} = n^{[1]}$. Therefore

$$\begin{aligned} x_2 &= \left\lfloor \frac{z_1}{y_1} \right\rfloor = \left\lfloor \frac{p^{[1]}}{n^{[1]}} \right\rfloor = \left\lfloor \frac{p^{[1]}}{p^{[2]}} \right\rfloor = r^{[2]} \\ y_2 &\equiv z_1 \pmod{y_1} \equiv p^{[1]} \pmod{p^{[2]}} = m^{[2]} \\ z_2 &= y_1 = p^{[2]} \end{aligned}$$

However if $n^{[1]} > m^{[1]}$ and we have type $t^{[2]} = 1$, $p^{[2]} = m^{[1]}$.

$$\begin{aligned} x'_2 &= \left\lfloor \frac{z_1}{y_1} \right\rfloor = \left\lfloor \frac{p^{[1]}}{n^{[1]}} \right\rfloor = \left\lfloor \frac{p^{[1]}}{p^{[1]} - m^{[1]}} \right\rfloor = \left\lfloor \frac{p^{[1]}}{p^{[1]} - p^{[2]}} \right\rfloor \\ y'_2 &\equiv z_1 \pmod{y_1} \equiv p^{[1]} \pmod{n^{[1]}} \\ z'_2 &= y_1 = n^{[1]} \end{aligned}$$

We know that $p^{[2]} < \frac{p^{[1]}}{2}$ and $x'_2 \in \mathcal{N}$ therefore $\frac{p^{[1]}}{p^{[1]} - p^{[2]}} < \frac{p^{[1]}}{p^{[1]} - \frac{p^{[1]}}{2}} = 2$ and $x'_2 = 1$. Also $n^{[1]} > \frac{p^{[1]}}{2}$ therefore $y'_2 \equiv p^{[1]} \pmod{n^{[1]}} = m^{[1]} = p^{[2]}$. To continue, we have

$$x'_2 = 1$$

$$
\begin{aligned}
y_2' &= m^{[1]} = p^{[2]} \\
z_2' &= n^{[1]} \\
x_3' &= \left\lfloor \frac{z_2'}{y_2'} \right\rfloor = \left\lfloor \frac{n^{[1]}}{m^{[1]}} \right\rfloor
\end{aligned}
$$

Therefore $x_2' + x_3' = \left\lfloor \frac{n^{[1]}}{m^{[1]}} \right\rfloor + 1 = \left\lfloor \frac{n^{[1]}}{m^{[1]}} + 1 \right\rfloor = \left\lfloor \frac{n^{[1]}+m^{[1]}}{p^{[2]}} \right\rfloor = \left\lfloor \frac{p^{[1]}}{p^{[2]}} \right\rfloor = r^{[2]}$ and we have

$$
\begin{aligned}
x_3' &= r^{[2]} - 1 \\
y_3' &\equiv z_2' \pmod{y_2'} \equiv n^{[1]} \pmod{m^{[1]}} \equiv p^{[1]} \pmod{p^{[2]}} = n^{[2]} \\
z_3' &= y_2' = p^{[2]}
\end{aligned}
$$

Therefore for the two different cases of type we have $x_2 = x_2' + x_3'$, $y_2 = y_3'$ and $z_2 = z_3'$.

Order i

For the j-th iteration, we will investigate the two cases where $y_j = n^{[i-1]}$ and $y_j = m^{[i-1]}$. For the first case, assume $x_j = r^{[i]}$, $y_j = n^{[i-1]}$ and $z_j = p^{[i-1]}$.

$$
\begin{aligned}
x_{j+1} &= \left\lfloor \frac{z_j}{y_j} \right\rfloor = \left\lfloor \frac{p^{[i-1]}}{n^{[i-1]}} \right\rfloor \\
y_{j+1} &\equiv z_j \pmod{y_j} \equiv p^{[i-1]} \pmod{n^{[i-1]}} \\
z_{j+1} &= y_j = n^{[i-1]}
\end{aligned}
$$

If $n^{[i-1]} < m^{[i-1]}$ and we have type $t^{[i]} = 0$, $p^{[i]} = n^{[i-1]}$. Therefore

$$
\begin{aligned}
x_{j+1} &= \left\lfloor \frac{p^{[i-1]}}{p^{[i]}} \right\rfloor \\
y_{j+1} &\equiv p^{[i-1]} \pmod{p^{[i]}} = m^{[i]} \\
z_{j+1} &= p^{[i]}
\end{aligned}
$$

If $n^{[i-1]} > m^{[i-1]}$ and we have type $t^{[i]} = 1$, $p^{[i]} = m^{[i-1]}$.

$$
\begin{aligned}
x_{j+1}' &= \left\lfloor \frac{p^{[i-1]}}{n^{[i-1]}} \right\rfloor = \left\lfloor \frac{p^{[i-1]}}{p^{[i-1]} - m^{[i-1]}} \right\rfloor = \left\lfloor \frac{p^{[i-1]}}{p^{[i-1]} - p^{[i]}} \right\rfloor \\
y_{j+1}' &\equiv p^{[i-1]} \pmod{n^{[i-1]}} \\
z_{j+1}' &= y_1 = n^{[i-1]}
\end{aligned}
$$

We know that $p^{[i]} < \frac{p^{[i-1]}}{2}$ and $x'_2 \in \mathcal{N}$ therefore $\frac{p^{[i-1]}}{p^{[i-1]}-p^{[i]}} < \frac{p^{[i-1]}}{p^{[i-1]}-\frac{p^{[i-1]}}{2}} = 2$ and $x'_2 = 1$. Also $n^{[i-1]} > \frac{p^{[i-1]}}{2}$ therefore $y'_2 \equiv p^{[i-1]} \pmod{n^{[i-1]}} = m^{[i-1]} = p^{[i]}$. To continue, we have

$$\begin{aligned}
x'_{j+1} &= 1 \\
y'_{j+1} &= m^{[i-1]} = p^{[i]} \\
z'_{j+1} &= n^{[i-1]} \\
x'_{j+2} &= \left\lfloor \frac{z'_{j+1}}{y'_{j+1}} \right\rfloor = \left\lfloor \frac{n^{[i-1]}}{m^{[i-1]}} \right\rfloor
\end{aligned}$$

Therefore

$$x'_{j+1} + x'_{j+2} = \left\lfloor \frac{n^{[i-1]}}{m^{[i-1]}} \right\rfloor + 1 = \left\lfloor \frac{n^{[i-1]}}{m^{[i-1]}} + 1 \right\rfloor = \left\lfloor \frac{n^{[i-1]} + m^{[i-1]}}{p^{[i]}} \right\rfloor = \left\lfloor \frac{p^{[i-1]}}{p^{[i]}} \right\rfloor = r^{[i]}$$

and we have

$$\begin{aligned}
x'_{j+2} &= r^{[i]} - 1 \\
y'_{j+2} &\equiv z'_{j+1} \pmod{y'_{j+1}} \equiv n^{[i-1]} \pmod{m^{[i-1]}} \equiv p^{[i-1]} \pmod{p^{[i]}} = n^{[i]} \\
z'_{j+2} &= y'_{j+1} = p^{[i]}
\end{aligned}$$

Therefore for the two different cases of type we have $x_{j+1} = x'_{j+1} + x'_{j+2}$, $y_{j+1} = z_{j+1} - y'_{j+2}$ and $z_{j+1} = z'_{j+2}$.

The second case, where $y_j = m^{[i-1]}$, follows a similar argument to the first. By induction we have the theorem.

$\square$

It is useful to note the state of the run structure that corresponds to the calculation of a non-ordinary convergent in the continued fraction expansion. In the continued fraction algorithm process this corresponds to the calculation of a partial denominator of $x_n = 1$. As we have discussed, a run length of 1 has no meaning for orders greater than 1. This occurs when $t^{[2]} = 1$ at order 2 or $t^{[i-1]} = t^{[i]}$ at order $i > 2$.

Note also that if α is rational, the last partial denominator x_n must be greater than 1 from equation 4.29 when $\alpha_n = 0$ since $\alpha_{n-1} < 1$. Therefore the final run length $r^{[i_{max}]} > 1$.

So we have shown the connection between the continued fractions algorithm, the structural run parameters, $p^{[i]}$, $n^{[i]}$ and $m^{[i]}$ and the run length, $r^{[i]}$. The results of which are summarised in Algorithm 4.5.

Algorithm 4.5 Extending the continued fractions algorithm to calculate the run parameters for all orders.

integer $runParameters(p^{[1]}, p^{[0]})$
{For order 1 (type 1 by definition)}
$z_0 = p^{[0]}$
$z_1 = p^{[1]}$
{Calculate the first run length}
$r^{[1]} = \lfloor \frac{z_0}{z_1} \rfloor$
{Calculate the next value of y}
$y_1 = z_0 - z_1 r^{[1]}$
{The values for m and n}
$n^{[1]} = y_1$
$m^{[1]} = p^{[1]} - n^{[1]}$

while $z_i > 1$
    $z_{i+1} = y_i$
    $x_{i+1} = \lfloor \frac{z_i}{y_i} \rfloor$

    {If the partial denominator x_{i+1} is greater than 1,
    no extra iteration is needed.}
    if $x_{i+1} > 1$
        $y_{i+1} = z_i - z_{i+1} x_{i+1}$
        {The run length and structural run parameters}
        $r^{[j+1]} = x_{i+1}$
        $p^{[j+1]} = z_{i+1}$
        $m^{[j+1]} = y_{i+1}$
        $n^{[j+1]} = p^{[j+1]} - m^{[j+1]}$

    {Else, the partial denominator x_{i+1} is 1,
    need to add another iteration of the cf algorithm}
    else
        $z_{i+2} = z_i - z_{i+1}$
        $x_{i+2} = \lfloor \frac{z_{i+1}}{y_{i+1}} \rfloor$
        $y_{i+2} = z_{i+1} - z_{i+2} x_{i+2}$
        {The run length and structural run parameters}
        $r^{[j+1]} = x_{i+2} + 1$
        $p^{[j+1]} = z_{i+2}$
        $n^{[j+1]} = y_{i+2}$
        $m^{[j+1]} = p^{[j+1]} - n^{[j+1]}$

4.5.2 The Continued Fraction Expansion of an Irrational Slope

From theorem 3.3.2 we have that the n-th convergent of the regular continued fraction expansion of α provides a suitable approximation of α such that the lines $\ell : y = \alpha x$ and $\ell' : y = \frac{P_n}{Q_n}x$ produce identical intersection patterns in the frame $\mathcal{L}_{(P_n,Q_n)}$.

We will use this result to provide an algorithm for characterising the structure of the hierarchy of runs of the intersection pattern $\mathcal{P}_\ell$ of the line ℓ within the given frame $\mathcal{L}_{(f_x,f_y)}$. The continued fraction expansion of α is calculated and the n-th convergent, $\frac{P_n}{Q_n}$, where $Q_n \geq f_x$ is used as the rational approximation to the irrational slope.

The slope α is irrational and therefore the variables y_j and z_j will also be irrational such that $0 < (y_j, z_j) < 1$. However it is straightforward to define the relationship between the values of $p^{[i]}, m^{[i]}$ and $n^{[i]}$ and the values of z_j and y_j based on the result of Theorem 4.5.1. For order 1

$$\begin{aligned} p^{[1]} &= \lfloor \alpha r^{[1]} \rfloor = \lfloor z_1 Q_n \rfloor \\ n^{[1]} &= \lfloor y_1 Q_n \rfloor \end{aligned}$$

For order i, type 0

$$\begin{aligned} p^{[i]} &= \lfloor z_{j+2} Q_n \rfloor \\ m^{[i]} &= \lfloor y_{j+2} Q_n \rfloor \end{aligned}$$

For order i, type 1

$$\begin{aligned} p^{[i]} &= \lfloor z_{j+2} Q_n \rfloor \\ n^{[i]} &= \lfloor y_{j+2} Q_n \rfloor \end{aligned}$$

As the slope α is irrational, the continued fraction expansion of α will be infinite. Therefore the approximation process is ceased when the n-th convergent of the continued fraction, $\frac{P_n}{Q_n}$, is a suitable approximation of α over $\mathcal{L}_{(f_x,f_y)}$ which occurs when $Q_n \geq f_x$.

Algorithm 4.6 Extending the continued fractions algorithm to calculate the rational approximation of the slope for the frame (f, f) and the run parameters associated with the approximation.

integer $runParameters(\alpha, f)$
$z_0 = 1$
$z_1 = \alpha$
$y_1 = z_0 - z_1 r^{[1]}$
{Calculate the first run length}
$r^{[1]} = \lfloor \frac{z_0}{z_1} \rfloor$
{Store the floating point values for m and n}
$m^{[1]} = z_1 - y_1$
$n^{[1]} = y_1$
{Calculate the value of the first partial convergent.}
$P_0 = 0$
$Q_0 = 1$
$P_1 = 1$
$Q_1 = r^{[1]}$

while $Q_i < f_x$
  $z_{i+1} = y_i$
  {Calculate the next partial denominator}
  $x_{i+1} = \lfloor \frac{z_i}{y_i} \rfloor$
  {Calculate the next partial convergent of the approximation.}
  $P_{i+1} = x_{i+1} P_i + P_{i-1}$
  $Q_{i+1} = x_{i+1} Q_i + Q_{i-1}$

  {If the partial denominator x_{i+1} is greater than 1,
  no extra iteration is needed.}
  if $x_{i+1} > 1$
    $y_{i+1} = z_i - z_{i+1} x_{i+1}$
    {Store the run length}
    $r^{[j+1]} = x_{i+1}$
    {Store the floating point values for m, n and p}
    $p^{[j+1]} = z_{i+1}$
    $m^{[j+1]} = y_{i+1}$
    $n^{[j+1]} = z_{i+1} - y_{i+1}$
  {Else, the partial denominator x_{i+1} is 1,
  need to add another iteration of the cf algorithm}
  else
    $x_{i+2} = \lfloor \frac{z_{i+1}}{y_{i+1}} \rfloor$
    $z_{i+2} = z_i - z_{i+1}$
    $y_{i+2} = z_{i+1} - z_{i+2} x_{i+2}$
    {Calculate the run length}
    $r^{[j+1]} = x_{i+2} + 1$
    {Store the floating point values for m, n and p}
    $p^{[j+1]} = z_{i+2}$

$n^{[j+1]} = y_{i+2}$
$m^{[j+1]} = z_{i+2} - y_{i+2}$
{Calculate the next partial convergent of the approximation.}
$P_{i+2} = x_{i+2}P_{i+1} + P_i$
$Q_{i+2} = x_{i+2}Q_{i+1} + Q_i$

{Calculate the integer values for the parameters m, n and p for all orders
- where P_n/Q_n is the final partial convergent calculated.}
for $i = 0$ **to** n
$m^{[i]} = m^{[i]}Q_n$
$n^{[i]} = n^{[i]}Q_n$
$p^{[i]} = p^{[i]}Q_n$

4.6 Applications

The connection between continued fractions, the gcd algorithm and the hierarchy of runs within the intersection pattern of the line and the lattice allows us to describe several properties of the run parameters and the intersection pattern. It also allows us to use the extensive analysis performed on Euclid's algorithm to characterise the depth of the hierarchy.

4.6.1 The GCD Properties of the Run Parameters

We will start with a simple property of the structural run parameters, $p^{[i-1]}$, $p^{[i]}$, $n^{[i]}$ and $m^{[i]}$ for $i \geq 1$, which all share the same greatest common divisor regardless of order.

Lemma 4.6.1 *For order $i \geq 1$, $\gcd(p^{[i]}, p^{[i-1]}) = \gcd(p^{[i]}, m^{[i]}) = \gcd(p^{[i]}, n^{[i]}) = \gcd(a, d)$.*

Proof : For order 1,

$$\begin{aligned}
\gcd(d, a) &= \gcd(p^{[0]}, p^{[1]}) = \gcd(p^{[0]} \bmod p^{[1]}, p^{[1]}) = \gcd(n^{[1]}, p^{[1]}) \\
&= \gcd(n^{[1]}, p^{[1]} - n^{[i]}) = \gcd(n^{[1]}, m^{[1]}) \\
&= \gcd(n^{[1]} + m^{[i]}, m^{[1]}) = \gcd(p^{[1]}, m^{[1]})
\end{aligned}$$

Similarly for order $i > 1$, and type $t^{[i]} = 0$,

$$\begin{aligned}
\gcd(p^{[i-1]}, p^{[i]}) &= \gcd(p^{[i-1]} \bmod p^{[i]}, p^{[i]}) = \gcd(m^{[i]}, p^{[i]}) \\
&= \gcd(m^{[i]}, p^{[i]} - m^{[i]}) = \gcd(m^{[i]}, n^{[i]})
\end{aligned}$$

And for type $t^{[i]} = 1$

$$\begin{aligned}
\gcd(p^{[i-1]}, p^{[i]}) &= \gcd(p^{[i-1]} \bmod p^{[i]}, p^{[i]}) = \gcd(n^{[i]}, p^{[i]}) \\
&= \gcd(n^{[i]}, p^{[i]} - n^{[i]}) = \gcd(n^{[i]}, m^{[i]})
\end{aligned}$$

By induction the lemma is shown. □

4.6.2 The Depth of the Hierarchy of Runs

The relationship between the run parameters and the parameters of the least remainder gcd algorithm allow us to make some comments about the depth of the hierarchy of runs within the intersection pattern of a line and a lattice. Knuth [82] provides a rigorous analysis of Euclid's gcd algorithm. From this analysis we have an upper bound on the depth of the hierarchy of runs, i_{max}, of the line $\ell : y = \frac{a}{d}x$ and of the line $\ell : y = \alpha x$ defined within the frame $\mathcal{L}_{(d,a)}$. From the analysis of Euclid's gcd algorithm, it is evident that $i_{max} = O(\log\ d)$.

4.6.3 Equivalence of Intersection Patterns

Consider all the possible intersection patterns $\mathcal{P}_{\ell'}$ of the lines $\ell' : y = \frac{a'}{d'}x$, where $\mathcal{P}_{\ell'} \subseteq \mathcal{P}_{\ell}$. For example take the line $\ell : y = \frac{29}{41}x$ shown in figure 4.1. The set of lines defined by their slope, $L = \{\frac{a'}{d'}\}$ where $\mathcal{P}_{\ell'} \subset \mathcal{P}_{\ell}$ is

$$L = \left\{\frac{24}{34}, \frac{17}{24}, \frac{12}{17}, \frac{7}{10}, \frac{6}{9}, \frac{5}{7}, \frac{4}{6}, \frac{3}{4}, \frac{2}{3}, \frac{1}{2}, \frac{1}{1}\right\} \tag{4.35}$$

The relevant run parameters of all orders of the lines defined by the set L are listed in Table 4.3. We have divided these lines into three sets to explain their similar properties. The first set contains those lines which coincide with the first run of order $i = i'_{max} - 1$ in $\mathcal{P}_{\ell}$ and a' and d' are coprime.

$$L_0 = \left\{\frac{12}{17}, \frac{5}{7}, \frac{2}{3}, \frac{1}{1}\right\} \tag{4.36}$$

The second set contains those lines where a' and d' are coprime but do not coincide with the first run of order $i = i'_{max} = 1$ in $\mathcal{P}_\ell$. These lines are

$$L_1 = \left\{ \frac{17}{24}, \frac{7}{10}, \frac{3}{4}, \frac{1}{2} \right\} \tag{4.37}$$

The third set contains those lines which coincide with the few runs of similar length of order $i = i'_{max} - 1$ in $\mathcal{P}_\ell$ and therefore a' and d' are not coprime. These lines are

$$L_2 = \left\{ \frac{24}{34}, \frac{6}{9}, \frac{4}{6} \right\} \tag{4.38}$$

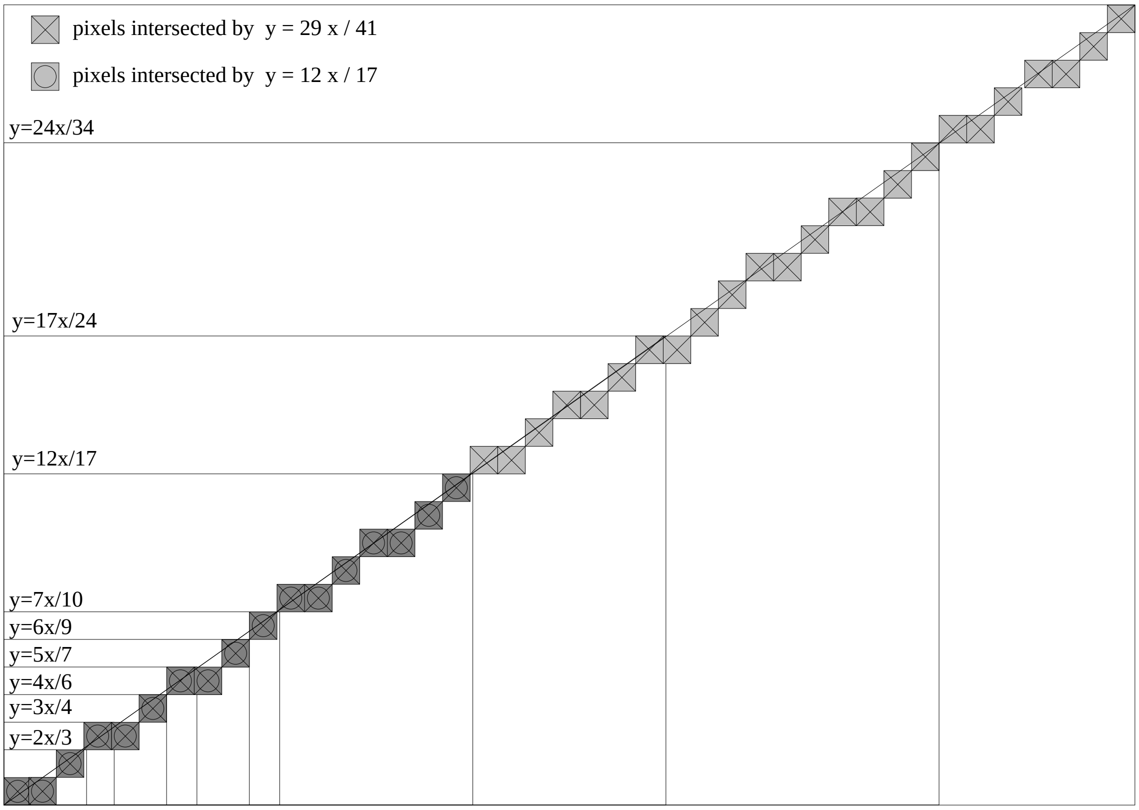


Figure 4.1: The intersection patterns of the lines $\ell' : y = \frac{a'}{d'}$ contained in the line $\ell : y = \frac{29}{41}x$ where $a' \leq 29$ and $d' \leq 41$.

$y =$	$\frac{29}{41}x$	$\frac{12}{17}x$	$\frac{5}{7}x$	$\frac{2}{3}x$	$\frac{1}{1}x$	$\frac{17}{24}x$	$\frac{7}{10}x$	$\frac{3}{4}x$	$\frac{1}{2}x$	$\frac{24}{34}x$	$\frac{6}{9}x$	$\frac{4}{6}x$
Order 0												
$p^{[0]} =$	41	17	7	3	1	24	10	4	2	34	9	6
Order 1												
$p^{[1]} =$	29	12	5	2	1	17	7	3	1	24	6	4
$r^{[1]} =$	1	1	1	1	1	1	1	1		1	1	1
Order 2												
$p^{[2]} =$	12	5	2	1		7	3	1		10	3	2
$r^{[2]} =$	2	2	2	2		2	2	2		2	2	2
$t^{[2]} =$	0	0	0	na		0	0	0		0	na	na
Order 3												
$p^{[3]} =$	5	2	1			3	1			4		
$r^{[3]} =$	2	2	2			2	2			2		
$t^{[3]} =$	1	1	na			1	1			1		
Order 4												
$p^{[4]} =$	2	1				1				2		
$r^{[4]} =$	2	2				2				2		
$t^{[4]} =$	0	na				0				na		
Order 5												
$p^{[5]} =$	1											
$r^{[5]} =$	1											
$t^{[5]} =$	na											

Table 4.3: The run parameters of the lines ℓ' that produce similar intersection patterns to ℓ.

To explain the correspondence between the lines of sets one and two and ℓ, we need look no further than the continued fraction expansion of the slope of ℓ.

$$\frac{29}{41} = \cfrac{1}{1+\cfrac{1}{2+\cfrac{1}{2+\cfrac{1}{2+\cfrac{1}{2}}}}}$$

$$\frac{12}{17} = \cfrac{1}{1+\cfrac{1}{2+\cfrac{1}{2+\cfrac{1}{2}}}} \qquad \frac{17}{24} = \cfrac{1}{1+\cfrac{1}{2+\cfrac{1}{2+\cfrac{1}{(2+1)}}}}$$

$$\frac{5}{7} = \cfrac{1}{1+\cfrac{1}{2+\cfrac{1}{2}}} \qquad \frac{7}{10} = \cfrac{1}{1+\cfrac{1}{2+\cfrac{1}{(2+1)}}}$$

$$\frac{2}{3} = \cfrac{1}{1+\cfrac{1}{2}} \qquad \frac{3}{4} = \cfrac{1}{1+\cfrac{1}{(2+1)}}$$

$$\frac{1}{1} = \frac{1}{1} \qquad \frac{1}{2} = \frac{1}{(1+1)}$$

From the results of Appendix A, we can also use the pattern matching results to provide these results. We can also account for the third set of lines that are contained in the intersection pattern. We need to consider the sequences of runs that define $\mathcal{P}_\ell$ for orders 2 to $i_{max} - 1 = 4$.

For order 4, $R_\ell^{[4]} = \{\ sl\ \}^{[[4]]}$. The subsequences of runs of order 4 that can be contained in $R_\ell^{[4]}$ are $\{\ s, l, ss\ \}^{[[4]]}$. The run sequence $\{\ ss\ \}^{[[4]]}$ is defined by the line of slope $\frac{24}{34}$, $\{\ l\ \}^{[[4]]}$ by $\frac{17}{24}$, and $\{\ s\ \}^{[[4]]}$ by $\frac{12}{17}$.

For order 3, we need only concern ourselves with the patterns smaller than the first run of order 4. We have $R_\ell^{[3]} = \{\ lslss\ \}^{[[3]]}$. The subsequences of runs of order 3 that can be contained in $R_\ell^{[3]}$ are $\{\ s, l\ \}^{[[3]]}$. The run sequence $\{\ l\ \}^{[[3]]}$ is defined by the line of slope $\frac{7}{10}$, and $\{\ s\ \}^{[[3]]}$ by $\frac{6}{9}$. Note the sequence $\{\ ll\ \}^{[[3]]}$ cannot be contained within the $\mathcal{P}_\ell$ as the type $t^{[3]} = 1$.

For order 2, $R_\ell^{[2]} = \{\ sslslsslslsl\ \}^{[[2]]}$ The subsequences of runs of order 2 that can be contained in $R_\ell^{[3]}$ and are shorter than the first run of order 3 are $\{\ s, l, ss, sl, sss\ \}^{[[2]]}$. The run sequence $\{\ sss\ \}^{[[2]]}$ is defined by the line of slope $\frac{6}{9}$, $\{\ sl\ \}^{[[2]]}$ by $\frac{5}{7}$, $\{\ ss\ \}^{[[2]]}$ by $\frac{4}{6}$, $\{\ l\ \}^{[[2]]}$ by $\frac{3}{4}$ and $\{\ s\ \}^{[[2]]}$ by $\frac{2}{3}$.

An effect of this result is that we can tighten the termination condition we imposed on the rational based approximation of a line with irrational slope in Theorem 3.3.2 in Chapter 3 from $f_x \leq Q_{n-1}$ to $f_x \leq Q_n$. Thus saving an iteration of the continued fraction algorithm used to produce the rational approximation. This is because we have shown the intersection pattern of the line $y = \frac{P_{n-1}}{Q_{n-1}}x$ is contained within the intersection pattern of $y = \frac{P_n}{Q_n}x$.

Theorem 4.6.2 *The intersection patterns $\mathcal{P}_\ell$ and $\mathcal{P}_{\ell'}$ of the lines $\ell : y = \alpha x$ and $\ell' : y = \frac{P_n}{Q_n}x$, are identical in the frame $\mathcal{L}_{(f_x, f_y)}$ where α is irrational, P_n/Q_n is the n-th partial convergent of the regular continued fraction expansion of α and $f_x \leq Q_n$.*

4.7 Conclusions

The profound connection between number theory and the digitised line has been recogonised by a number of researchers. However a full description of the link between continued fractions and the hierarchy of runs has not been provided until now. The link produces a number of useful results. The knowledge we have about Euclid's algorithm can be applied to describing the depth of the hierarchy of runs in the digitised line and therefore also the algorithms which produce it [45, 22]. Properties of the run parameters can be explained, such as the common greatest divisor of the parameters $p^{[i]}, m^{[i]}$ and $n^{[i]}$. Finally the lines of lesser length which produce an indentical intersection pattern over their length to a line can be produced. However as we have shown through the pattern or run matching results presented in

Appendix A a more complete method exists. The link between run length and continued fractions also allows us to consider continued fractions in a purely geometric light.

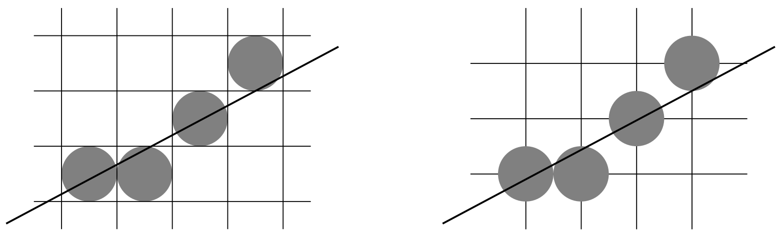
Chapter 5

Application to Line Drawing in Computer Graphics

5.1 Introduction

Line drawing is one of the oldest and most fundamental problems in computer graphics. It can be described as which pixels best describe or digitise a line defined between two pixels (x_0, y_0) and (x_1, y_1). We will assume that $x_0 < x_1$ and $y_0 < y_1$.

Assuming we are working with a rectilinear pixel space, the lattice is an obvious abstraction, where a pixel can be defined by a lattice point or the interior of a cell of the lattice. We will define a pixel as the interior of a lattice cell.



(a) A pixel defined as the interior of a cell.

(b) A pixel defined at a lattice position.

Figure 5.1: Possible abstractions of pixel space by a lattice.

The problem of which lattice cells provide the best digital representation of a line defined between two pixel locations (x_0, y_0) and (x_1, y_1) can be answered by calculating the 8–connected intersection pattern of the line $y = \frac{y_1 - y_0}{x_1 - x_0}x + \frac{1}{2}$ between the cells (x_0, y_0) and (x_1, y_1). An example of the intersection pattern generated by the line $y = \frac{17}{41}x + \frac{1}{2}$ is given in Figure 5.2 and represents the line drawn from pixel $(0, 0)$ to $(41, 17)$.

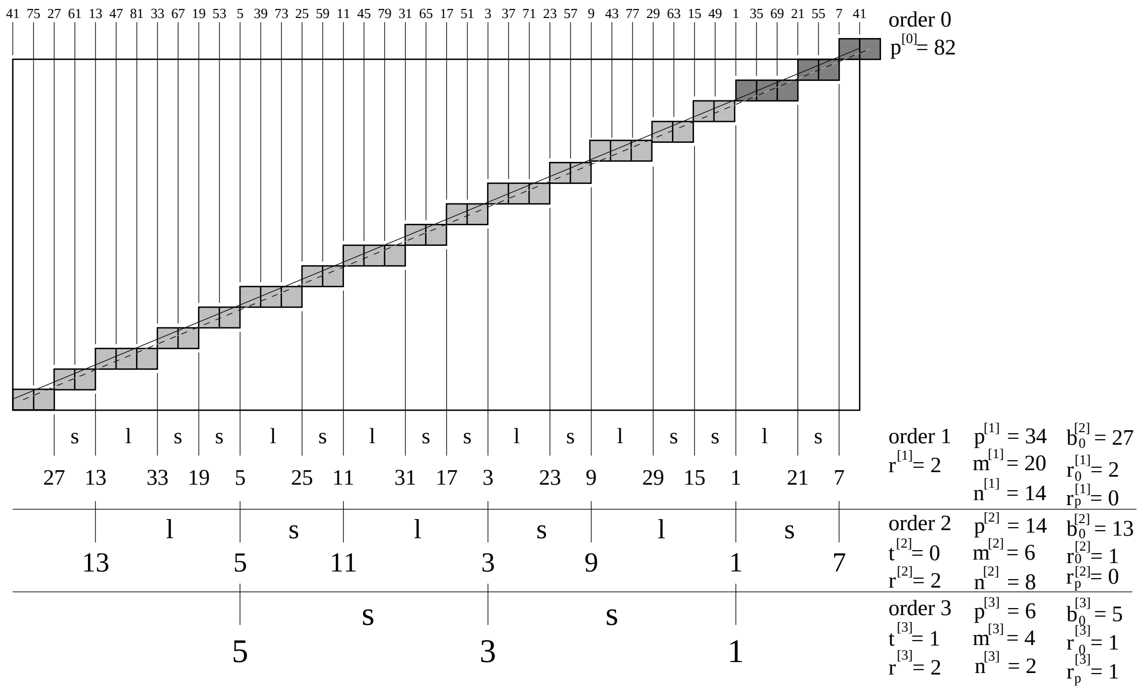


Figure 5.2: Translation of the line for use in the line drawing algorithm.

It is apparent that the line to be drawn will always have a rational slope and intercept as the problem is restricted to using integer endpoints. We wish to form a line $\ell : y = \frac{a}{d}x + \frac{b}{d}$ such that the denominators of the slope and intercept are equal. We will use the line $\ell : y = \frac{2(y_1 - y_0)}{2(x_1 - x_0)}x + \frac{x_1 - x_0}{2(x_1 - x_0)}$ and have $a = 2(y_1 - y_0)$, $d = 2(x_1 - x_0)$ and $b = x_1 - x_0$.

We will present two types of algorithms for line drawing based on the hierarchy of runs within the intersection pattern. The first type of algorithm will be based on the numerator sequence of a certain order. The orders we will discuss are orders $i = 0, 1$ and 2. Therefore the order i algorithm will be based on the order i numerator sequence and will build the intersection pattern by setting runs of order i. The second type of algorithm to be discussed in a recursive algorithm that utilises every order within the intersection pattern. This algorithm only uses the initial numerator values of each order and does not calculate the entire numerator sequence of any order.

5.2 Line Drawing Algorithms based on the Numerator Sequence

We present three algorithms for drawing a line between the pixels (x_0, y_0) and (x_1, y_1) based on the numerator sequence of orders 0, 1 and 2.

We will start by initialising the variables which are common to all three algorithms.

$$\Delta x = x_1 - x_0$$
$$\Delta y = y_1 - y_0$$
$$p^{[0]} = d = 2\Delta x$$
$$p^{[1]} = a = 2\Delta y$$
$$b = \Delta x$$

5.2.1 The Order 0 Algorithm

While we have not regarded order 0 in our discussion of the structure of the hierarchy of runs in the intersection pattern of the line, with a small change to our definition it is straightforward to define an order 0 algorithm. To declare an order 0 algorithm we define $m^{[0]} = p^{[1]} = a$ and $n^{[0]} = p^{[0]} - m^{[0]} = d - a$. If $b_j^{[0]} > b_{j+1}^{[0]}$ the next pixel is $(x+1, y+1)$. If not, the next pixel is $(x+1, y)$.

We have included the order 0 algorithm because it will form a basis for our discussions regarding the order 1 and order 2 algorithms, all of which possess a similar structure. The order 0 algorithm is based on the primitive which sets runs of order 0, the familiar $setPixel(x, y)$ command. The algorithm is constructed by firstly initialising the relevant variables. They are the position of the current run (x, y), the decision variable $b_j^{[0]}$ and the two decision increments $m^{[0]}$ and $n^{[0]}$.

The position of the initial run of order 0 is $(x, y) = (x_0, y_0)$. The initial value of the decision variable is the initial numerator value, $b_0^{[0]} = b$. However following the discussion in section 2.5 we subtract $n^{[0]}$ from this value to align the critical decision value with 0, therefore $b_0^{[0]} = b - n^{[0]}$.

Algorithm 5.1 The Order 0 Line Drawing Algorithm.

$\Delta x = x_1 - x_0$
$\Delta y = y_1 - y_0$
$m^{[0]} = 2\Delta y$
$n^{[0]} = 2(\Delta x - \Delta y)$
$b_0^{[0]} = b - n^{[0]} = \Delta x - 2dy$

$setPixel(x, y)$
while $x < x_1$
    $x = x + 1$
    if $b_j^{[0]} < 0$
        $y = y + 1$
        $b_{j+1}^{[0]} = b_j^{[0]} + m^{[0]}$
    else
        $b_{j+1}^{[0]} = b_j^{[0]} - n^{[0]}$
    $setPixel(x, y)$

The reader should note that the algorithm is identical to Bresenham's line drawing algorithm, except for some sign changes. We will discuss the basis for this correspondance further in Appendix B.

5.2.2 The Order 1 Algorithm

Let us now consider the order 1 algorithm. This algorithm is based on the primitive $setRunOrder1(x, y, r)$ which sets a run of length r at position (x, y).

As we are constructing the intersection pattern of the line $y = \frac{a}{d}x + \frac{1}{2}$, the construction of the algorithm follows the discussion in section 3.2. Firstly the run parameters are initialised. Then the initial partial run length is calculated and the initial partial run set. The initial numerator value of order 1 is calculated and the full length runs of order 1 placed in the intersection pattern. To complete the algorithm the final partial run length is calculated and the run set.

However, due to the unique construction of the intersection pattern based on the intercept value of $\frac{1}{2}$ we can improve on the results discussed in section 3.2. We will therefore work through the algorithm due to the special situation we have with an intercept of $\frac{1}{2}$.

Initialisation

The first undertaking is to calculate the initial run length and initial numerator value of order 1.

From Theorem 3.2.5, we know that if $a > d - a$ or more simply $a > b$ then $r^{[1]} = 1$ and therefore $r_0^{[1]} = 0$ and $b_0^{[1]} = b$. If $a \leq b$ the initial run length of order 1, $r_0^{[1]}$, can be calculated by equation 2.1.

$$r_0^{[1]} = \left\lceil \frac{d-b}{a} \right\rceil = \left\lceil \frac{p^{[0]}}{2p^{[1]}} \right\rceil = \left\lceil \frac{r^{[1]}}{2} + \frac{n^{[1]}}{2p^{[1]}} \right\rceil = \left\lceil \frac{r^{[1]}}{2} \right\rceil \tag{5.1}$$

since $n^{[1]} < p^{[1]}$ and $\frac{n^{[1]}}{2p^{[1]}} < \frac{1}{2}$. Therefore when $r^{[1]}$ is odd the length of the first run $r_0^{[1]} = \frac{r^{[1]}+1}{2}$ and when $r^{[1]}$ is even $r_0^{[1]} = \frac{r^{[1]}}{2}$. A similar property was used in the run length slice algorithm described by Bresenham [17]. To calculate the initial numerator value, we have from Theorem 3.2.5,

$$b_0^{[1]} = b + r_0^{[1]}p^{[1]} - p^{[0]} = r_0^{[1]}p^{[1]} - \frac{p^{[0]}}{2} \tag{5.2}$$

If $r^{[1]}$ is odd

$$\begin{aligned} b_0^{[1]} &= r_0^{[1]}p^{[1]} - \frac{p^{[0]}}{2} \\ &= \frac{(r^{[1]}+1)p^{[1]}}{2} - \frac{p^{[0]}}{2} \\ &= \frac{m^{[1]}}{2} \end{aligned} \tag{5.3}$$

If $r^{[1]}$ is even

$$\begin{aligned} b_0^{[1]} &= r_0^{[1]}p^{[1]} - \frac{p^{[0]}}{2} \\ &= \frac{(r^{[1]}p^{[1]}}{2} + p^{[1]} - \frac{p^{[0]}}{2} \\ &= p^{[1]} - \frac{n^{[1]}}{2} \end{aligned} \tag{5.4}$$

We should note here that if $p^{[1]} = b_0^{[0]}$ we have the special case that $r^{[1]} = 2$ and the maximum order is 2. This condition can be handled separately to improve the efficiency of the algorithm. We will describe this and other special cases later.

If the initial partial run length is not zero, the run is set at position (x_0, y_0). The next run will therefore be set at $(x, y) = (x_0 + r_0^{[1]}, y_0 + 1)$. Therefore we have all the necessary information to initialise the relevant variables and set the initial partial run of order 1 if it exists. The initialisation process is described by the following code segment.

{The slope $> \frac{1}{2}$}
if $p^{[1]} > b_0^{[0]}$
    {The possible run lengths}
    $r^{[1]} = 1$
    $r'^{[1]} = 2$

    {The initial partial run length and numerator value}
    $r_0^{[1]} = 0$
    $b_0^{[1]} = b_0^{[0]}$
    $x = x_0$
    $y = y_0$

    {The structural parameters $n^{[1]}$ and $m^{[1]}$}
    $n^{[1]} = p^{[0]} - p^{[1]}$
    $m^{[1]} = p^{[1]} - n^{[1]}$

{The slope $\leq \frac{1}{2}$}
else
    {The possible run lengths}
    $r^{[1]} = \lfloor \frac{p^{[0]}}{p^{[1]}} \rfloor$
    $r'^{[1]} = r^{[1]} + 1$

    {The initial partial run length and numerator value}
    if $r^{[1]}$ is odd
        $r_0^{[1]} = \frac{r'^{[1]}}{2}$
        $b_0^{[1]} = \frac{m^{[1]}}{2}$
    else
        $r_0^{[1]} = \frac{r^{[1]}}{2} + 1$
        $b_0^{[1]} = p^{[1]} - \frac{n^{[1]}}{2}$

    {Set the initial partial run length}
    $setRunOrder1(x_0, y_0, r_0^{[1]})$
    $x = x_0 + r_0^{[1]}$
    $y = y_0 + 1$

    {The structural parameters $n^{[1]}$ and $m^{[1]}$}
    $n^{[1]} = p^{[0]} - r^{[1]} p^{[1]}$
    $m^{[1]} = p^{[1]} - n^{[1]}$

Drawing the full order 1 runs

The main body of the algorithm is to draw the full runs of order 1 using the numerator sequence. As discussed in section 2.5, the values of the numerator sequence are offset by $-n^{[1]}$ so the internal decision of which possible run length can be made against zero. Termination of the loop occurs when the y-coordinate equals the value y_1, which corresponds to the y coordinate of the final partial run length.

{Numerator sequence for order 1}
$b_1^{[1]} = b_0^{[1]} - n^{[1]}$
while $y < y_1$
    if $b_j^{[1]} < 0$
        {Set a long run}
        $setRunOrder1(x, y, r'^{[1]})$
        $x = x + r'^{[1]}$
        $b_{j+1}^{[1]} = b_j^{[1]} + m^{[1]}$
    else
        {Set a short run}
        $setRunOrder1(x, y, r^{[1]})$
        $x = x + r^{[1]}$
        $b_{j+1}^{[1]} = b_j^{[1]} - n^{[1]}$
    $y = y + 1$

Termination

The only task for termination of the algorithm is to set the final run of order 1 of length $r_p^{[1]}$. When $\frac{a}{d} < \frac{1}{2}$, the final run length $r_p^{[1]} = r_0^{[1]}$. However for the case of $\frac{a}{d} \geq \frac{1}{2}$ we have $r^{[1]} = 1$, $r_0^{[1]} = 0$ and $r_p^{[1]} = 1$. Therefore a final run of order 1 must always exist.

{Set the final partial run}
if $p^{[1]} > b_0^{[0]}$
    $r_p^{[1]} = 1$
else
    $r_p^{[1]} = r_0^{[1]}$
$setRunOrder1(x, y, r_p^{[1]})$

We present the complete algorithm for drawing the line (x_0, y_0) to (x_1, y_1) using first order runs.

Algorithm 5.2 The Order 1 Line Drawing Algorithm

$\Delta x = x_1 - x_0$
$p^{[0]} = 2\Delta x$
$p^{[1]} = 2(y_1 - y_0)$

if $p^{[1]} > \Delta x$ {The slope $> \frac{1}{2}$}
  {The possible run lengths}
  $r^{[1]} = 1$
  $r'^{[1]} = 2$
  {The structural parameters $n^{[1]}$ and $m^{[1]}$}
  $n^{[1]} = p^{[0]} - p^{[1]}$
  $m^{[1]} = p^{[1]} - n^{[1]}$
  {The final partial run length and initial numerator value}
  $r_p^{[1]} = 1$
  $b_1^{[1]} = \Delta x - n^{[1]}$
  {Start point}
  $x = x_0$
  $y = y_0$

else {The slope $\leq \frac{1}{2}$}
  {The possible run lengths}
  $r^{[1]} = \lfloor \frac{p^{[0]}}{p^{[1]}} \rfloor$
  $r'^{[1]} = r^{[1]} + 1$
  {The structural parameters $n^{[1]}$ and $m^{[1]}$}
  $n^{[1]} = p^{[0]} - r^{[1]} p^{[1]}$
  $m^{[1]} = p^{[1]} - n^{[1]}$
  {The partial run lengths and initial numerator value}
  if $r^{[1]}$ is odd
    $r_p^{[1]} = r_0^{[1]} = \frac{r'^{[1]}}{2}$
    $b_1^{[1]} = \frac{m^{[1]}}{2} - n^{[1]}$
  else
    $r_p^{[1]} = r_0^{[1]} = \frac{r^{[1]}}{2} + 1$
    $b_1^{[1]} = m^{[1]} - \frac{n^{[1]}}{2}$
  {Set the initial partial run length}
  $setRunOrder1(x_0, y_0, r_0^{[1]})$
  {Start point}
  $x = x_0 + r_0^{[1]}$
  $y = y_0 + 1$

{Set the full runs of order 1}
while $y < y_1$

if $b^{[1]}_{j+1} < 0$
  {Set a long run}
  $setRunOrder1(x, y, r'^{[1]})$
  $x = x + r'^{[1]}$
  $y = y + 1$
  $b^{[1]}_{j+1} = b^{[1]}_j + m^{[1]}$
else
  {Set a short run}
  $setRunOrder1(x, y, r^{[1]})$
  $x = x + r^{[1]}$
  $y = y + 1$
  $b^{[1]}_{j+1} = b^{[1]}_j - n^{[1]}$

{Set the final partial run}
$setRunOrder1(x, y, r^{[1]}_p)$

5.2.3 Special Cases

When calculating the intersection pattern, a number of special cases exist which we can use to improve the efficiency of the algorithm. We will discuss two. Consider that we are using an order i based algorithm. The first special case we will discuss is when the maximum order $i_{max} = i$. In this case the line will be composed entirely of short runs of order i. Hence the numerator sequence does not have to be calculated and the runs are simply set with no further calculation required.

For the order 1 algorithm, if the maximum order of the intersection pattern is 1, the slope is equivalent to $\frac{g}{gd'}$. The length of each run is $r^{[1]} = d'$ and there are g of them. This case occurs when $n^{[1]} = 0$ and can only occur if the slope $< \frac{1}{2}$ or $p^{[1]} < b^{[0]}_0$. This special case is particularly important for line drawing as many of the lines drawn in graphics applications are reasonably short. Of the 45 lines $y = \frac{a}{d}x + \frac{1}{2}$ such that $0 < a < d$ and the length of the line is up to 10 pixels in length, i.e. $d \leq 10$, 17 lines have a maximum order of 1.

{Special Case: $n^{[1]} = 0$}
if $n^{[1]} = 0$
  while $y < y_1$
    $setRunOrder1(x, y, r^{[1]})$
    $x = x + r^{[1]}$
    $y = y + 1$

```
{Set the final partial run}
setRunOrder1(x, y, r_0^[1])
return
```

The second special case which can be utilised using an order i algorithm is when $n^{[i]} = m^{[i]}$. This means that the maximum order of the line is $i+1$ but more importantly that the line is composed of single occurrences of long and short runs of order i although the sequence has to be determined for the type of order i.

This second special case occurs for the order 1 algorithm when $n^{[1]} = m^{[1]}$. In this case, the line consists of a sequence of long and short runs of order 1. Utilising this special case the numerator sequence does not have to be calculated and the runs can be set. Again at order 1 this is important for short lines. Of the 45 lines of length 10 and under, a further 7 lines fit this case and can be set very efficiently.

```
{Special Case: n^[1] = m^[1]}
y_p = y_1 - 1
if n^[1] = m^[1]
    if r_0^[1] is odd
        while y < y_p
            setRunOrder1(x, y, r'^[1])
            x = x + r'^[1]
            y = y + 1
            setRunOrder1(x, y, r^[1])
            x = x + r^[1]
            y = y + 1
        setRunOrder1(x, y, r'^[1])
        x = x + r'^[1]
        y = y + 1
    else
        while y < y_p
            setRunOrder1(x, y, r^[1])
            x = x + r^[1]
            y = y + 1
            setRunOrder1(x, y, r'^[1])
            x = x + r'^[1]
            y = y + 1
        setRunOrder1(x, y, r^[1])
        x = x + r^[1]
        y = y + 1
```

```
{Set the final partial run}
setRunOrder1(x, y, r_p^[1])
return
```

5.2.4 The Order 2 Algorithm

The order 2 algorithm is similar to the order 1 algorithm with the added complexity of type. Depending on the type of the line, two shapes of the order 2 run exist, ls^+ (shape 1) and l^+s (shape 3). Therefore the algorithm is divided into two sections based on the value of the type parameter and we will be using two primitives $setRunOrder2Shape1(x, y, r_j^{[2]})$ to set a run with shape 1 (ls^+) and $setRunOrder2Shape3(x, y, r_j^{[2]})$ to set a run with shape 3 (l^+s). We will also use the primitive $setRunsOrder1(x, y, g, r)$ to set g runs of length r.

Initialisation

The first task is to set the first partial run of order 1. This follows the procedure described in the order 1 algorithm. We also define the possible run lengths of order 2.

{Calculate the possible run lengths}
$r^{[2]} = \lfloor \frac{p^{[1]}}{p^{[2]}} \rfloor$
$r'^{[2]} = r^{[2]} + 1$

Type 0: Initialisation

Consider the calculation of the initial partial run length of order 2, $r_0^{[2]}$, and the initial numerator value of order 2, $b_0^{[2]}$. The initial numerator value of order 2 is calculated by finding the first order 1 numerator value that is less than $p^{[2]}$. Therefore if $b_0^{[1]} < p^{[2]}$, $r_0^{[2]} = 0$ and $b_0^{[2]} = b_0^{[1]}$.

If $b_0^{[1]} \geq p^{[2]}$ then by Theorems 3.2.3, 3.2.4 and 3.2.6

$$r_0^{[2]} = \left\lfloor \frac{b_0^{[1]}}{p^{[2]}} \right\rfloor \tag{5.5}$$

$$b_0^{[2]} = b_0^{[1]} - r_0^{[2]} p^{[2]} \tag{5.6}$$

The shape of an order 2 type 0 run is ls^+ therefore, if the initial partial run length is not zero, the run will consist of $r_0^{[2]}$ short runs of order 1. The next run will therefore be set at $(x, y) = (x + r_0^{[2]} r^{[1]}, y + r_0^{[2]})$.

```
{Calculate the structural parameters}
p^[2] = n^[1]
n^[2] = p^[1] - r^[2] p^[2]
m^[2] = p^[2] - n^[2]

{Calculate the initial numerator value
 and the initial partial run}
if b_0^[1] < p^[2]
    b_0^[2] = b_0^[1]
else
    r_0^[2] = b_0^[1] / p^[2]
    b_0^[2] = b_0^[1] - r_0^[2] p^[2]
    {Set the initial partial run}
    setRunOrder2Shape1(x, y, r_0^[2])
    x = x + r_0^[2] r^[1]
    y = y + r_0^[2]
```

Type 0: Intermediate Runs

The run lengths of the full runs of order 2 are calculated using the numerator sequence of order 2. As discussed in section 2.5, the values of the numerator sequence are offset by $-n^{[2]}$ so the internal decision of run length can be made against zero.

We know that there are $p^{[2]}/2$ runs of order 2 in the line. One of them may be truncated. Therefore there are $p^{[2]}/2 - 1$ full runs of order 2.

For type 0 the shape of the run is ls^+ therefore the position of the next order 2 run can be calculated by $(x, y) = (x + r_j^{[2]} r^{[1]} + 1, y + r_j^{[2]})$.

```
    {Numerator sequence for order 2}
    runs = p^[2]/2 - 1
    b_1^[2] = b_0^[2] - n^[2]
```

```
while runs > 0
    if b^{[2]}_{j+1} < 0
        {Set a short run}
        setRunOrder2Shape1(x, y, r^{[2]})
        x = x + r^{[2]} r^{[1]} + 1
        y = y + r^{[2]}
        b^{[2]}_{j+1} = b^{[2]}_j + m^{[2]}
    else
        {Set a long run}
        setRunOrder2Shape1(x, y, r'^{[2]})
        x = x + r'^{[2]} r^{[1]} + 1
        y = y + r'^{[2]}
        b^{[2]}_{j+1} = b^{[2]}_j - n^{[2]}
    runs = runs - 1
```

Type 0: Termination

The length of the final partial run of order 2, $r_p^{[2]}$, can be calculated by the y-coordinates at the end of the last full run of order 2, y, and the end of the line, y_1. Taking into consideration that the last partial run of order 1 must also be set at the end of the line, $r_p^{[2]} = y_1 - y$.

The shape of an order 2, type 0 run is ls^+, therefore we can set a second order run of shape 1 and length $r_p^{[2]}$.

```
{Set the final partial runs of order 2 & 1}
if r^{[2]}_0 ≠ 0 setRunOrder2Shape1(x, y, y_1 - y)
setRunOrder1(x_1 - r^{[1]}_p + 1, y_1, r^{[1]}_p)
```

Type 1: Initialisation

The initial partial run length, $r_0^{[2]}$, and the initial numerator value, $b_0^{[2]}$ can be calculated by Theorems 3.2.3, 3.2.4 and 3.2.6 if $b_0^{[1]} \geq p^{[2]}$.

$$r_0^{[2]} = \left\lfloor \frac{p^{[1]} - b_0^{[1]}}{p^{[2]}} \right\rfloor + 1 \tag{5.7}$$

$$b_0^{[2]} = b_0^{[1]} + r_0^{[2]} p^{[2]} - p^{[1]} \tag{5.8}$$

If $b_0^{[1]} < p^{[2]}$, $r_0^{[2]} = 0$ and $b_0^{[2]} = b_0[1]$.

The shape of an order 2 type 1 run is l^+s therefore if the initial partial run length is not zero, the run will consist of $r_0^{[2]} - 1$ long runs and one short run of order 1. The next run will therefore be set at $(x, y) = (x + r_0^{[1]}(r^{[1]} + 1) - 1, y + r_0^{[1]})$.

{Calculate the structural parameters}
$p^{[2]} = m^{[1]}$
$n^{[2]} = p^{[1]} - r^{[2]}p^{[2]}$
$m^{[2]} = p^{[2]} - n^{[2]}$

{Calculate the initial numerator value
and the initial partial run}
if $b_0^{[1]} < p^{[2]}$
    $b_0^{[2]} = b_0^{[1]}$
else
    $r_0^{[2]} = \frac{p^{[1]} - b_0^{[1]}}{p^{[2]}}$
    $b_0^{[2]} = r_0^{[2]}p^{[2]} - b_0^{[1]}$
    $setRunOrder2Shape3(x, y, r_0^{[2]})$
    $x = x + r_0^{[2]}r'^{[1]} - 1$
    $y = y + r_0^{[2]}$

Type 1: Intermediate Runs

Calculation of the intermediate runs of type 1 follows the example of type 0 except that the shape of the run is l^+s. Therefore the start of the next order 2 run is $(x, y) = (x + r_j^{[2]}r'^{[1]} - 1, y + r_j^{[2]})$.

{Numerator sequence for order 2}
$b_1^{[2]} = b_0^{[2]} - n^{[2]}$
$runs = \frac{p^{[2]}}{2} - 1$
while $runs > 0$
    if $b_{j+1}^{[2]} < 0$
        {Set a long run}
        $setRunOrder2Shape1(x, y, r'^{[2]})$
        $x = x + r'^{[2]}r'^{[1]} - 1$
        $y = y + r'^{[2]}$
        $b_{j+1}^{[2]} = b_j^{[2]} + m^{[2]}$
    else
        {Set a short run}
        $setRunOrder2Shape1(x, y, r^{[2]})$
        $x = x + r^{[2]}r'^{[1]} - 1$

$y = y + r^{[2]}$
$b^{[2]}_{j+1} = b^{[2]}_j - n^{[2]}$
$runs = runs - 1$

Type 1: Termination

The shape of an order 2, type 1 run is $l^{+}s$, therefore we must set $r^{[2]}_p = y_1 - y$ long order 1 runs.

$\{$Set the final partial runs of order 2 & 1$\}$
if $r^{[2]}_0 \neq 0$ $setRunsOrder1(x, y_1 - y, r^{[2]}_p)$
$setRunOrder1(x_1 - r^{[1]}_p + 1, y_1, r^{[1]}_p)$

The Complete Algorithm

We present the full algorithm for drawing the line (x_0, y_0) to (x_1, y_1) using second order runs.

Algorithm 5.3 The order 2 line drawing algorithm

$\Delta x = x_1 - x_0$
$p^{[0]} = 2\Delta x$
$p^{[1]} = 2(y_1 - y_0)$

$\{$The slope $> \frac{1}{2}\}$
if $p^{[1]} > \Delta x$
  $\{$The possible run lengths$\}$
  $r^{[1]} = 1$
  $r'^{[1]} = 2$
  $\{$The structural parameters $n^{[1]}$ and $m^{[1]}\}$
  $n^{[1]} = p^{[0]} - p^{[1]}$
  $m^{[1]} = p^{[1]} - n^{[1]}$
  $\{$The initial partial run length and numerator value$\}$
  $r^{[1]}_p = 1$
  $b^{[1]}_0 = \Delta x$
  $\{$Start point$\}$
  $x = x_0$
  $y = y_0$

$\{$The slope $\leq \frac{1}{2}\}$

else
{The possible run lengths}
$r^{[1]} = \lfloor \frac{p^{[0]}}{p^{[1]}} \rfloor$
$r'^{[1]} = r^{[1]} + 1$
{The structural parameters $n^{[1]}$ and $m^{[1]}$}
$n^{[1]} = p^{[0]} - r^{[1]} p^{[1]}$
$m^{[1]} = p^{[1]} - n^{[1]}$
{The initial partial run length and numerator value}
if $r^{[1]}$ is odd
$r_p^{[1]} = r_0^{[1]} = \frac{r'^{[1]}}{2}$
$b_0^{[1]} = \frac{m^{[1]}}{2}$
else
$r_p^{[1]} = r_0^{[1]} = \frac{r^{[1]}}{2} + 1$
$b_1^{[1]} = p^{[1]} - \frac{n^{[1]}}{2}$
{Set the initial partial run length}
$setRunOrder1(x_0, y_0, r_0^{[1]})$
{Start point}
$x = x_0 + r_0^{[1]}$
$y = y_0 + 1$

{Type 0, $p^{[2]} = n^{[1]}$}
if $n^{[1]} < m^{[1]}$
{The possible run lengths}
$r^{[2]} = \left\lfloor \frac{p^{[1]}}{n^{[1]}} \right\rfloor$
$r'^{[2]} = r^{[2]} + 1$
{The structural parameters}
$n^{[2]} = p^{[1]} - r^{[2]} n^{[1]}$
$m^{[2]} = n^{[1]} - n^{[2]}$
{The initial numerator value and partial run length}
if $b_0^{[1]} < n^{[1]}$
$b_1^{[2]} = b_0^{[1]} - n^{[2]}$
else
$r_0^{[2]} = \left\lfloor \frac{b_0^{[1]}}{n^{[1]}} \right\rfloor$
$b_1^{[2]} = b_0^{[1]} - r_0^{[2]} n^{[1]} - n^{[2]}$
{Set the initial partial run}
$setRunOrder2Shape1(x, y, r_0^{[2]})$
$x = x + r_0^{[2]} r^{[1]}$
$y = y + r_0^{[2]}$
{The possible increments for the x coordinate}
$\delta x_s = r^{[2]} r^{[1]} + 1$
$\delta x_l = \delta x_s + r^{[1]}$

{Numerator sequence for order 2}
$runs = \left\lfloor \frac{n^{[1]}}{2} \right\rfloor$
while $(runs = runs - 1) > 0$

if $b^{[2]}_{j+1} < 0$
{Set a short run}
$setRunOrder2Shape1(x, y, r^{[2]})$
$x = x + \delta x_s$
$y = y + r^{[2]}$
$b^{[2]}_{j+1} = b^{[2]}_j + m^{[2]}$
else
{Set a long run}
$setRunOrder2Shape1(x, y, r'^{[2]})$
$x = x + \delta x_l$
$y = y + r'^{[2]}$
$b^{[2]}_{j+1} = b^{[2]}_j - n^{[2]}$

{Set the final partial runs of order 2 & 1}
if $y < y_1$ $setRunOrder2Shape1(x, y, y_1 - y)$
$setRunOrder1(x_1 - r^{[1]}_p + 1, y_1, r^{[1]}_p)$

{Type 1, $p^{[2]} = m^{[1]}$}
else
{The possible run lengths}
$r^{[2]} = \left\lfloor \frac{p^{[1]}}{m^{[1]}} \right\rfloor$
$r'^{[2]} = r^{[2]} + 1$
{The structural parameters}
$n^{[2]} = p^{[1]} - r^{[2]} m^{[1]}$
$m^{[2]} = m^{[1]} - n^{[2]}$
{The initial numerator value and partial run}
if $b^{[1]}_0 < m^{[1]}$
$b^{[2]}_1 = b^{[1]}_0 - n^{[2]}$
else
$pb = p^{[1]} - b^{[1]}_0$
$r^{[2]}_0 = \left\lfloor \frac{pb}{m^{[1]}} \right\rfloor$
$b^{[2]}_1 = r^{[2]}_0 m^{[1]} - pb - n^{[2]}$
$setRunOrder2Shape3(x, y, r^{[2]}_0)$
$x = x + r^{[2]}_0 r'^{[1]} - 1$
$y = y + r^{[2]}_0$
{The possible increments for the x coordinate}
$\delta x_s = r^{[2]} r^{[1]} + 1$
$\delta x_l = \delta x_s + r^{[1]}$

{Numerator sequence for order 2}
$runs = \left\lfloor \frac{m^{[1]}}{2} \right\rfloor$
while $(runs = runs - 1) > 0$
if $b^{[2]}_{j+1} < 0$
{Set a long run}
$setRunOrder2Shape1(x, y, r'^{[2]})$
$x = x + \delta x_l$

$y = y + r'^{[2]}$
$b_{j+1}^{[2]} = b_j^{[2]} + m^{[2]}$
else
{Set a short run}
$setRunOrder2Shape1(x, y, r^{[2]})$
$x = x + \delta x_s$
$y = y + r^{[2]}$
$b_{j+1}^{[2]} = b_j^{[2]} - n^{[2]}$

{Set the final partial runs of order 2 & 1}
if $y < y1$ $setRunsOrder1(x, y_1 - y, r_p^{[2]})$
$setRunOrder1(x_1 - r_p^{[1]} + 1, y_1, r_p^{[1]})$

5.3 The Recursive Run–Based Line Drawing Algorithm

To define a general algorithm which uses all of the orders of runs within the intersection pattern to draw the line, we will describe a recursively based algorithm for higher order runs which is called from a parent algorithm based on order 1 runs. Let i_{max} be the maximum order of the intersection pattern. A high level description of the recursive algorithm for higher order runs based on type is

$order(p^{[i-1]}, m^{[i-1]}, n^{[i-1]}, t^{[i-1]}, b_0^{[i-1]})$
if type 0
{Calculate the run parameters for order i, type 0}

{Draw initial partial run of order i}

if $i < i_{max}$
$order(p^{[i]}, m^{[i]}, n^{[i]}, t^{[i]}, b_0^{[i]})$
else
{Draw $\gcd(a, d) - 1$ occurrences of the maximum order run}

{Draw final partial run of order i}
else type 1
{Calculate the run parameters for order i, type 1}

{Draw initial partial run of order i}

if $i < i_{max}$

$order(p^{[i]}, m^{[i]}, n^{[i]}, t^{[i]}, b_0^{[i]})$
else
{Draw $\gcd(a, d) - 1$ occurrences of the maximum order run}

{Draw final partial run of order i}

Hence, the run parameters for order i are calculated. The initial partial run of order i is set. If the maximum order is met, $i = i_{max}$, the run of maximum order is drawn $\gcd(\Delta x, \Delta y) - 1$ times. If the maximum order is not met, the algorithm recurses. Lastly, the final partial run of order i is set and the algorithm completes.

Termination of the recursion occurs when $i = i_{max}$. For type 0 higher order runs, this occurs when $n^{[i]} = 0$. For type 1, when $m^{[i]} = 0$. The $\gcd(\Delta x, \Delta y) = \frac{p^{[i_{max}]}}{2}$.

The recursive section of the algorithm is called from an algorithm which sets the order 1 runs.

$line(x_0, y_0, x_1, y_1)$
{Calculate the run parameters for order 1}

{Draw initial partial run of order 1}

if $i < i_{max}$
$order(p^{[1]}, m^{[1]}, n^{[1]}, t^{[1]}, b_0^{[1]})$

{Draw final partial run of order 1}

The calculation of the first order run parameters follows the discussion of the order 1 algorithm, as does setting the initial and final partial runs of order 1. The special case of $i_{max} = 1$ is included to stop unnecessary recursion.

To calculate the run parameters for order $i > 1$ type 0 we have

if $n^{[i-1]} < m^{[i-1]}$
$t^{[i]} = 0$
$s^{[i]} = t^{[i-1]} \oplus t^{[i]}$
$p^{[i]} = n^{[i-1]}$
$r^{[i]} = \left\lfloor \frac{p^{[i-1]}}{p^{[i]}} \right\rfloor$
$m^{[i]} = p^{[i-1]} - p^{[i]} r^{[i]}$
$n^{[i]} = p^{[i]} - m^{[i]}$

if $b_0^{[i-1]} < p^{[i]}$
    $r_0^{[i]} = 0$
    $b_0^{[i]} = b_0^{[i-1]}$
else
    $r_0^{[i]} = \left\lfloor \frac{b_0^{[i-1]}}{p^{[i]}} \right\rfloor$
    $b_0^{[i]} = b_0^{[i-1]} - r_0^{[i]} p^{[i]}$

$p^{[i+1]} = \min(n^{[i]}, m^{[i]})$
if $b_0^{[i]} < p^{[i+1]}$
    $r_p^{[i]} = r^{[i]} - r_0^{[i]}$
else
    $r_p^{[i]} = r^{[i]} - r_0^{[i]} - 1$

For order $i > 1$ type 1 we have

if $m^{[i-1]} < n^{[i-1]}$
    $t^{[i]} = 1$
    $s^{[i]} = t^{[i-1]} \oplus t^{[i]}$
    $p^{[i]} = m^{[i-1]}$
    $r^{[i]} = \left\lfloor \frac{p^{[i-1]}}{p^{[i]}} \right\rfloor$
    $n^{[i]} = p^{[i-1]} - p^{[i]} r^{[i]}$
    $m^{[i]} = p^{[i]} - n^{[i]}$

    if $b_0^{[i-1]} < p^{[i]}$
        $r_0^{[i]} = 0$
        $b_0^{[i]} = b_0^{[i-1]}$
    else
        $r_0^{[i]} = \left\lfloor \frac{p^{[i-1]} - b_0^{[i-1]}}{p^{[i]}} \right\rfloor + 1$
        $b_0^{[i]} = b_0^{[i-1]} + r_0^{[i]} p^{[i]} - p^{[i-1]}$

    $p^{[i+1]} = \min(n^{[i]}, m^{[i]})$
    if $b_0^{[i]} > p^{[i+1]}$
        $r_p^{[i]} = r^{[i]} - r_0^{[i]}$
    else
        $r_p^{[i]} = r^{[i]} - r_0^{[i]} - 1$

To set the initial partial run of order $i > 1$, the shape of the run depends on the type of order $i-1$ as well as the type of order i. To draw the initial partial run of order i, type 0, we have two possibilities. If $t^{[i-1]} = 0$ the shape is sl^+, therefore the initial partial run will consist of $r_0^{[i]}$ long runs of order $i-1$. If $t^{[i-1]} = 1$ the shape is s^+l, and an order i run of shape s^+l and length $r_0^{[i]}$ can be drawn.

{Draw initial partial run of order i}
if $t^{[i-1]} = 0$
{Shape 0 (sl^+): Set $r_0^{[i]}$ long runs of order $i-1$}
for $k = 0$ **to** $k = r_0^{[i]}$
$setShape(i-1, x, y, r^{[i-1]} + 1)$
else
$setShape(i, x, y, r_0^{[i]})$

Similarly for order i, type 1

{Draw initial partial run of order i}
if $t^{[i-1]} = 0$
{Shape 1 (ls^+): Set $r_0^{[i]}$ short runs of order $i-1$}
for $k = 0$ **to** $k = r_0^{[i]}$
$setShape(i-1, x, y, r^{[i-1]})$
else
$setShape(i, x, y, r_0^{[i]})$

The algorithm to draw the final partial run of order $i > 1$, is developed in a similar fashion to setting the initial partial run length. For type 0,

{Draw final partial run of order i}
if $t^{[i-1]} = 1$
{Shape 2 (s^+l): Set $r_p^{[i]}$ short order $i-1$ runs}
for $k = 0$ **to** $k = r_p^{[i]}$
$setShape(i-1, x, y, r^{[i-1]})$
else
$setShape(i, x, y, r_p^{[i]})$

For type 1,

{Draw final partial run of order i}
if $t^{[i-1]} = 1$
{Shape 3 (l^+s): Set $r_p^{[i]}$ long order $i-1$ runs}
for $k = 0$ **to** $k = r_p^{[i]}$

$setShape(i-1, x, y, r^{[i-1]}+1)$
else
$setShape(i, x, y, r_p^{[i]})$

The primitive to draw a run of order i and shape $s^{[i]}$ will recursively draw the runs of order $i-1$ which comprise the run until order 1 is reached. In this case the actual run is set. The positional variables x and y are altered by the *setShape* primitive.

$setShape(i, x, y, r)$
{If the order $i = 2$, set the run}
if $i = 1$
$setRunOrder1(x, y, r)$
$x = x + r$
$y = y + 1$
return

{If the order $i > 2$, set the r runs of order $i - 1$}
case $s^{[i]} = 0$ {Shape sl^+}
{Set 1 short order $i-1$ run, followed by $r-1$ long runs}
$setShape(i-1, x, y, r^{[i-1]})$
for $k = 0$**to** $k = r$
$setShape(i-1, x, y, r^{[i-1]}+1)$
case $s^{[i]} = 1$ {Shape ls^+}
{Set 1 long order $i-1$ run, followed by $r-1$ short runs}
$setShape(i, x, y, r^{[i-1]}+1)$
for $k = 0$**to** $k = r$
$setShape(i-1, x, y, r^{[i-1]})$
case $s^{[i]} = 2$ {Shape s^+l}
{Set $r-1$ short order $i-1$ run, followed by 1 long runs}
for $k = 0$**to** $k = r$
$setShape(i-1, x, y, r^{[i-1]})$
$setShape(i, x, y, r^{[i-1]}+1)$
case $s^{[i]} = 3$ {Shape l^+s}
{Set $r-1$ long order $i-1$ run, followed by 1 short runs}
for $k = 0$**to** $k = r$
$setShape(i-1, x, y, r^{[i-1]}+1)$
$setShape(i, x, y, r^{[i-1]})$

5.4 Experimental Results And Analysis

To consider the time complexity of the line drawing algorithms of order 0, 1 and 2, we will present experimental and theoretical results which compare these algorithms with the standard algorithms presented by Bresenham. The algorithms we will discuss are

- Bresenham's original algorithm,
- Bresenham's algorithm incorporating the double step and symmetry acceleration techniques described by Rokne [109, 143],
- Bresenham's run length slice algorithm,
- an adapted run length slice algorithm which uses the primitives based on order 1 and 2 runs,
- the order 0 run–based algorithm,
- the order 1 run–based algorithm,
- the order 2 run–based algorithm,
- the recursively defined run–based algorithm.

The experimental section of this work involved timing software–based implementations of the line drawing algorithms. The data set used was the set of lines $(0, 0)$ to (d, a) where $a = 1 \ldots d$ for each value of $d = 2 \ldots f$ within the frame $\mathcal{L}_{(f,f)}$. The frame size used was 256 pixels. The algorithms implemented were

- Bresenham's original algorithm,
- Bresenham's accelerated algorithm,
- Bresenham's run length slice algorithm,
- the adapted run length slice algorithm,
- the order 1 run–based algorithm,
- the order 2 run–based algorithm.

The order 0 algorithm and recursively–based algorithm were not included in the experimental analysis. The order 0 algorithm was not included because it is identical to Bresenham's original line drawing algorithm as we have shown in Appendix B. The recursively–based algorithm, as it is currently implemented, is not efficient over the size of the frames used in the experiment. However, it does provide an important result which will be discussed in the theoretical analysis.

The experiments were conducted using a Sun SparcStation 20 running the Ultrix operating system. The algorithms were implemented in C++ and each memory primitive was inlined. To gain a measurable cost in cpu seconds, each algorithm was run 10,000 times within a loop and the Unix system call `times()` was used to measure the total execution time of the loop. The programs were compiled using the Sun CC compiler with no optimisation flags set. With optimisation turned on, an experiment rerun several times without any change in the parameters of the experiment provided starkly varying results. We assume that because the machine used to run the experiment operates under a time sharing environment, the different state of the resources of the machine used for optimising the performance of a program caused the noted variations each time an experiment was run.

We will present the results of two experiments. The first involves only the scan conversion results of each algorithm. The pixels which constitute the digital line are calculated but not set in memory. Therefore the calls to the memory primitives were removed from the code. The second experiment involved the full implementation of the algorithms. For this experiment, the memory primitives `setPixel`, `setRunOrder1`, `setRunsOrder1`, `setRunOrder2Shape1`, and `setRunOrder2Shape3` were designed to have a cost that was comparable and reasonably small. To provide such an environment, a pixel was represented by a byte and the Unix facility `memset` used to set pixels and runs. No equivalent facility exists to set runs of order $i > 2$, therefore repeated calls to `memset` were used yielding a slower but comparable primitive. For both of these experiments, the results are presented based on the slope and the length of the line.

- Figure 5.3 presents the results of the scan conversion experiment sorted by slope.

- Figure 5.4 presents the results of the scan conversion experiment sorted by length.
- Figure 5.5 presents the results of the full implementation experiment sorted by slope.
- Figure 5.6 presents the results of the full implementation experiment sorted by length.

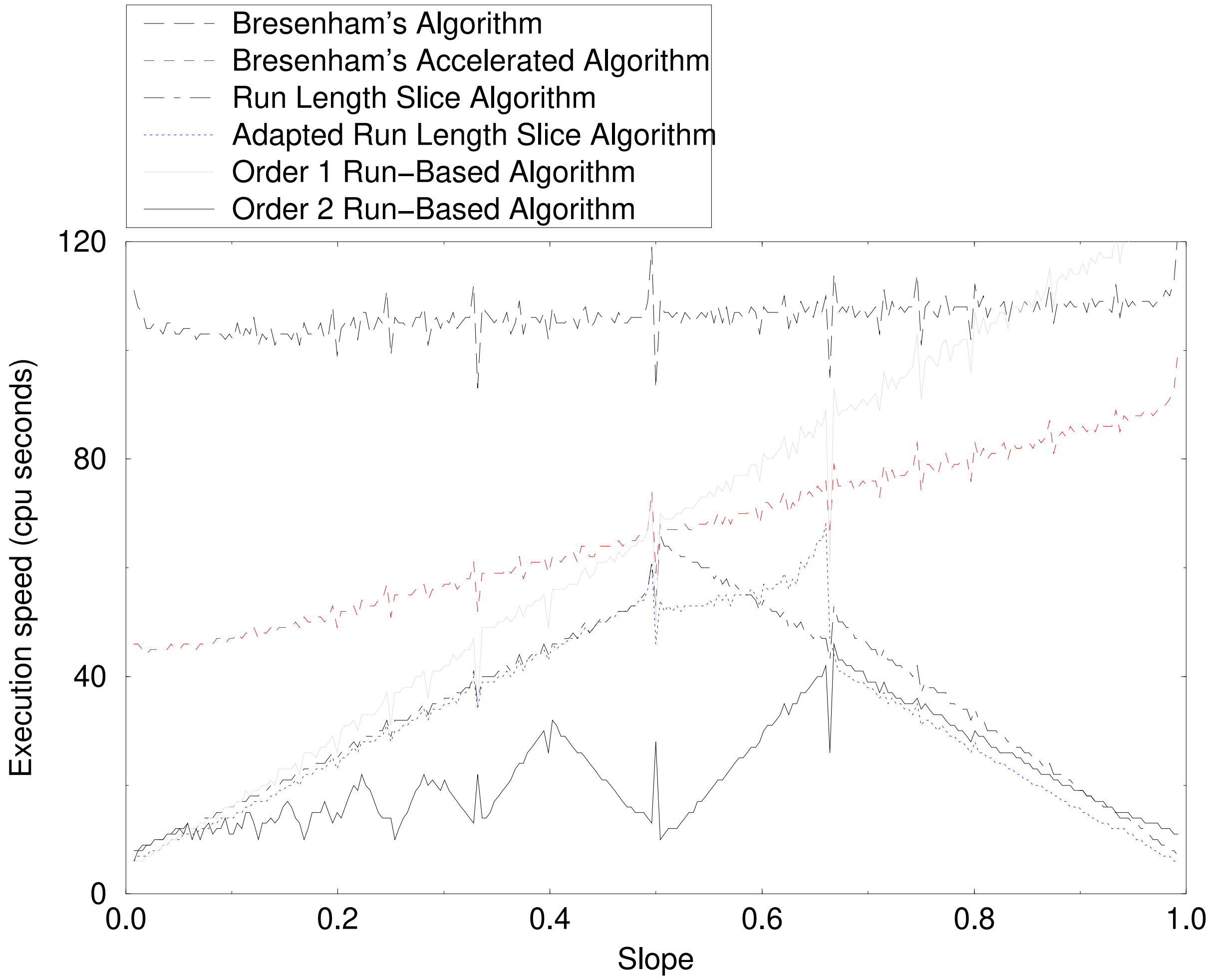


Figure 5.3: The dependence of execution speed of the scan conversion of the line drawing algorithms on the slope of the line.

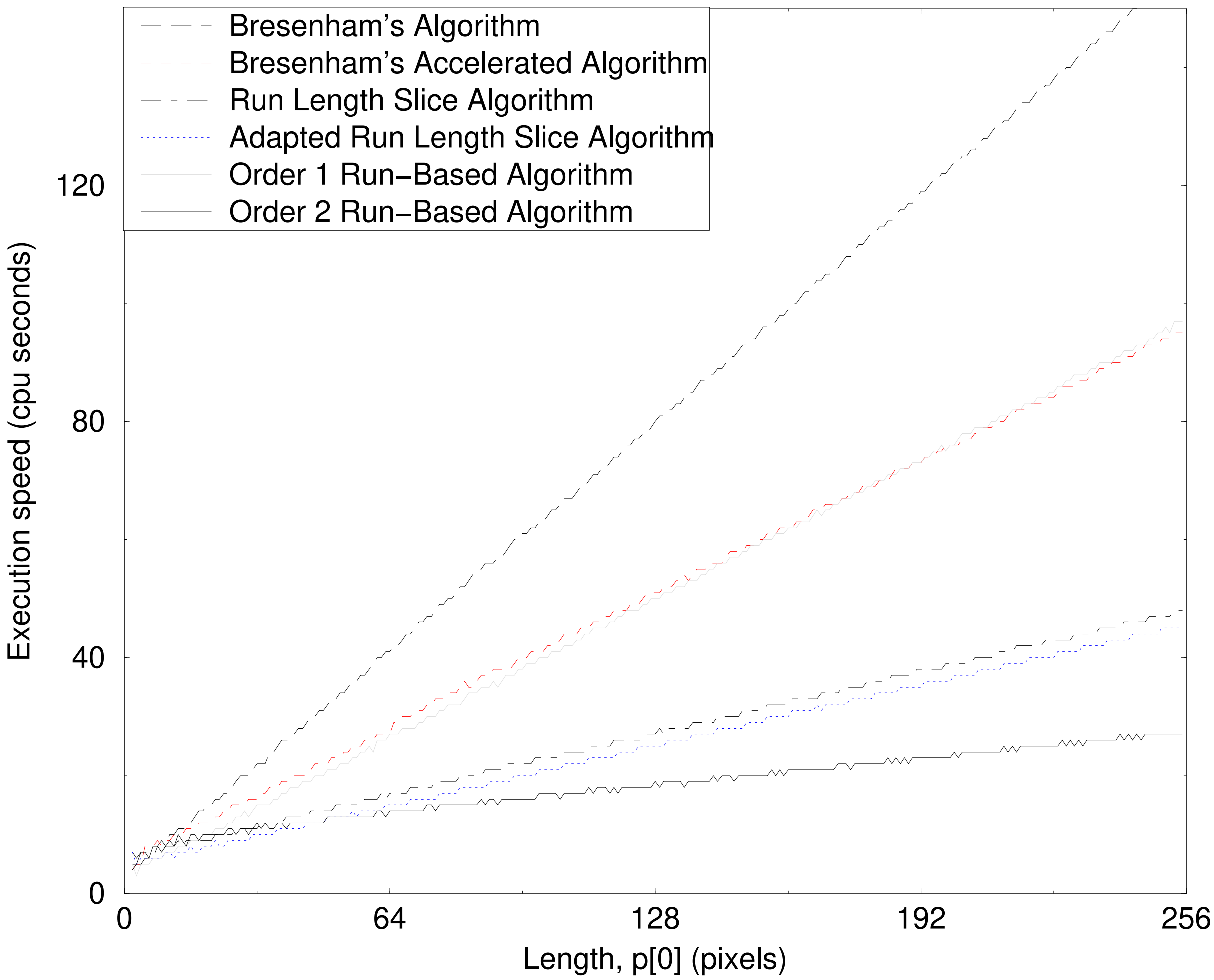


Figure 5.4: The dependence of execution speed of the scan conversion of the line drawing algorithms on the length of the line.

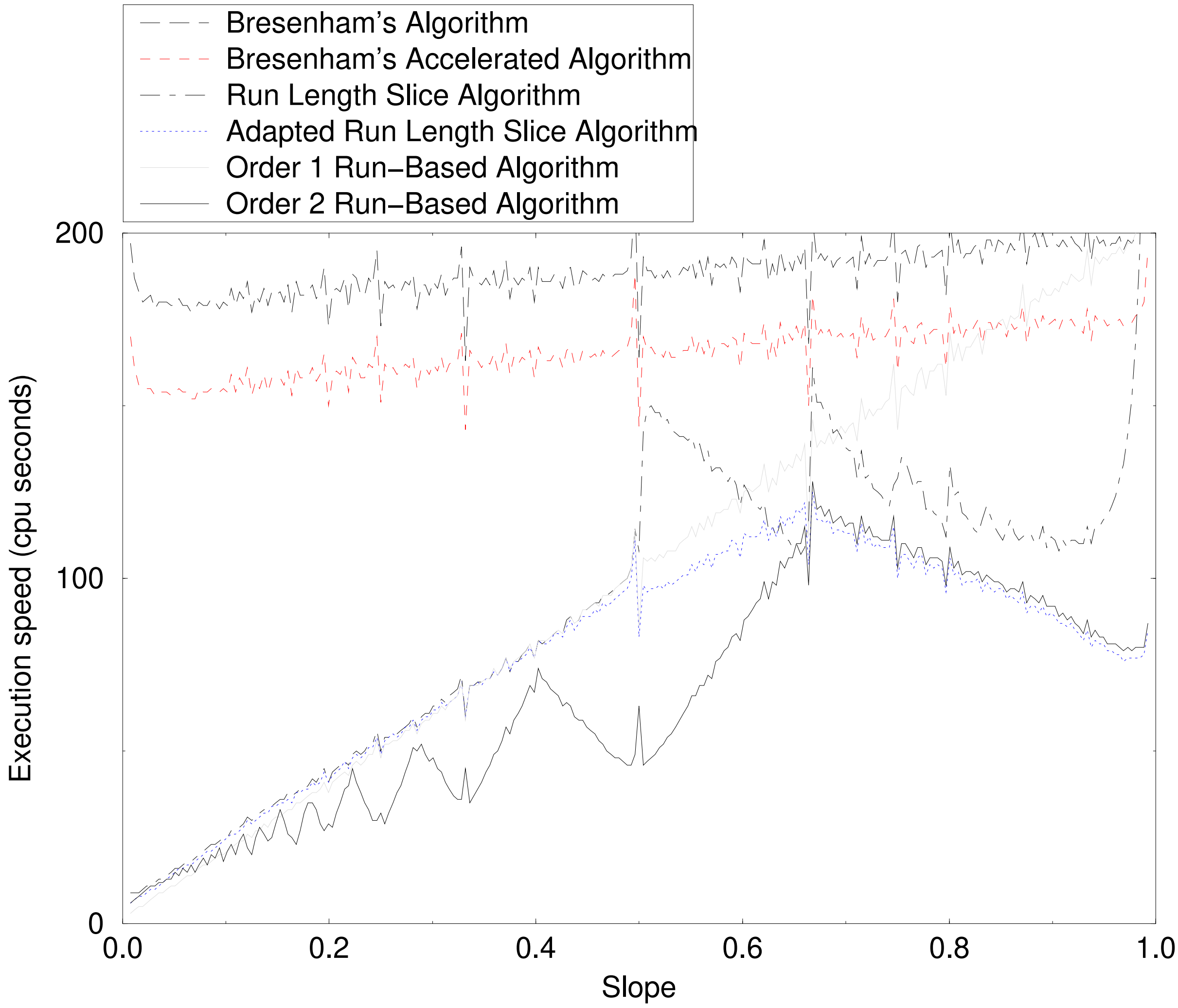


Figure 5.5: The dependence of execution speed of the line drawing algorithms on the slope of the line.

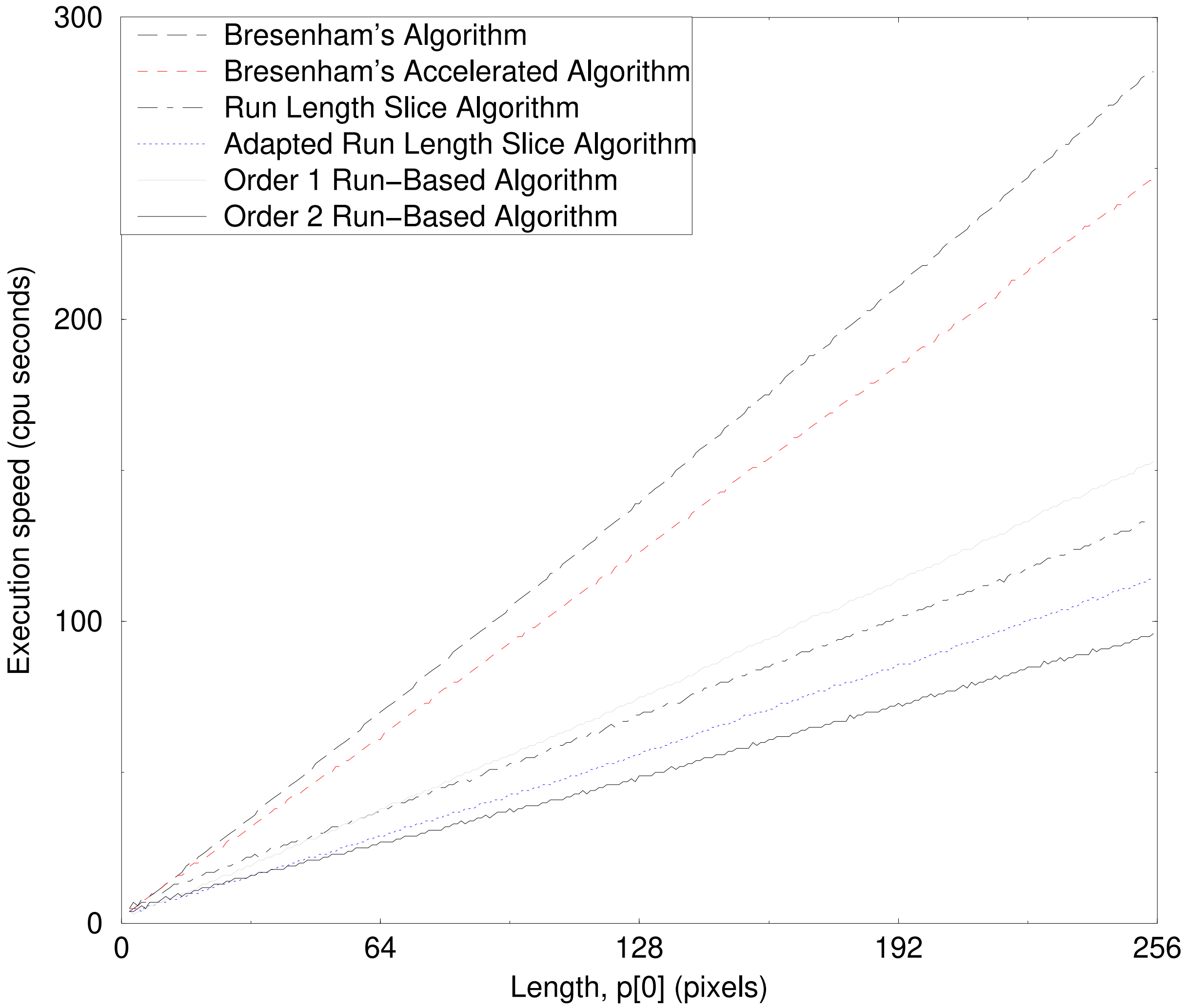


Figure 5.6: The dependence of execution speed of the line drawing algorithms on the length of the line.

To analyse the algorithms and the experimental results we have presented, we will conduct a complexity analysis and a count of the arithmetic operations used in the algorithms. For the complexity analysis we will make the assumption that the cost of the memory operations is $O(1)$. We will show that the complexity of the algorithms of order $i = 0, 1, 2$ is $O(p^{[i]})$.

To analyse the operator counts, we will not consider any special cases and we make the assumption that the following groups of operations incur similar costs.

- Additive integer arithmetic: $(+, -)$.
- Comparison operations: $(<, >, \leq, \geq)$.
- Shift operations: $(\lhd, \rhd)$.

The shift operators are included as all division and multiplication of integers by powers of two were implemented using a shift operation. To check if a number is odd the bitwise "and" operator was used to check the value of 1 against the least significant bit of the number.

5.4.1 The Order 0 Algorithm

As we discuss in Appendix B, the order 0 algorithm is identical to Bresenham's original line drawing algorithm. Bresenham's algorithm has undergone extensive analysis over the years so we shall simply provide the result of the complexity analysis of this algorithm.

Theorem 5.4.1 *Where the* `setPixel` *operation has* $O(1)$ *time cost, the order 0 run–based line drawing algorithm has time complexity* $O(p^{[0]})$.

From Figures 5.3 and 5.5, as would be expected, Bresenham's line algorithm or the order 0 algorithm, with or without the acceleration techniques applied, produced results which were independent of the slope and depend only on the length of the line. This is also apparent from the operator counts.

Table 5.1: Operator counts for the order 0 run–based algorithm.

Operator	Count
Memory	$p^{[0]}$
Assignment	$2p^{[0]} + 6$
Additive	$2p^{[0]} + 4$
Comparison	$2p^{[0]}$
Shift	2

5.4.2 The Order 1 Algorithm

From the experimental results, the order 1 algorithm depends linearly on both the slope and the length of the line. The reason for the dependence of the algorithm on the slope of the line is the dependence on the value on $p^{[1]}$, the number of runs of order 1 in the intersection pattern of the line.

Theorem 5.4.2 *Where the* `setRunOrder1` *operation has* $O(1)$ *time cost, the order 1 run–based line drawing algorithm has time complexity* $O(p^{[1]}) = O(\frac{n}{r^{[1]}})$.

Proof : By definition there are $\frac{p^{[1]}}{2}$ runs of order 1 in the intersection pattern of the line. Each run is drawn within a while loop that loops $\frac{p^{[1]}}{2}$ or $\frac{p^{[1]}}{2} - 1$ times. The time complexity of the algorithm is therefore $O(p^{[1]})$. There are $n^{[1]}$ runs of length $r^{[1]} + 1$ in the intersection pattern and $m^{[1]}$ runs of length $r^{[1]}$. Therefore $n = m^{[1]}r^{[1]} + n^{[1]}(r^{[1]} + 1)$. We have $p^{[1]} = m^{[1]} + n^{[1]}$. Hence $O(p^{[1]}) = O(\frac{n}{r^{[1]}})$. $\square$

The summary of operator costs supports Theorem 5.4.1.

Table 5.2: Operator counts for the order 1 run–based algorithm.

Operator	Count
Memory	$p^{[1]} + 0.5$
Assignment	$3p^{[1]} + 8.5$
Additive	$3p^{[1]} + 4$
Comparison	$2p^{[1]} - 0.5$
Division	0.5
Multiplication	0.5
Shift	3
Bitwise logical	0.5

5.4.3 The Order 2 Algorithm

The order 2 run based algorithm displays a more complicated behaviour than the linear behaviours of the orders 0 and 1 algorithms.

Theorem 5.4.3 *Where the* `setRunOrder2` *operation has $O(1)$ time cost, the order 2 run–based line drawing algorithm has time complexity $O(p^{[2]}) = O(\frac{n}{r^{[1]}r^{[2]}})$.*

Proof : By definition there are $\frac{p^{[2]}}{2}$ runs of order 2 in the intersection pattern of the line. Each run is drawn within a while loop that loops $\frac{p^{[2]}}{2}$ or $\frac{p^{[2]}}{2} - 1$ times. The time complexity of the algorithm is therefore $O(p^{[2]})$.

For type 0, there are $m^{[2]}$ runs of length $r^{[2]} + 1$ in the intersection pattern and $n^{[2]}$ runs of length $r^{[2]}$. Therefore $p^{[1]} = m^{[2]}r^{[2]} + n^{[2]}(r^{[2]} + 1)$. In a run of order 2, type 0 and length r there are $r - 1$ short runs of order 1 and one long run.

$$\begin{aligned} n &= m^{[2]}(r^{[2]}r^{[1]} + 1) + n^{[2]}(r^{[2]}r^{[1]} + r^{[1]} + 1) \\ &= p^{[2]}r^{[2]}r^{[1]} + p^{[2]} + n^{[2]}r^{[1]} \end{aligned}$$

For type 1, there are $n^{[2]}$ runs of length $r^{[2]} + 1$ in the intersection pattern and $m^{[2]}$ runs of length $r^{[2]}$. Therefore $p^{[1]} = n^{[2]}r^{[2]} + m^{[2]}(r^{[2]} + 1)$. In a run of order 2, type 1 and length r there are $r - 1$ long runs of order 1 and 1 short run.

$$\begin{aligned} n &= n^{[2]}(r^{[2]}r^{[1]} + r^{[2]} - 1) + m^{[2]}(r^{[2]}r^{[1]} + r^{[2]} + r^{[1]}) \\ &= p^{[2]}r^{[2]}r^{[1]} + p^{[2]}r^{[2]} + m^{[2]}r^{[1]} - n^{[2]} \end{aligned}$$

Hence $O(p^{[1]}) = O(\frac{n}{r^{[1]}r^{[2]}})$. $\square$

The periodic behaviour of the algorithm when looking at the results presented by slope can therefore be described by the dependence of the algorithm on the value of $p^{[2]}$. In Figure 5.7 we describe the values of $p^{[i]}$ for $i = 1, 2$ over the range of the slope, $\alpha = \frac{a}{d}$ where $d = 1024$, $a = 1, \ldots, d$, and a and d are in least common terms.

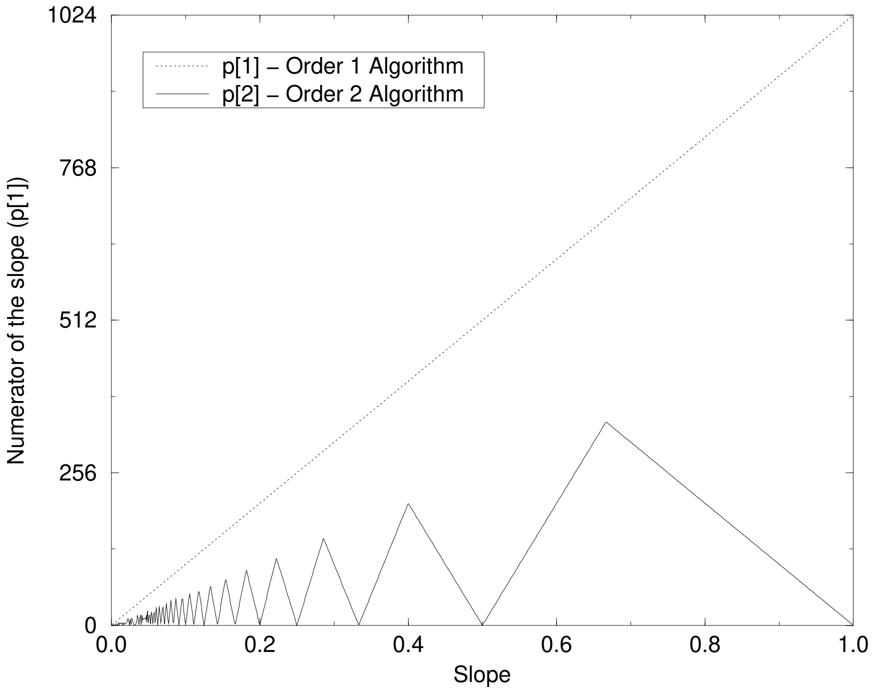


Figure 5.7: The number of runs of order i in the intersection pattern of the line $y = \frac{a}{d}x$ where $0 < a < d$ and $d = 1024$.

Table 5.3: Operator counts for the order 2 run–based algorithm.

Operator	Counts
Memory	$p^{[2]} + 2.5$
Assignment	$4p^{[2]} + 19$
Additive	$4p^{[2]} + 14.5$
Comparison	$2p^{[2]} + 2.5$
Division	2.5
Multiplication	4.5
Shift	4
Bitwise logical	0.5

5.4.4 The Run Length Slice Algorithm

For the comparison of the run based algorithms against Bresenham's run length slice algorithm, we implemented two versions of the run length slice algorithm. The first was the algorithm presented by Bresenham [17]. In this version, detailed in Algorithm B.3 in Appendix B, we replaced the setting of the individual pixels with the primitives `setRunOrder1` and `setRunsOrder1` where appropriate.

In Appendix B we show the correspondence between the run length slice algorithm and the theory for the run–based structure of the line we have presented. This correspondence allows us to adapt the run length slice algorithm to also use runs of order 2. In the second version of the algorithm we implement these changes. This version of the algorithm is detailed in Algorithm B.4 in Appendix B.

The behaviour of these algorithms can be explained by the values of the parameters $p'^{[1]} = \min(\Delta y, \Delta x - \Delta y)$ and $r'^{[1]} = \left\lfloor \frac{\Delta x}{p'^{[1]}} \right\rfloor$.

Theorem 5.4.4 *Where the* `setRunOrder2` *and* `setRunOrder1` *operations have* $O(1)$ *time cost, the run length slice line drawing algorithm has time complexity* $O(p'^{[1]}) = O(\frac{n}{r'^{[1]}})$.

The behaviour of $p'^{[1]}$ is illustrated in Figure 5.8. When the slope is less than one half, the run length slice algorithm behaves similarly to the order 1 algorithm. When the slope is greater than two thirds, the algorithm behaves like the order 2 algorithm. Between these two slopes, the algorithm is a mixture of the two. These facts are discussed in detail in Appendix B.

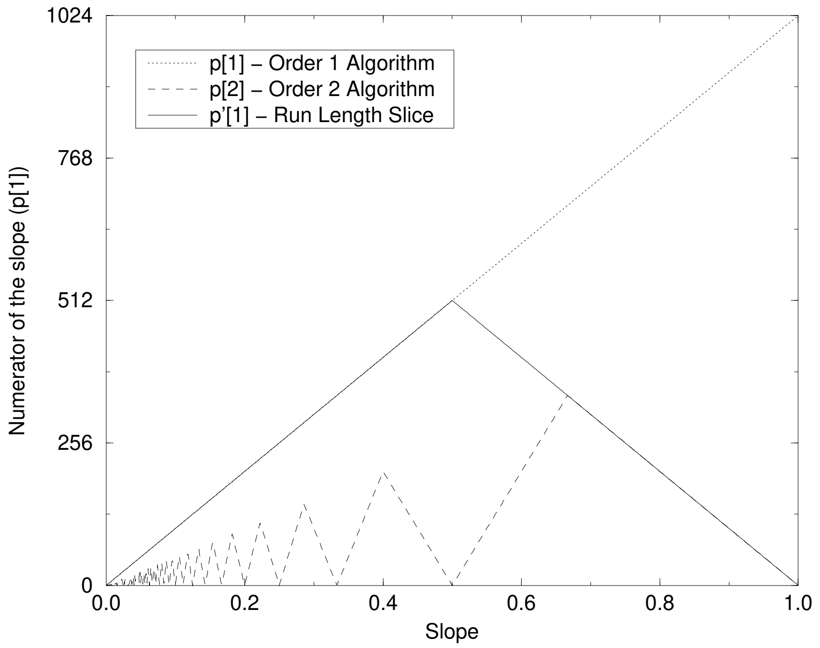


Figure 5.8: The number of runs length slices in the intersection pattern of the line $y = \frac{a}{d}x$ where $0 < a < d$ and $d = 1024$.

5.4.5 The Recursive Run–Based Algorithm

We now present the complexity analysis of the recursively defined run–based algorithm. Initially we will be discussing the time complexity of this algorithm to scan convert the line. We will then discuss possible implementations of the memory primitive used by the algorithm, the `setShape` command, and their time complexities. The recursion is based on the structure of the following segment of code for order i.

$order(p^{[i-1]}, m^{[i-1]}, n^{[i-1]}, t^{[i-1]}, b_0^{[i-1]})$
  {Calculate the run parameters for order i, type 0}
  {Draw initial partial run of order i}
  if $i < i_{max}$
    $order(p^{[i]}, m^{[i]}, n^{[i]}, t^{[i]}, b_0^{[i]})$
  else
    {Draw the $p^{[i-1]} - 1$ occurrences of the maximum order run}
  {Draw final partial run of order i}

From the connection between the hierarchy of runs in the intersection pattern of the line and continued fractions, we can use the wealth of information regarding the complexity of Euclid's gcd algorithm in analysing this algorithm.

Theorem 5.4.5 *The time complexity of the recursive run–based algorithm is* $O(\log p^{[0]} \cdot A_{\log p^{[0]}})$ *where* A_q *is the maximum cost of any q-bit arithmetic operation.*

Proof : By the correspondence between $p^{[i]}$ and the parameters of the Euclidean algorithm we have that $i_{max} = O(\log p^{[0]})$. The actual time cost is then $O(\log p^{[0]} \cdot A_{\log p^{[0]}})$. □

The memory primitive used to set a run of order i is the `setShape` command. As outlined in section 5.3 the `setShape` command is based on setting order 1 runs. Therefore if a run of order $i > 1$ has to be set it is divided into its composite runs of order $i - 1$ until order 1 in which the runs are set in memory. Therefore the `setShape` primitive is $O(p^{[1]} \log n)$. If a run of $p^{[i]}$ is used as the base primitive the time complexity will be $O(p^{[i]} \log n)$.

5.5 Conclusions

The problem of line drawing lies at the foundation of computer graphics. With the development of new technologies for the display and reproduction of images, the need for efficient techniques for line drawing is as relevant today as it was when Bresenham's presented his archetype line drawing algorithm in the early 1960's. Since then the development of line drawing algorithms has been reigned by the disparity between our knowledge of the structure of the digital line and our ability to produce efficient conditionally based algorithms which are easily converted into software and hardware appliances. In Chapter 2 we presented a theory for the hierartical run–based structure of the digitised line which bridges this gap. In this chapter, we presented three algorithms for line drawing based on the run structure of the digital line at orders 0, 1 and 2. From this set of algorithms, the order 2 algorithm outperforms Bresenham's pixel–based and run length slice algorithm except for very short lines.

The essential difference between Bresenham's algorithms and the order 1 and 2 run–based algorithms is that in Bresenham's algorithms lines are handled differently based on their slope being above or below the value of $\frac{1}{2}$. This equates to defining a type for order 1 based on the slope value. In Appendix B we will show that the order 0 algorithm is identical to Bresenham's pixel–based algorithm. We also discuss the similarities between Bresenham's run length slice algorithm and the algorithms of orders 1 and 2. The run–based structure we have presented essentially unifies these algorithms into a common line drawing theory. A common line drawing theory presents us with a number of opportunities. One of the most useful is that an algorithm can be developed to draw a line by using runs of a particular order depending on the length or complexity of structure of the line. We have presented one such algorithm that uses the maximum order of a line. However this algorithm is relatively unpragmatic as presented and requires further development. What appears to be a more promising avenue is to limit the depth of the recursion into the run hierarchy based on the cost of initialising the next order of run parameters compared to the cost of drawing the runs of the current order. For example, if the cost of initialising the next order of run parameters is equivalent to drawing n runs of this order, when the value of $p^{[i]} \leq n$, the recursion into the hierarchy ceases and the runs are drawn.

Such a technique also draws into question the use of the acceleration techniques such as symmetry and/or double, tri or quad steps [64, 142, 143, 109, 61, 68, 66, 145]. It should be investigated if they are more or less prudent than using the next order of runs.

In the experimental analysis of the algorithms we compared, we provided an environment in which a run of order 1 can be set in a time comparable to setting a single pixel. While this may appear to be contentious assumption on our behalf, I can find no reason to limit our work to the `setPixel` command when appropriate memory appliances to set a run have been available for over a decade [136]. Even if we do limit ourselves to reducing the problem to the pixel level the use of higher orders still improves the scan conversion of the line which is supported by the experimental and theoretical analysis.

It should be noted however that experimental results based solely on software implementation provides at best an indication of the performance of the algorithms presented. The system used for the experimental analysis was a multi–user time–sharing machine. Porting the algorithms to single–user machines or machines of different architectures may provide different results. Having said this, the theoretical analysis and the similarity of structure of the operator counts indicate that any disparities should be minimal. The implementation of these algorithms in hardware however is a different question. For hardware implementation, the logical simplicity of an algorithm is also important. The run–based algorithms being more complex than the pixel–based algorithms indicates that the application of the run–based techniques in hardware would need careful consideration.

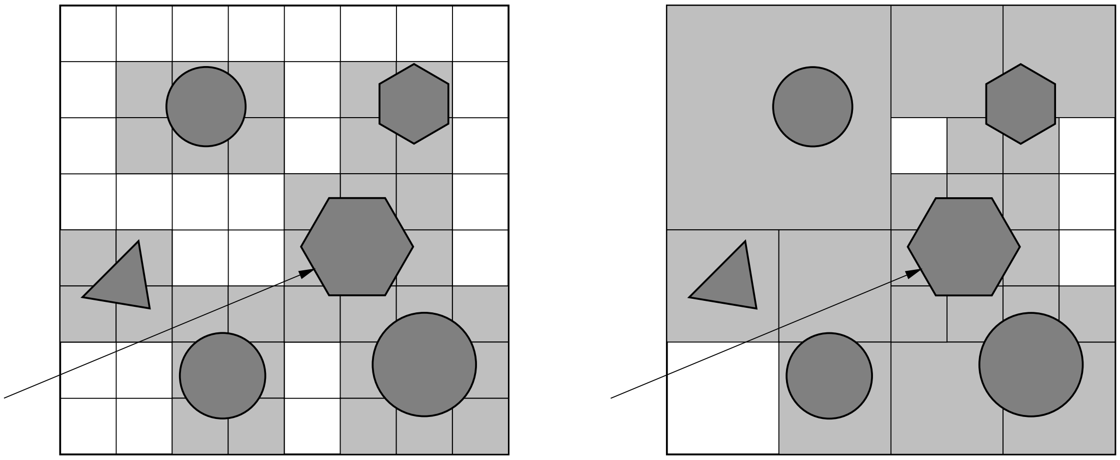
Chapter 6

Application to Ray Tracing in Computer Graphics

6.1 Introduction

From the inception of ray tracing as a global illumination model in computer graphics, the computational expense of finding the point of closest intersection of a ray and an object in the scene being rendered has been one of the major determining factors in the efficiency of the technique. Early analysis of ray tracing by Whitted [135] indicated that more than 95% of the running time of a ray tracer can be spent performing this operation. Hence a variety of techniques have been developed to lessen the dependence of ray tracing on the cost of finding the point of closest intersection. A survey of these techniques was produced by Arvo and Kirk [6] in which one of the main techniques they review is the use of spatial subdivisions.

The use of spatial subdivisions aims to increase the efficiency of finding the point of closest intersection by reducing the number of ray-object intersection tests that are conducted. By dividing the space in the scene being rendered, each ray can be traced through the subdivision until the ray enters a spatial element which contains an object or a part of an object from the scene. The ray is then tested against the object and if an intersection is found, the process ceases. If not, the ray is traced through the rest of the subdivision. The efficiency of the idea hinges on the ability



(a) Lattice - uniform spatial subdivision.

(b) Quadtree - nonuniform spatial subdivision.

Figure 6.1: Spatial subdivision types.

to efficiently trace the path of the ray through the subdivision and how tightly the subdivision describes the scene.

Two of the main spatial subdivision techniques are uniform and nonuniform spatial subdivisions. For a uniform spatial subdivision, the space surrounding the scene is uniformly segmented into a subdivision based on the lattice. Each cell in the lattice contains a list of the objects which intersect or are interior to the cell. One of the central nonuniform spatial subdivision techniques that is used is the three-dimensional octree or its two-dimensional equivalent the quadtree. To create an octree–based subdivision for use in ray tracing, Glassner [67] described a scheme where each spatial element of the octree is divided if two or more objects are contained in the voxel. The depth of the subdivision ceases, however, at a predefined level in the hierarchy. Therefore at the finest segmentation of space a spatial element of the octree may be associated with more than one object.

For example, consider the problem in two-dimensions. Given a simple scene we subdivide the space surrounding the scene using a two-dimensional lattice and a quadtree, Figure 6.1. The shaded areas in each figure depict a spatial element which contains a part or all of an object. If the ray to be tested intersects such a spatial element, a ray-object intersection test is conducted.

To trace a ray through a two-dimensional uniform spatial subdivision the Digitial Difference Analsyer (DDA) technique is typically used. In three dimensions the DDA technique has been extended by Fujimoto *et al.* [59, 60] by using two two-dimensional DDA algorithms and compressing the common dimension. The 2D-DDA and 3D-DDA techniques can also be used to trace a ray through a quadtree or octree [50, 75]. In the case of the quadtree, if each spatial element in subdivided to the maximum bound of the hierarchy, the spatial elements of the quadtree would describe the cells of the lattice. The ray can therefore be traced through this underlying lattice, in which each cell is contained in a subdivision of the quadtree.

Both spatial subdivision techniques have proven to be very successful in reducing the number of ray-object intersection tests that are required and therefore the time efficiency of the ray tracing technique. However the importance of developing even more efficient techniques is amplified by an observation made by Arvo and Kirk. They state that even though so much effort has been placed into making ray tracing more efficient through reducing or refining ray-object intersection tests, the ever increasing complexity of scenes being modeled has meant that Whitted's original analysis that 95% of the time taken to ray trace an image is spent finding the point of closest intersection still holds true.

6.2 Ray Tracing a Two-Dimensional Line

We will be present a run–based technique for calculating the intersection pattern of a ray and a lattice in two dimensions. The ray can be considered a line with real slope and intercept. We will assume that the lattice defines a spatial subdivision or an abstraction of one of predetermined size. We will only consider the frame of the lattice $\mathcal{L}_{(f_x,f_y)}$ where the ray intersects the cell $(0,0)$ and the upper bound of the frame is (f_x, f_y). Therefore we will represent the ray by the line $\ell : y = \alpha x + \beta$, where $0 < \alpha < 1$ and $0 \leq \beta < 1$.

Until now we have concentrated our study on the 8–connected intersection pattern of the line and the lattice. For the application of ray tracing however, each cell of the lattice through which the ray passes is required. This is essentially the problem of calculating the 4–connected intersection pattern except where corner intersections must be considered. The question of corner intersections is important as unnecessary object-ray intersection tests can be avoided. Calculating these unnecessary object-ray intersections may prove more expensive than checking for corner intersections when dealing with complex objects.

The ray tracing algorithm we will describe is a conditional style algorithm which utilises the first two orders of runs. We only use the first two orders for two main reasons. Firstly, we have to consider that the slope of the line, α, may be rational. As we are considering the cases of a slope of 0 and 1 as special cases, the maximum order of the runs in the intersection pattern has to be at least 2. More simply, there have to be at least two runs of order 1 in the intersection pattern. If we were to implement the algorithm using runs of order $i > 2$, we would have to take into consideration that the maximum order of the line may be less than i. As we discussed in Chapter 5, special cases where the maximum order is 1 or 2 can be included to improve efficiency, especially for short traces. Having too many special cases however will make the implementation overly verbose.

Secondly, we wish to develop a sequential conditional style algorithm in which the number of shapes of the runs is a minimum while maximising the number of cells maintained in a run. At order 2 only two run shapes are possible. For orders 3 and above, four are possible. Therefore the central decision loop would be considerably larger and more cumbersome.

The basic ray tracing algorithm in two dimensions is described in Algorithm 6.1.

Algorithm 6.1 The Two-Dimensional Run-Based Ray Tracing Algorithm.

{Calculate the necessary run parameters.
$p^{[0]}, p^{[1]}, m^{[1]}, n^{[1]}, r^{[1]},\ p^{[2]}, m^{[2]}, n^{[2]}, r^{[2]}$ and $t^{[2]}$.}
{Calculate the first numerator value of order 0, $b_0^{[0]}$.}

{Calculate the length of the first partial run of order 1, $r_0^{[1]}$.}
{Trace the initial partial run of order 1.}
{Calculate the initial numerator value of order 1, $b_0^{[1]}$.}

{Calculate the length of the first partial run of order 2, $r_0^{[2]}$.}

{Trace the initial partial run of order 2.}
{Calculate the initial numerator value of order 2, $b_0^{[2]}$.}

{For each remaining full length run of order 2:}
while {The ray is inside the frame}
    {Test for a possible corner intersection}
    {Trace the run}
{Test for final possible corner intersection}

{Calculate the length of the final partial run of order 2, $r_p^{[2]}$.}
{Trace the final partial run of order 2.}

{Calculate the length of the final partial run of order 1, $r_p^{[1]}$.}
{Trace the final partial run of order 1.}

Similar to the line drawing algorithm, we will be using a function or primitive to trace the cells contained in a run of order 2 or a run of order 1. We will be using four tracing primitives:

$traceRunOrder2Shape1(x, y, r_j^{[2]}, r^{[1]})$:

traces a run of order 2, shape 1 and length $r_j^{[2]}$ given the length of a short order 1 run is $r^{[1]}$,

$traceRunOrder2Shape3(x, y, r_j^{[2]}, r^{[1]})$:

traces a run of order 2, shape 3 and length $r_j^{[2]}$,

$traceRunsOrder1(x, y, l, r_j^{[1]})$:

traces a l order 1 runs of length $r_j^{[1]}$,

$traceRunOrder1(x, y, r_j^{[1]})$:

traces a single order 1 run of length $r_j^{[1]}$.

Each function traces the 4–connected intersection pattern defined by the run or set of runs. We will define the numerator sequence such that, if a corner intersection is to occur, it will occur between these runs. Therefore the single pixels whose corners may be intersected by the ray are traced separately.

To describe the process of tracing a ray through a lattice, we will be describing two separate but related algorithms. The first will be for rays with a rational slope, the second for irrational slopes. The reason for separating these algorithms is the different requirements for detecting corner intersections of each type of line. While the separation of these algorithms is undesirable because prior knowledge of the rationality of the slope is required, we will discuss possible solutions after we have presented the algorithm and the problems associated with combining the two.

6.2.1 Tracing a Ray with Rational Slope

We initially concentrate on the problem of tracing the ray of rational slope defined by the line $\ell : y = \frac{a}{d}x + \beta$ where $0 < a < d$ and $0 \leq \beta < 1$.

From section 3.2, we know that if the slope is rational, the intersection pattern, $\mathcal{P}_\ell$, will be identical to the intersection pattern, $\mathcal{P}_{\ell'}$, produced by the line $\ell' : y = \frac{a}{d}x + \frac{b}{d}$ where $b = \lfloor \beta d \rfloor$. We also know that a corner intersection can only occur if $\ell = \ell'$. That is, if the intercept β is also a rational number, the denominator of the slope and the intercept are the same and $\gcd(a, d) = \gcd(b, d)$. In this case a corner intersection occurs at the start of the run of order i associated with the numerator sequence value 0. Conversely, if the intercept is irrational or is rational but has a different denominator to the slope, no corner intersections can occur.

As we are utilising only the first two orders of the approximation, the first step of Algorithm 6.1 is to calculate the run parameters of the first two orders, $p^{[0]}$, $p^{[1]}$, $m^{[1]}$, $n^{[1]}$, $r^{[1]}$, $p^{[2]}$, $m^{[2]}$, $n^{[2]}$, $r^{[2]}$ and $t^{[2]}$. The calculation of these parameters is described in section 2.5.1. The initial numerator value of order 0 from equation 3.1, is simply $b_0^{[0]} = b$. As the slope is rational we can also include a number of special cases discussed in section 5.2.3, considering that we are using the first two orders. To help describe the progression of the algorithm, we will use the example of the line $\ell' : y = \frac{17}{41}x + \frac{23}{41}$ which includes a corner intersection, Figure 6.2.

We know that if the initial numerator value of order 0 is less than $p^{[1]}$, $b_0^{[0]} < p^{[1]}$, then the initial partial run length of order 1 is zero and the initial numerator value of order 1 is the same as order 0, $b_0^{[1]} = b_0^{[0]}$. If not, the initial partial run length of order 1, $r_0^{[1]}$, and the initial numerator value of order 1, $b_0^{[1]}$ are calculated and the initial partial run is traced. One special condition that can occur in a small number

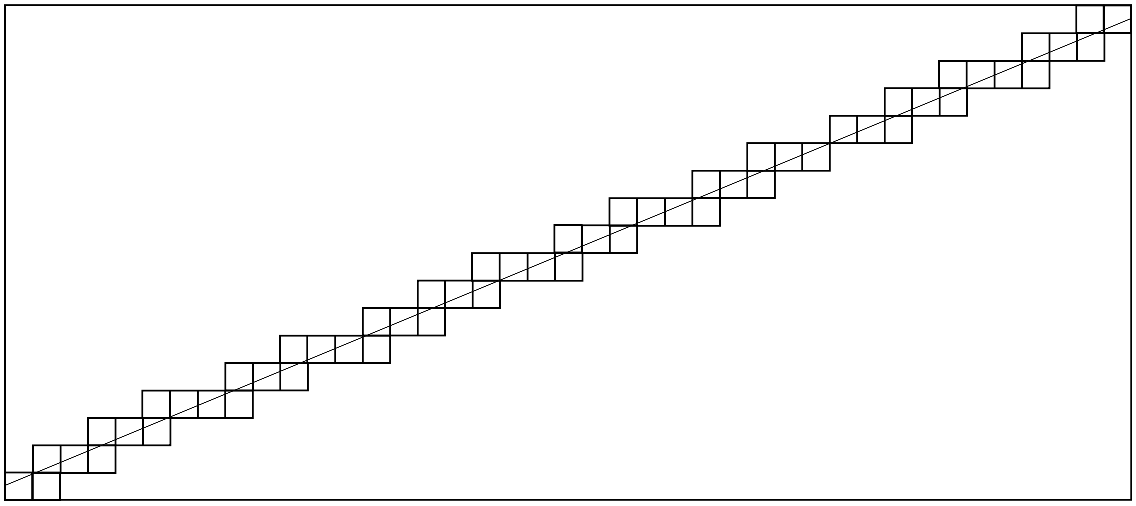


Figure 6.2: The intersection pattern of $\ell' : y = \frac{17}{41}x + \frac{23}{41}$.

of cases is that initial partial run of order 1 is greater than the width of the frame. If so, we set $r_0^{[1]} = f_x$, trace the run and end the algorithm. Similarly for the initial partial run of order 2.

For the example we have $r_0^{[1]} = 2$ and $b_0^{[1]} = 16$ from Figure 6.3.

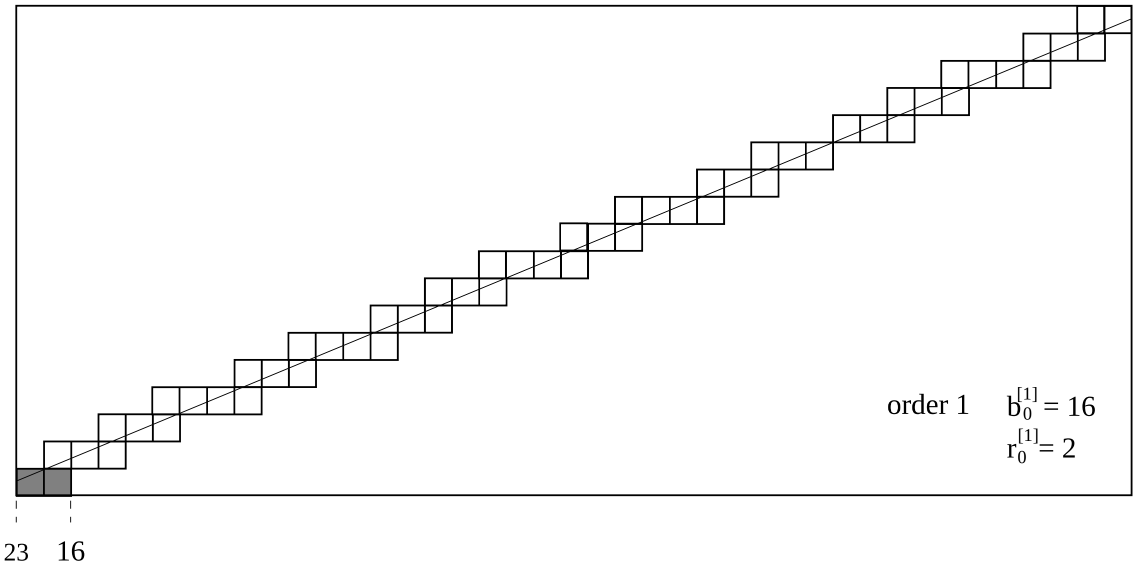


Figure 6.3: Step 1: The initial partial run length and initial numerator value of order 1.

Similarly for order 2, we know that if the initial numerator value of order 1 is less than $p^{[2]}$, $b_0^{[1]} < p^{[2]}$, then the initial partial run length of order 2 is zero, and the initial numerator value of order 2 is the same as order 1, $b_0^{[2]} = b_0^{[1]}$. If not, the initial partial run length of order 1, $r_0^{[2]}$, and the initial numerator value of order 1, $b_0^{[2]}$ are calculated and the initial partial run is traced.

The difference between orders 1 and 2 however is that runs of order 2 have the property of shape. If the type $t^{[2]} = 0$, we have shape 1: ls^+. Therefore the initial partial run of order 2 will consist of $r_0^{[2]}$ short runs of order 1. If the type $t^{[2]} = 1$, we have shape 3: l^+s and the initial partial run still has shape 3 but has been truncated. Therefore a order 2 run of shape 3 and length $r_0^{[2]}$ is traced taking care that if the length is one that only a single long run of order 1 is traced.

From the example we have $r_0^{[2]} = 2$ and $b_0^{[2]} = 2$. For order 2 we have type 0, so two short runs of order 1 are traced.

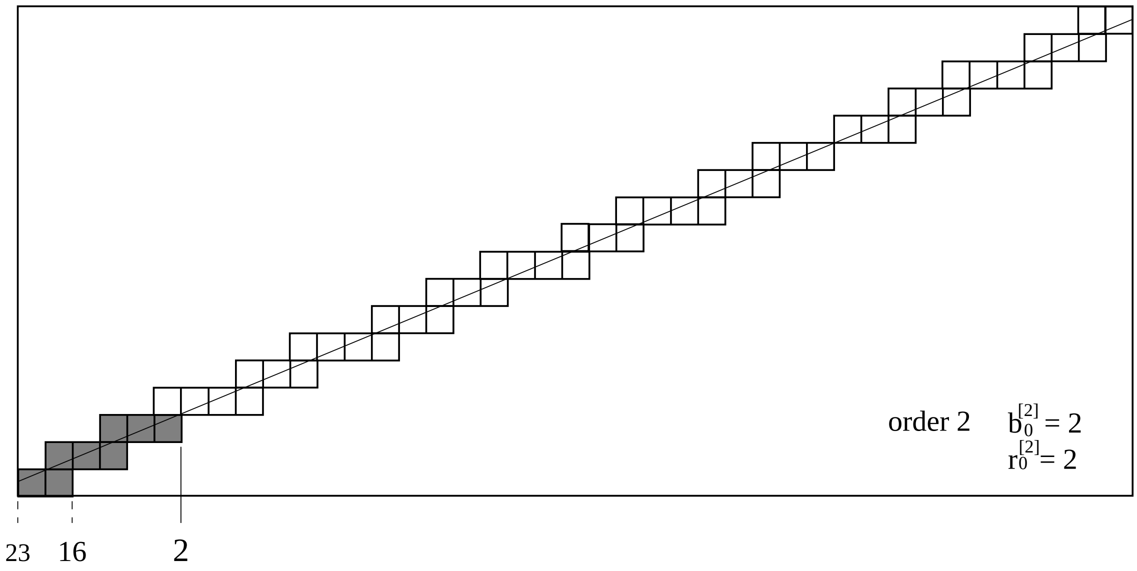


Figure 6.4: Step 2: The initial partial run length and initial numerator value of order 2.

From here we will be tracing the full length runs of order 2. We have the added complexity that a corner intersection may occur. As discussed, a corner intersection will occur at each occurrence of the numerator value $b_j^{[2]} = 0$ if $\ell = \ell'$. If the numerator value is non-zero, or if it is zero and corner intersections are not possible, the lattice cell that connects the previous run and the following run is traced. If the value is zero and corner intersections are possible, the lattice cell is skipped and the next run length or order 2 is calculated and the run is traced.

For our example we trace the full length runs until the current numerator value is zero. The lattice cells depicted by light grey are those pixels that are set between the runs where a corner intersection could occur if the numerator value is zero.

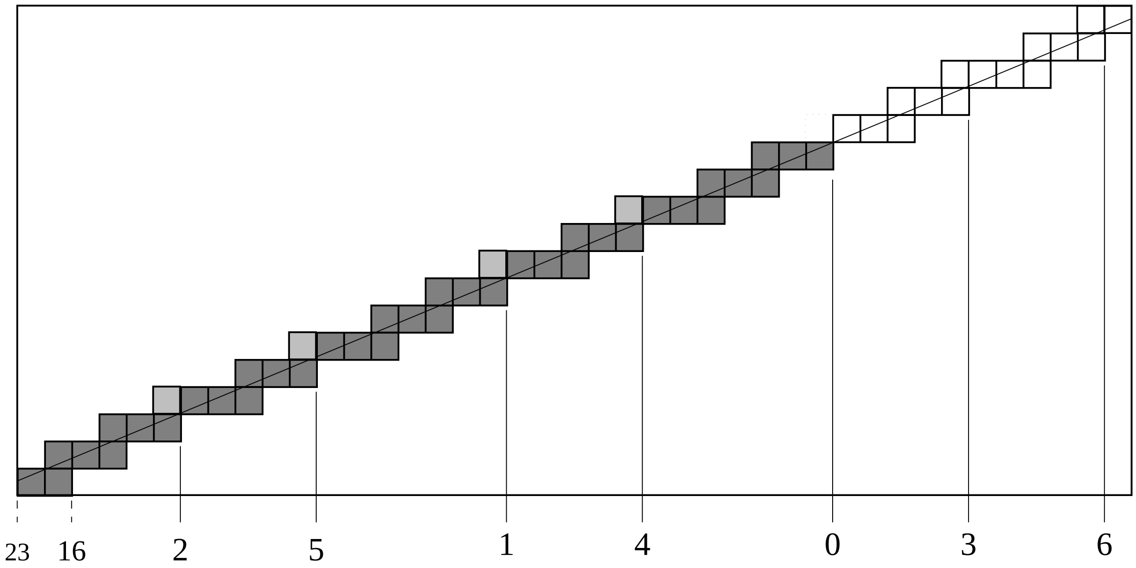


Figure 6.5: Step 3: Tracing the full length runs until a numerator value of 0.

At this point, we need to know if $\ell = \ell'$. If so, a corner intersection will occur. The choice comes down to if $\beta = \frac{b}{d}$ or $\beta d = b$. It should be noted that if $d < f_x$ then more than one corner intersection can occur within the frame, $\mathcal{L}_{(f_x, f_y)}$.

In the case of our example, a corner intersection does occur and the lattice cell is skipped. The remaining full length runs are traced and the final possible corner intersection is tested for.

All that remains is to calculate the final partial runs of order 2 and order 1 and trace them. The final partial runs could be calculated from the results of section 3.2. However there is a simpler way. Given that the position of the next run is given by (x, y), the number of full order 1 runs will be $r_p^{[2]} = p^{[1]} - y$ as the initial cell intersected by definition is cell $(0, 0)$. The number of order 0 runs left to trace will be $r_p^{[1]} = f_x - x$.

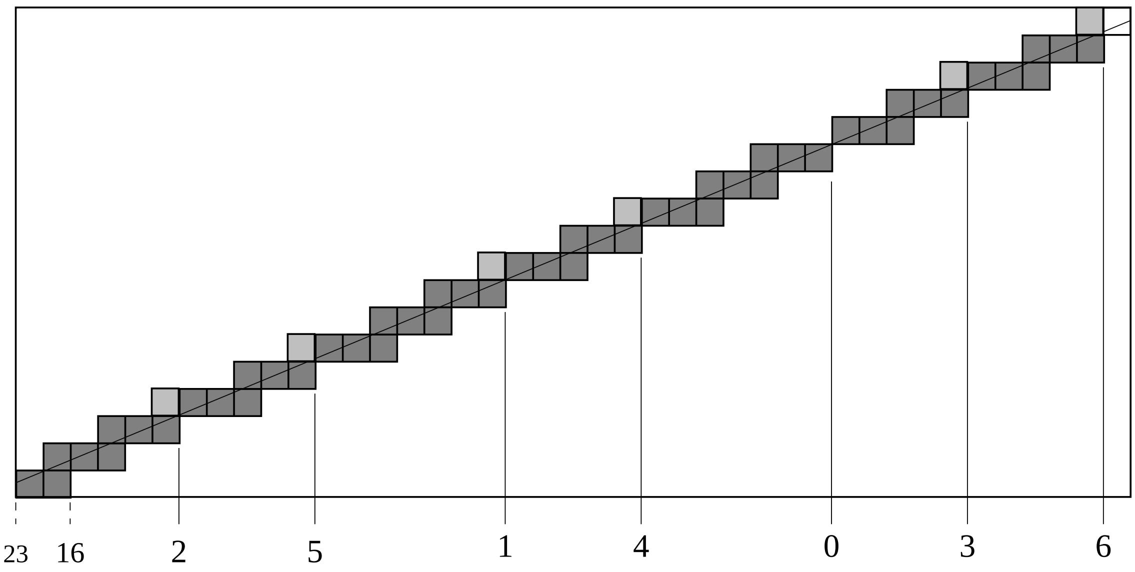


Figure 6.6: Step 4: The corner intersection is detected and the remaining full length runs are traced.

6.2.2 Tracing a Ray with Irrational Slope

The algorithm for tracing a ray with an irrational slope follows an almost identical development to the algorithm for a ray with a rational slope. The differences are that the irrational slope is replaced by a rational approximation and during the approximation process the run parameters are calculated. Also the case of tracing the ray is made more complex by the fact that the approximation for a range of intercept values may not be the best possible approximation.

The first phase of the algorithm is to approximate the line ℓ with a line with rational slope and intercept, $\ell' : y = \frac{a}{d}x + \frac{b}{d}$ and calculate the run parameters. The approximation process is described in section 4.5.2. The continued fraction expansion of α is calculated until the slope forms a suitable approximation over the frame $\mathcal{L}_{(f_x, f_y)}$, the n-th partial denominator $Q_n \geq f_x$. The difference between the approximation, $\frac{p^{[1]}}{p^{[0]}} = \frac{a}{d} = \frac{P_n}{Q_n}$, and the slope is $0 < \alpha - \frac{a}{d} < \frac{1}{d^2}$.

From the continued fraction expansion of α the run parameters are also calculated for all orders during the approximation process. As we are utilising only the first two orders of the approximation, we only require the run parameters of the first two orders, $p^{[0]}, p^{[1]}, m^{[1]}, n^{[1]}, r^{[1]}, p^{[2]}, m^{[2]}, n^{[2]}, r^{[2]}$ and $t^{[2]}$. The initial numerator value of order 0 can be calculated from equation 3.1, $b_0^{[0]} = b = \lfloor \beta d \rfloor$.

The problem of two possible approximations to the line ℓ is also easily handled. Given the value of the intercept β, if $\frac{1}{d} - \frac{f_x}{d^2} \leq \beta - \frac{b}{d} < \frac{1}{d}$ from Theorem 3.3.5 we have that the best possible approximation can be one of the two lines $\ell' : y = \frac{a}{d}x + \frac{b}{d}$ or $\ell'' : y = \frac{a}{d}x + \frac{b+1}{d}$.

However, from Theorem 3.3.4 we know that the intersection patterns produced by either line will be identical up to the run defined by the value of 0 in the numerator sequence associated with the line ℓ'. Therefore we trace the line ℓ'' until the numerator value of 0 is found. This may not occur within the bounds of the frame, if not, either of the lines ℓ' or ℓ'' is a suitable approximation of ℓ over the frame $\mathcal{L}_{(f_x,f_y)}$.

If the numerator value of 0 does occur within the frame, which line is the better approximation for ℓ needs to be decided. The value of equation 3.15 is calculated, $\alpha x + \beta - y$, where (x, y) is the lattice position at the beginning of the next run associated with the numerator value 0. If it is greater than or equal to zero, then by Theorem 3.3.5, the line $\ell'' : y = \frac{a}{d}x + \frac{b+1}{d}$ is the best rational–based approximation for the line ℓ. If it is less than zero, then the best approximation is the line $\ell' : y = \frac{a}{d}x + \frac{b}{d}$. If the line ℓ' proves to be the better line of approximation, very little needs to be done to continue tracing the ray. The current numerator value has to decremented so that the numerator sequence associated with the line ℓ' is calculated. Secondly the previous run length has to reevaluated. From Theorem 2.4.8 it is apparent that for the case of $t^{[2]} = 0$, if the previous run was long it should be made short. Conversely for the case of $t^{[2]} = 1$, the previous run was short and should be made long. It is therefore prudent to calculate the numerator sequence one value ahead of what is needed.

To handle corner intersections, from section 3.3.2, if the slope is irrational, only one corner intersection can occur and only if the intercept is also irrational. A corner intersection occurs when the value of equation 3.15 $\alpha x + \beta - y = 0$. Therefore the testing for corner intersections can easily be incorporated into the process described above for determining which line ℓ' or ℓ'' is the best approximation for ℓ.

6.3 Ray Tracing a Three-Dimensional Line

In this section we will describe our initial ideas regarding the tracing of a three dimensional ray through a three-dimensional lattice.

The technique we will describe follows a similar line to the development of the 3D-DDA technique from the DDA algorithm. Fujimoto *et al.* [59, 60] developed the 3D-DDA algorithm by tracing two two-dimensional DDA based lines through the three-dimensional lattice and compressing the common dimension.

In the initial experiments, two order 1 traces are made through the XY and XZ projections of the lattice. This process follows the process described in the previous section. Each of the XY and XZ traces returns the runs of order 1 in these planes intersected by the projected two-dimensional rays.

Take the first run from each of the XY and XZ traces and compare their lengths $runXY_0$ and $runXZ_0$. If the lengths of these runs are the same, the length of the first three-dimensional run is also this length, $runXYZ_0 = runXY_0 = runXZ_0$. Therefore, to calculate the length of the next three-dimensional run, the next two two-dimensional run lengths, $runXY_1$ and $runXZ_1$, are calculated and the process continues.

If one of the two-dimensional lengths is shorter than the other, take the case where $runXY_0 < runXZ_0$, the length of the three-dimensional run is the length of the shorter, $runXYZ_0 = runXY_0$. To calculate the next run length, the length of the original longer run is reduced by the shorter, $runXZ_0 = runXZ_0 - runXY_0$ and the next two-dimensional run length, $runXY_1$, is calculated. The process continues as such.

If a corner intersection occurs in one of the two-dimensional traces but not the other, an edge intersection occurs in three dimensions. If a corner intersection occurs in both two-dimensional traces, a corner intersection occurs in three dimensions.

6.4 Experimental Results and Analysis

To determine the time complexity of the run–based ray tracing model, we will present experimental results which compare the algorithms presented against the standard DDA algorithms in two and three dimensions.

The experimental and theoretical analysis of the ray tracing application follows a similar vein to the line drawing application. The experiments were conducted using a Sun SparcStation 20 running the Ultrix operating system. The algorithms were implemented in C++. To measure the amount of time to complete an algorithm, the Unix system call `times()` was used. To gain a measurable cost in cpu seconds, each algorithm was run a number of times within a loop. The time taken to complete the loop was measured. The programs were compiled using the Sun CC compiler. No compiler–based optimisation was used for the same reasons as discussed in section 5.4.

6.4.1 The Two-Dimensional Ray Tracing Algorithm

For the two-dimensional ray tracing experiment, two algorithms were compared. The DDA algorithm using fixed point arithmetic and the order 2 run–based algorithm. The first part of the experiment sought to describe the dependence of the execution speed of each of the algorithms on the length of the ray being traced.

To perform this experiment a number of frame sizes were defined, $f_0 = \{1, 2, \ldots 32\}$, $f_1 = \{32, 48, \ldots 256\}$, $f_2 = \{256, 288, \ldots 1024\}$ and $f_3 = \{1024, 1152, \ldots 2048\}$. A random set of rays with irrational slope were calculated by randomly defining the slope and intercept of the line representing the ray. The set of rays was traced through the entire frame using each algorithm. The time taken using each algorithm was totaled. In total 250 rays were generated for each frame and they were traced 250 times by each algorithm. The random number generator used was the `erand48()` function from the drand48 C library. The results of the experiment are presented in Figure 6.7.

The second part of the experiment was to describe the dependence of the execution

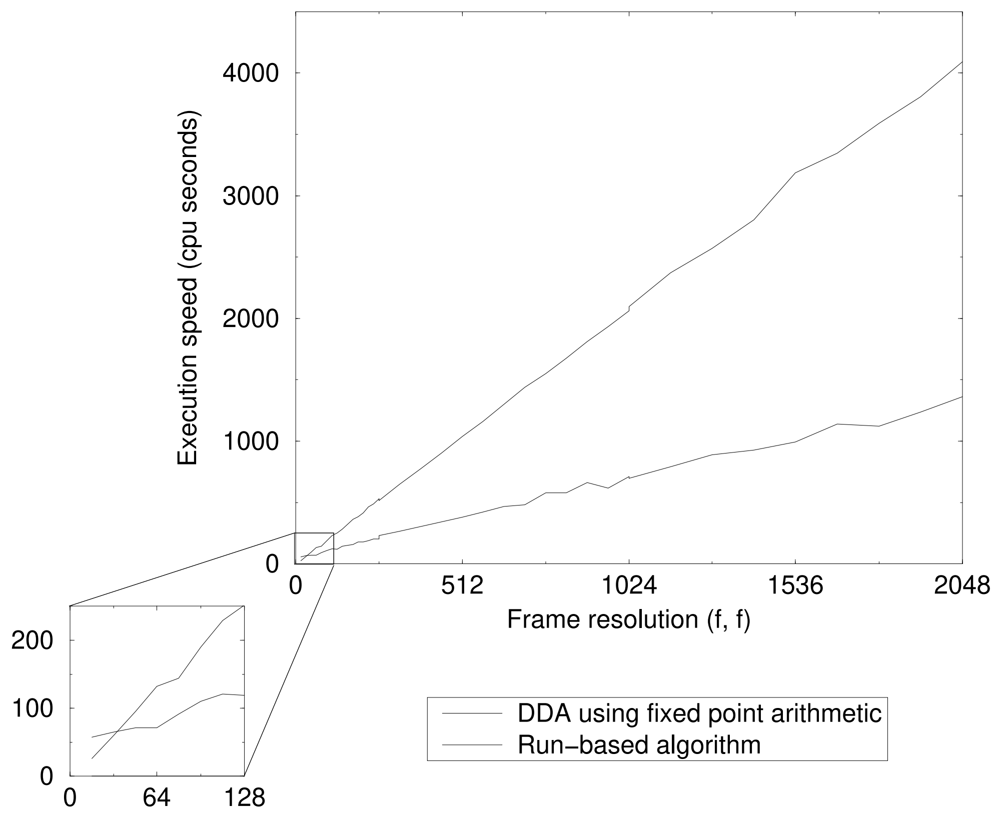


Figure 6.7: Execution speed in cpu seconds of the run–based algorithm and the DDA algorithm implemented using fixed point arithmetic for the intersection of the line $y = \alpha x + \beta$ and the two-dimensional lattice of size (f, f).

speed on the slope of the line. The frame size was set at $f = 1024$. In total 1000 rays with irrational slope were calculated by randomly defining the slope and intercept of the line representing the ray. Each were traced through the frame 100 times by each algorithm. The time taken to complete each set of traces was sorted by slope and is presented in Figure 6.8. The behaviour of each algorithm follows the discussion presented in Chapter 5.

6.4.2 The Three-Dimensional Ray Tracing Algorithm

We present our preliminary results of our software–based experiments of ray tracing a three-dimensional lattice comparing the run–based ray tracing technique of order 1 and the 3D-DDA technique. The reader should note that these experiments

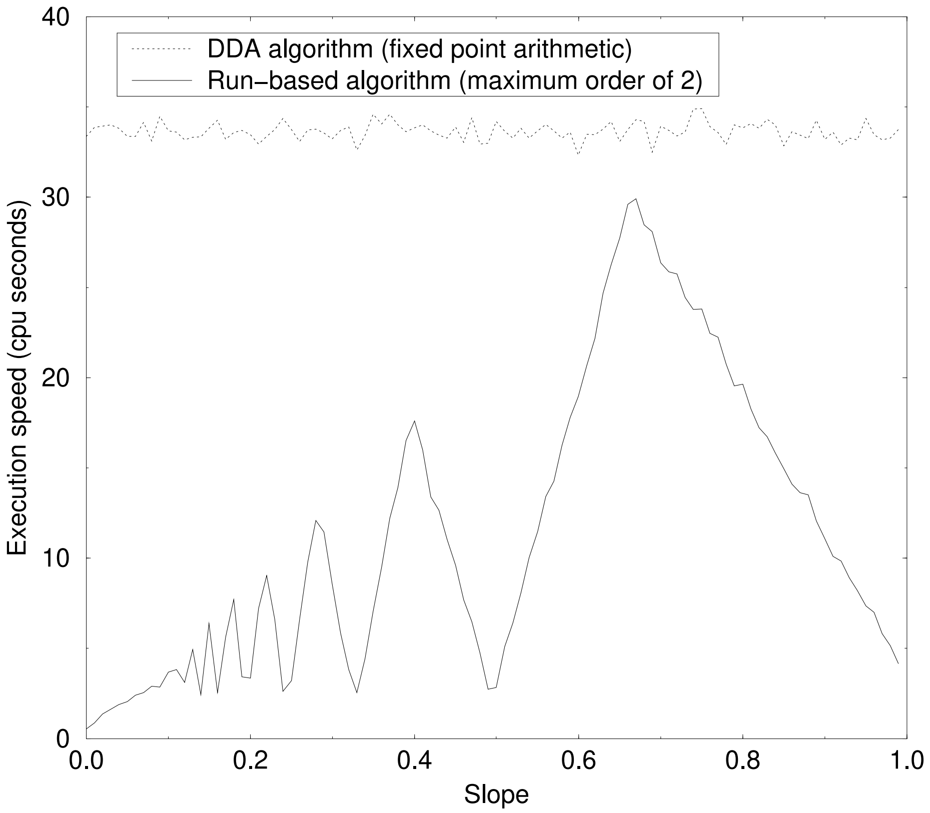


Figure 6.8: Average execution speed in cpu seconds of the run–based algorithm and the DDA algorithm implemented using fixed point arithmetic for the intersection of the line $y = \alpha x + \beta$ and the two-dimensional lattice of size (f, f).

are not conclusive but are included because they support our opinion that further development of run–based ray tracing in three dimensions is warranted.

To compare the relative speeds of the run–based technique and the 3D-DDA, we measured only the time taken to calculate the cells intersected by the ray. The 3D-DDA algorithm calculated each cell intersected. The run–based algorithm calculated each three-dimensional run of order 1 that was intersected. Therefore if each cell has to be incrementally checked for the containment of an element of the scene the results of the experiment will be diminished.

We limited our experiment to rays defined by the three-dimensional line $y = \alpha x$. No corner intersections were calculated. Therefore these results are not conclusive but indicate that if the two-dimensional algorithm for ray tracing a lattice could

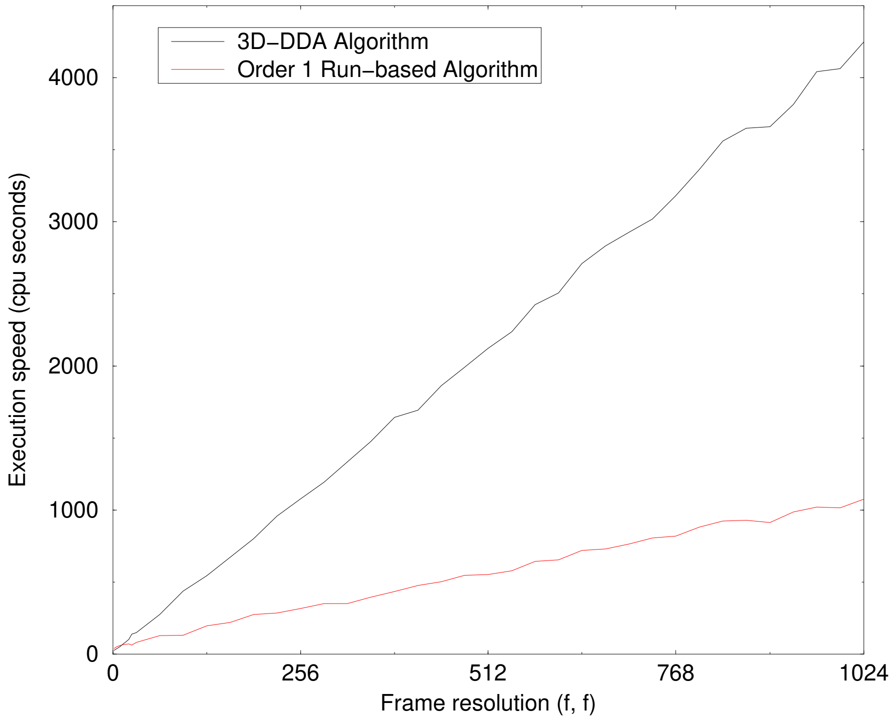


Figure 6.9: Execution speed in cpu seconds of the run–based algorithm of order 1 and the 3D-DDA algorithm implemented using fixed point arithmetic for the intersection of the line $y = \alpha x$ and the three-dimensional lattice of size (f, f, f).

be unified and is more time efficient than the two-dimensional DDA algorithm, the run–based algorithm in three dimensions should improve on the 3D-DDA.

To conduct the experiment a frame size was defined. A random set of rays were calculated by randomly defining the slope and intercept of the line defining the ray. The random number generator used was the `erand48()` function from the drand48 C library. The set of rays was traced through the entire frame using each algorithm. The time taken using each algorithm was totaled.

Each ray was traced 100 times. An equivalent set of 500 rays was traced through each frame by each algorithm. The frame sizes used were, from 2 to 26 with steps of 4, (i.e. $2, 6, 10, \ldots, 26$) and from 32 to 1024 with steps of 32. The results are presented in Figure 6.9.

6.5 Conclusion

We have described a run–based algorithm for tracing a ray through two-dimensional and three-dimensional lattices. The results, while preliminary, suggest there is merit in pursuing run–based ray tracing to improve current techniques.

The only difficulty of this technique is that prior knowledge of the rationality of the slope of the ray is required, unless corner intersections can be ignored and needless object-ray intersection tests are not unwelcome. While the occurrence of corner intersections is rare and would have to align with a lattice cell that contains an object, making needless tests even rarer, it still cannot be ignored.

The knowledge of rationality of slope is not difficult to build into the process of ray tracing. From the definition of primary rays, to the calculation of reflected, transmitted and shadow rays, the formulae defined provide all the information needed to define the rationality. While a single algorithm that incorporates rays of rational and irrational slope would be more easily accepted by those implementing ray tracing techniques, the increased performance of a run–based ray tracing scheme which used only integer arithmetic would surely prove worthwhile.

It may also be possible to combine the two algorithms for tracing a ray with rational and irrational slope into one algorithm. One possibility might be to develop a real or floating point version of the algorithm based on the results returned by the approximation process. The extra information contained in these real equivalents of the run parameters may contain the necessary detail to properly handle a combined algorithm.

6.5.1 Use in Nonuniform Spatial Subdivision Techniques

While uniform spatial subdivisions are common in a number of applications such as volume visualisation and medical imaging, the memory requirements needed to store such an exhaustive subdivision means their use in many applications is limited. Nonuniform spatial subdivisions, which are more sensitive to distribution of space within a scene, are more commonly used in applications such as ray tracing. One of the most common being the octree or in two dimensions, the quadtree.

Arvo and Kirk [6] compare the attributes of uniform and nonuniform spatial subdivision schemes including the lattice and octree. In their comparison they state, regarding uniform spatial subdivisions

> ... there are two distinguishing features of this approach which are direct consequences of the voxel regularity: (1) the subdivision is totally independent of the structure of the environment, and (2) the voxels pierced by a ray can be accessed very efficiently by incremental calculation. The first is a disadvantage which must be weighed against the benefits of the second.

The use therefore of an octree appears to answer this criticism. The octree is adapted to the structure of the environment and the voxels intersected by the ray can be calculated incrementally with little overhead using a technique based on the underlying lattice. The octree structure however bears no relationship to the ray.

If however the spatial subdivision scheme is based on runs, the subdivision scheme would have a close relationship to the environment but also to the intersection pattern of the ray and the lattice and hence the ray. An example of how the space could be subdivided in given in Figure 6.10. The cells of the lattice which contain a part of an object are organised into runs. For each row of the lattice, the run can be described by its column of origin and its length. The runs can then be sorted into lists based on their column of origin.

As the ray is traced through the lattice, the problem is reduced to finding an overlap between a run of intersected cells and a run in a particular list.

A run–based subdivision scheme also provides opportunities for increasing the efficiency of the tracing algorithm. The two two-dimensional traces are logically separate. Therefore one two-dimensional plane can be chosen and traced. Only when a possible intersection is found is the second trace performed. We know the coordinate of the possible intersection on the mutual axis, therefore an intersection cannot occur until the second trace has reached this position along the mutual axis.

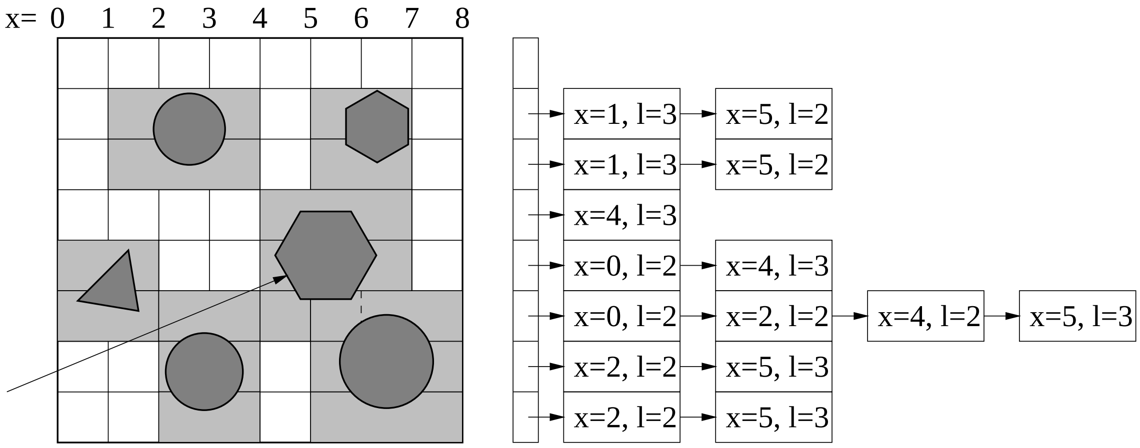


Figure 6.10: A run–based nonuniform spatial subdivision scheme.

Such an idea could provide an extremely efficient technique for ray tracing using nonuniform spatial subdivisions. However there are a number of problems and considerations which need to be answered. Firstly and most importantly, a universal run–based ray tracing technique is needed where the knowledge of the rationality of the slope of the ray is not required.

Secondly, the memory requirements for such a run–based spatial subdivision technique would in all probability be slightly more than for an octree subdivision. While a detailed analysis is required, we can offer some initial thoughts on the matter. The worst case scenario for an octree and a run–based subdivision scheme is the checkerboard. Each subdivision within the octree is fully extended to its maximum hierarchy and a run is associated with every cell of the lattice. In this case, the run–based subdivision scheme should be marginally better than the octree description. However for the best case scenario of any empty scene, the octree representation will be far better as the lists associated with each row will be empty but will still need to be represented.

Lastly, the scene needs to be subdivided and the run–based representation of the objects needs to be calculated. This should be quite an efficient process however. The technique of Z-buffering has been well researched and has been proven to be very efficient in scan conversion applications. The process of generating the run–based subdivision is very similar to Z-buffering except the process needs to store information about the depth of an object.

Chapter 7

Conclusions

While it has been known for quite some time that the digitised line is composed of a hierarchical structure of runs, this knowledge has had little bearing on the techniques used in computer graphics as there has been no efficient method to reproduce the run structure of the line except at the lowest levels [20].

In this dissertation we have presented a structural definition of the digitised line based on the hierarchy of runs in the intersection pattern of lattice cells produced by intersecting the line and the lattice. We have provided a natural extension to the definition of a run to include the definition of a pixel (lattice cell) and the recursive definition of a run of runs. We have described how the intersection pattern can be generated using a conditional style algorithm at any level within the hierarchy thus bridging the gap between the structural and conditional descriptions of the digitised line. The description of the hierarchy of runs and the conditional run–based algorithm provides a way of unifying line drawing algorithms such as Bresenham's pixel–based line drawing algorithm and run length slice algorithm and developing even more efficient algorithms.

The connection between the structure of runs in the intersection pattern, Euclid's gcd algorithm and regular continued fractions in the literature is incomplete. We have shown that the connection is between the structure of runs, the least remainder gcd algorithm and non-ordinary continued fractions. This connection allows for continued fractions to be considered from a solely geometric viewpoint and for the rational based approximation of lines with irrational slope and real intercept. This result is applied to ray tracing a uniform spatial subdivision where the use of structural features of the intersection pattern of the subdivision and a ray have not been used before.

To investigate certain properties of the digitised line we introduced a technique based on matching the patterns of runs within the line which we used to prove the sufficiency of the run parameters in defining the line and to produce all of the lines of lesser length and integer endpoints that are contained within a line.

During the development of this work, we have made a number of decisions which warrant comment. Firstly, as we have discussed in Chapter 5 and Appendix B, we decided not to define a type for order 1. This is essentially the difference between Bresenham's run length slice algorithm and the run–based algorithms of orders 1 and 2. However, the effects of this decision should be investigated. The reason we took this path was based on the traditional definition of the run, its alignment to the x-axis and the ease in implementing a memory primitive based on this alignment.

Secondly, it could be asked why we choose the definition of the structural parameters $m^{[i]}$ and $n^{[i]}$ as we did. Why didn't we set one to be the number of long runs of order i, and the other to be the number of short? The answer is simple, it complicates the calculation of the numerator sequence. The structural parameters are calculated in the initialisation phase once for each order. The numerator sequence is calculated in a looping structure which requires $p^{[i]}$ iterations. The complexity of the numerator sequence therefore dominates the behaviour of the algorithms we have presented and any inefficiencies introduced in its calculation is magnified $p^{[i]}$ times.

Lastly, it should be observed that for all of the results we have presented, the order 1 run behaves exactly as a run of order i and type 1. In fact defining a run of order 1 to be of type 1 by default seems a reasonable suggestion. However, I cannot explain why, I am not sure a reason exists, and I find it difficult to make such a definition without some justification for it, other than it works.

To complete our discussion, it is time to discuss future work and endeavours.

7.1 Future Work

7.1.1 Parallelisation

A variety of techniques for parallel line drawing have been previously described [5, 114, 131, 100, 139, 140, 94, 69, 70]. To parallelise the algorithms we have presented, a number of avenues exist. For the algorithms which use the numerator sequence to generate the intersection pattern, we can use a similar procedure as presented by Wright [139]. Wright looked at parallelising Bresenham's pixel–based line drawing algorithm. He divided the line into segments, calculated the start and end points of each line segment and drew the segments in parallel. For a run–based algorithm which uses the numerator sequence, each run can be drawn in parallel. The numerator sequence can be calculated in parallel using the results of Corollary 2.4.5. Therefore we can calculate the length of each run. For order 1, we can calculate the position of each run from the line equation as we know that each run has a unique y-coordinate. For order $i > 1$ the problem is more complicated. To calculate the position of the j-th run of order i we need to solve for $(x_j^{[i]}, y_j^{[i]})$, the Diophantine equation given by Equation 2.24, $b_j^{[i]} = p^{[1]} x_j^{[i]} - p^{[0]} y_j^{[i]}$.

A more promising approach for higher order algorithms, is parallelising the primitive which sets a run of order i. We know the position and shape of the run and the lengths of the runs of order $i-1$ which comprise it. Therefore to draw a run of order i we can draw the composite runs of order $i-1$ in parallel. This approach would have the greatest affect in the recursive–based algorithm we discussed in section 5.3.

7.1.2 The Properties of the Numerator Sequence

More has yet to be said about the numerator sequence. One aspect which can be expanded is the symmetry of the numerator sequence. Consider the numerator sequence of order 2 of the line $\ell : y = \frac{17}{41}x$ which is type 0 (figure 2.3).

$$b_{j=0\to 7}^{[i]} = \{\ 03625140\ \}$$

The run lengths of order 2 that comprise the intersection pattern, $\mathcal{P}_\ell$ are

$$r^{[i]}_{j=0\to6} = \{\ sslslsl\ \}$$

However if we define $b'^{[i]}_j = p^{[i]} - b^{[i]}_j$ we produce the altered numerator sequence

$$b'^{[i]}_{j=0\to7} = \{\ 74152627\ \}$$

To calculate the run lengths we can then use the type 1 rules which would lead to a reduction in the code needed to implement the software appliances we have described.

The similarity between the numerator sequence and the Spirograph introduced by Dorst and Duin [45] is also interesting.

7.1.3 Line Drawing

In Chapter 5 and Appendix B we have discussed the use of the results presented to unify various line drawing algorithms. We also discussed the use of recursively based algorithms for drawing exceptionally long lines. One promising outcome of this work may be an algorithm which adjusts the order of the runs used to draw the line based on the characteristics of the line.

We also discussed the prospect of a run–based primitive to complement the `setPixel` command. What remains is to complete its effective implementation in hardware. It may also have application to drawing anti-aliased lines. The extra pixels which are effected by drawing an anti-aliased line are those which lie before and after each run of order 1. It should be investigated if the results presented by other authors could be altered to draw an anti-aliased line based solely on the run parameters.

7.1.4 Ray Tracing

The use of higher-order structures in ray tracing a uniform spatial subdivision is relatively untreated. Due to the importance in ray tracing of efficiently calculating the point of closest intersection, it is intended to complete the work presented in higher dimensions. The increase of efficiency of using runs to ray trace a uniformly defined subdivision and the use of such techniques in tracing nonuniform subdivision indicates that the technique should be extendible to ray tracing subdivisions such as the quadtree and octree. A possibility that also bares promise is to develop a subdivision scheme based on runs.

Due to the problems of the combinatorial explosion of rays traced in the technique of distributed ray tracing, it will be important to develop an understanding of the sensitivity of $\mathcal{P}_\ell$ to perturbations of the run parameters. This could open the way to form distributions of rays or lines based solely on perturbating the run parameters. Possibilities may also exist to calculate the run parameters of the reflected and refracted rays in some cases directly from the parameters of the incident ray.

7.1.5 Higher-Dimensional Objects

We have completed the treatment of the structure of runs in the two-dimensional line. It should be possible to extend these results to higher dimensions [80, 125, 31, 53]. Due to the correspondence between the plane in higher dimensions and the line, it should also be possible to develop a technique for calculating the structure of the digitised plane. Higher dimensional objects are of particular interest to work in computational geometry. The major step will be from two dimensions to three as the scalar entity of slope will be replaced by the vector quantity of gradient. Therefore progression to dimensions higher than three should follow by extending the dimensionality of the gradient.

Appendix A

Matching Patterns of Runs

In this section we will be considering what is the minimum set of conditions needed to show the containment of a sequence of runs in another. We will consider two sequences of runs at order i, $R_0^{[i]}$ and $R_1^{[i]}$. It will be assumed that the two sequences start at the same lattice cell and that the run lengths and type of all orders previous to the order i are the same for the two sequences.

Each sequence of runs can be considered as a set of pixels or runs of order 0 and we can use results from set theory to show that one is or is not a subset of the other. The notation and properties of sets we will be using include

- If $R_0^{[i]} = R_1^{[i]}$ both sequences of runs contain identical sets of pixels.
- If $R_0^{[i]} \subset R_1^{[i]}$, the intersection patterns are equivalent over the length of $R_0^{[i]}$.
- If $R_0^{[i]} \subseteq R_1^{[i]}$, then $R_0^{[i]} \subset R_1^{[i]}$ or $R_0^{[i]} = R_1^{[i]}$.
- Where $\emptyset$, is the empty intersection pattern, $\emptyset \subset R_0^{[i]}$.
- If $R_0^{[i]}$ and $R_1^{[i]}$ have the same length, $R_0^{[i]} \subset R_1^{[i]}$ implies $R_1^{[i]} \subset R_0^{[i]}$ and therefore $R_0^{[i]} = R_1^{[i]}$.
- If $R_0^{[i]}$ and $R_1^{[i]}$ have the same length, $R_0^{[i]} \not\subset R_1^{[i]}$ implies $R_1^{[i]} \not\subset R_0^{[i]}$.
- If $R_0^{[i]} \subset R_1^{[i]}$ and $R_1^{[i]} \subset R_2^{[i]}$ then $R_0^{[i]} \subset R_2^{[i]}$.
- If $R_0^{[i]} \not\subset R_1^{[i]}$ and $R_1^{[i]} \subset R_2^{[i]}$ then $R_0^{[i]} \not\subset R_2^{[i]}$.

- If $R_0^{[i]} \subset R_1^{[i]}$ and $R_0^{[i]} \not\subset R_2^{[i]}$ then $R_1^{[i]} \not\subset R_2^{[i]}$.

If the run sequences, $R_0^{[i]}$ and $R_1^{[i]}$, start with a similar set of runs, these runs can be removed and the problem can be reduced to the question of containment of the ordered subset of runs that start with a run that differs. For example if $R_0^{[i]} = \{\ ls^r ls^{r+1} ls^r\ \}^{[[i]]}$ and $R_1^{[i]} = \{\ ls^r ls^r ls^{r+1}\ \}^{[[i]]}$, both sequences start with $\{\ ls^r ls^r\ \}^{[[i]]}$. The problem can be reduced to the question of containment of the sequences $R_0'^{[i]} = \{\ sls^r\ \}^{[[i]]}$ and $R_1'^{[i]} = \{\ ls^{r+1}\ \}^{[[i]]}$.

In the notation we have also defined the order of the runs in the sequences. This is because, to show containment of run sequences of orders above 1, we will be forming inductive proofs. For example let $R_0^{[i]} = \{\ sls\ \}^{[[i]]}$ and $R_1^{[i]} = \{\ ssl\ \}^{[[i]]}$. If the shape of the runs at order i is ls^+ and the length of a short run of order i is $r^{[i]}$, the question of containment can be asked of the run sequences $R_0^{[i-1]} = \{\ ls^{r^{[i]}} ls^{r^{[i]}+1} ls^{r^{[i]}}\ \}^{[[i-1]]}$ and $R_1^{[i-1]} = \{\ ls^{r^{[i]}} ls^{r^{[i]}} ls^{r^{[i]}+1}\ \}^{[[i-1]]}$. This in turn reduces to the question of containment of the subsequences $R_0'^{[i-1]} = \{\ sls^{r^{[i]}}\ \}^{[[i-1]]}$ and $R_1'^{[i-1]} = \{\ ls^{r^{[i]}+1}\ \}^{[[i-1]]}$.

Therefore we need concern ourselves only with run sequences that start with different runs lengths. From the following result, if the shorter run sequence has more than one run, it cannot be contained within the longer sequence. If it has only one run left, and that run is short, it is contained in the longer sequence. If it is long, it is contained in the longer sequence only if the type is 0.

Theorem A.0.1 *To show the containment of sequences of runs, six conditions are sufficient. Let $R^{[i]}$ be any sequence of runs composed of runs s and l.*

- $s \subset l \ldots$
- $l \subset s \ldots$ *iff type* $t^{[i]} = 0$
- $s \ldots \not\subset l \ldots$
- $l \ldots \not\subset s \ldots$
- $sR_0^{[i]} \subset sR_1^{[i]} \ldots$ *reduces to* $R_0^{[i]} \subset R_1^{[i]} \ldots$

- *$lR_0^{[i]} \subset lR_1^{[i]} \ldots$ reduces to $R_0^{[i]} \subset R_1^{[i]} \ldots$*

Proof : The proof is evident from Lemmas A.0.2–A.0.13. $\square$

Lemmas A.0.2–A.0.13, which we will present in the remainder of this appendix, contain all of the cases of which one or two runs could be contained in a longer run sequence (measured in pixels), where the two run sequences start with a different length run. The cases we will show are

$\{\ s\ \}^{[i]} \subset \{\ l\ \}^{[i]}$	
$\{\ l\ \}^{[i]} \subset \{\ ss\ \}^{[i]}$	$\{\ l\ \}^{[i]} \subset \{\ sl\ \}^{[i]}$ iff type $= 0$
$\{\ sl\ \}^{[i]} \not\subset \{\ ls\ \}^{[i]}$	
$\{\ ls\ \}^{[i]} \not\subset \{\ sl\ \}^{[i]}$	$\{\ sl\ \}^{[i]} \not\subset \{\ ll\ \}^{[i]}$
$\{\ ss\ \}^{[i]} \not\subset \{\ ls\ \}^{[i]}$	$\{\ ss\ \}^{[i]} \not\subset \{\ ll\ \}^{[i]}$
$\{\ ls\ \}^{[i]} \not\subset \{\ sss\ \}^{[i]}$	$\{\ ls\ \}^{[i]} \not\subset \{\ ssl\ \}^{[i]}$
$\{\ ll\ \}^{[i]} \not\subset \{\ sss\ \}^{[i]}$	$\{\ ll\ \}^{[i]} \not\subset \{\ ssl\ \}^{[i]}$
$\{\ ll\ \}^{[i]} \not\subset \{\ sls\ \}^{[i]}$	$\{\ ll\ \}^{[i]} \not\subset \{\ sll\ \}^{[i]}$

We start with the run sequences of length 2 that contain an equal number of pixels.

Lemma A.0.2 *For any order i, $\{\ sl\ \}^{[[i]]} \not\subset \{\ ls\ \}^{[[i]]}$ and $\{\ ls\ \}^{[[i]]} \not\subset \{\ sl\ \}^{[[i]]}$.*

Proof : Because the run sequences sl and ls have the same length, if $\{\ sl\ \}^{[[i]]} \not\subset \{\ ls\ \}^{[[i]]}$ then $\{\ ls\ \}^{[[i]]} \not\subset \{\ sl\ \}^{[[i]]}$. Therefore we only need to show only one of these cases.

For order 1, the lemma is obvious from the geometry of the runs of order 1. We define a run of order 1 by the position of its left–most pixel and its length (x, y, r). Where the length of a short run of order 1 is $r^{[1]}$, the sequence of runs $\{\ ls\ \}^{[[1]]}$ can be defined by $\{\ (0, 0, r^{[1]}+1), (r^{[1]}+2, 1, r^{[1]})\ \}^{[[0]]}$ and $\{\ sl\ \}^{[[1]]}$ by $\{\ (0, 0, r^{[1]}), (r^{[1]}+1, 1, r^{[1]}+1)\ \}^{[[0]]}$. It is apparent that the intersection patterns differ at x coordinates $x = \{r^{[1]}+1, r^{[1]}+2\}$. Therefore the lemma is shown for order 1.

For order 2, the two possible shapes are shape 1: ls^+ and shape 3: l^+s. We will consider the question of containment by looking at the equivalent sequences of runs at order 1. For example, for order 2 and shape 1 where the length of a short run is $r^{[2]}$, the sequence of runs $\{\ sl\ \}^{[[2]]}$ is equivalent to sequence of order 1 runs, $\mathbf{T}_{[1]} l s^{r^{[2]}-1} l s^{r^{[2]}}$.

Assume that $\{\ sl\ \}^{[[2]]} \subset \{\ ls\ \}^{[[2]]}$ given that $\{\ sl\ \}^{[[1]]} \not\subset \{\ ls\ \}^{[[1]]}$.

For shape 1: ls^+,	$\{\ sl\ \}^{[2]} \subset \{\ ls\ \}^{[2]}$
which implies	$\{\ ls^{r^{[2]}-1} l s^{r^{[2]}}\ \}^{[1]} \subset \{\ ls^{r^{[2]}} l s^{r^{[2]}-1}\ \}^{[1]}$
and reduces to	$\{\ \mathbf{ls}\ s^{r^{[2]}-1}\ \}^{[1]} \subset \{\ \mathbf{sl}\ s^{r^{[2]}-1}\ \}^{[1]}$
For shape 3: l^+s,	$\{\ sl\ \}^{[2]} \subset \{\ ls\ \}^{[2]}$
which implies	$\{\ l^{r^{[2]}-1} s l^{r^{[2]}} s\ \}^{[1]} \subset \{\ l^{r^{[2]}} s l^{r^{[2]}-1} s\ \}^{[1]}$
and reduces to	$\{\ \mathbf{sl}\ l^{r^{[2]}-1} s\ \}^{[1]} \subset \{\ \mathbf{ls}\ l^{r^{[2]}-1} s\ \}^{[1]}$

Both of these cases contradict our results at order 1.

For higher order runs, $i > 1$, let us look at the sequences of runs of order $i-1$ for each of the possible shapes. Assume that $\{\ sl\ \}^{[[i]]} \subset \{\ ls\ \}^{[[i]]}$ given that $\{\ sl\ \}^{[[i-1]]} \not\subset \{\ ls\ \}^{[[i-1]]}$.

For shape 0: sl^+,	$\{\ sl\ \}^{[i]} \subset \{\ ls\ \}^{[i]}$
which implies	$\{\ sl^{r^{[i]}-1} s l^{r^{[i]}}\ \}^{[i-1]} \subset \{\ sl^{r^{[i]}} s l^{r^{[i]}-1}\ \}^{[i-1]}$
and reduces to	$\{\ \mathbf{sl}\ l^{r^{[i]}-1}\ \}^{[i-1]} \subset \{\ \mathbf{ls}\ l^{r^{[i]}-1}\ \}^{[i-1]}$
For shape 1: ls^+,	$\{\ sl\ \}^{[i]} \subset \{\ ls\ \}^{[i]}$
which implies	$\{\ ls^{r^{[i]}-1} l s^{r^{[i]}}\ \}^{[i-1]} \subset \{\ ls^{r^{[i]}} l s^{r^{[i]}-1}\ \}^{[i-1]}$
and reduces to	$\{\ \mathbf{ls}\ s^{r^{[i]}-1}\ \}^{[i-1]} \subset \{\ \mathbf{sl}\ s^{r^{[i]}-1}\ \}^{[i-1]}$
For shape 2: s^+l,	$\{\ sl\ \}^{[i]} \subset \{\ ls\ \}^{[i]}$
which implies	$\{\ s^{r^{[i]}-1} l s^{r^{[i]}} l\ \}^{[i-1]} \subset \{\ s^{r^{[i]}} l s^{r^{[i]}-1} l\ \}^{[i-1]}$
and reduces to	$\{\ \mathbf{ls}\ s^{r^{[i]}-1} l\ \}^{[i-1]} \subset \{\ \mathbf{sl}\ s^{r^{[i]}-1} l\ \}^{[i-1]}$
For shape 3: l^+s,	$\{\ sl\ \}^{[i]} \subset \{\ ls\ \}^{[i]}$
which implies	$\{\ l^{r^{[i]}-1} s l^{r^{[i]}} s\ \}^{[i-1]} \subset \{\ l^{r^{[i]}} s l^{r^{[i]}-1} s\ \}^{[i-1]}$
and reduces to	$\{\ \mathbf{sl}\ l^{r^{[i]}-1} s\ \}^{[i-1]} \subset \{\ \mathbf{ls}\ l^{r^{[i]}-1} s\ \}^{[i-1]}$

All of these cases contradict our previous results. Therefore by induction the lemma is shown. $\square$

Lemma A.0.3 *For any order i, $\{\ s\ \}^{[[i]]} \subset \{\ l\ \}^{[[i]]}$.*

Proof : For order 1, the lemma is obvious from the geometry of the runs of order 1. To show that $\{\ s\ \}^{[[2]]} \subset \{\ l\ \}^{[[2]]}$, consider the two possible shapes at order 2.

For shape 1: ls^+,	$\{\ s\ \}^{[2]} \subset \{\ l\ \}^{[2]}$
which implies	$\{\ ls^{r^{[2]}-1}\ \}^{[1]} \subset \{\ ls^{r^{[2]}}\ \}^{[1]}$
and reduces to	$\{\ \mathbf{ls^{r^{[2]}-1}}\ \}^{[1]} \subset \{\ \mathbf{ls^{r^{[2]}-1}}\ s\ \}^{[1]}$

Therefore this case is shown.

For shape 3: l^+s,	$\{\ s\ \}^{[2]} \subset \{\ l\ \}^{[2]}$
which implies	$\{\ l^{r^{[2]}-1}s\ \}^{[1]} \subset \{\ l^{r^{[2]}}s\ \}^{[1]}$
and reduces to	$\{\ \mathbf{s}\ \}^{[1]} \subset \{\ \mathbf{l}\ s\ \}^{[1]}$

This case is shown, by induction.

For higher order runs, $i > 2$, for shape 0: sl^+, it is obvious that $\{\ \mathbf{sl^{r^{[i]}-1}}\ \}^{[[i]]} \subset \{\ \mathbf{sl^{r^{[i]}-1}}\ l\ \}^{[[i]]}$. Similarly for shape 1: ls^+, $\{\ \mathbf{ls^{r^{[i]}-1}}\ \}^{[[i]]} \subset \{\ \mathbf{ls^{r^{[i]}-1}}\ s\ \}^{[[i]]}$.

For shape 3: l^+s,	$\{\ s\ \}^{[i]} \subset \{\ l\ \}^{[i]}$
which implies	$\{\ l^{r^{[i]}-1}s\ \}^{[i-1]} \subset \{\ l^{r^{[i]}}s\ \}^{[i-1]}$
and reduces to	$\{\ \mathbf{s}\ \}^{[i-1]} \subset \{\ \mathbf{l}\ s\ \}^{[i-1]}$

By induction, this is shown.

For shape 2: s^+l,	$\{\ s\ \}^{[i]} \subset \{\ l\ \}^{[i]}$
which implies	$\{\ s^{r^{[i]}-1}l\ \}^{[i-1]} \subset \{\ s^{r^{[i]}}l\ \}^{[i-1]}$
and reduces to	$\{\ \mathbf{l}\ \}^{[i-1]} \subset \{\ \mathbf{sl}\ \}^{[i-1]}$

To answer the question of the containment of l in sl, we need to consider the runs of order $i-2$. For shape $s^{[i]} = 2$ we have $t^{[i]} = 1$ and $t^{[i-1]} = 0$. Therefore the possible shapes of the order $i-1$ runs are shapes 0 and 1.

For shape 1: ls^+,	$\{\ l\ \}^{[i-1]} \subset \{\ sl\ \}^{[i-1]}$
which implies	$\{\ ls^{r^{[i-1]}}\ \}^{[i-2]} \subset \{\ ls^{r^{[i-1]}-1}ls^{r^{[i-1]}}\ \}^{[i-2]}$
and reduces to	$\{\ \mathbf{s}\ \}^{[i-2]} \subset \{\ \mathbf{l}\ s^{r^{[i-1]}}\ \}^{[i-2]}$

By induction, this is shown.

For shape 0: sl^+,	$\{ \ l \ \}^{[i-1]} \subset \{ \ sl \ \}^{[i-1]}$
which implies	$\{ \ sl^{r^{[i-1]}} \ \}^{[i-2]} \subset \{ \ sl^{r^{[i-1]}-1} sl^{r^{[i-1]}} \ \}^{[i-2]}$
and reduces to	$\{ \ \mathbf{l} \ \}^{[i-2]} \subset \{ \ \mathbf{sl} \ l^{r^{[i-1]}-1} \ \}^{[i-2]}$

To answer this question, we need to consider order 2. For shape $s^{[3]} = 0$ we have $t^{[3]} = 0$ and $t^{[2]} = 0$ and therefore the shape at order 2 is $s^{[2]} = 1 : ls^+$.

For shape 1: ls^+,	$\{ \ l \ \}^{[2]} \subset \{ \ sl \ \}^{[2]}$
which implies	$\{ \ ls^{r^{[2]}} \ \}^{[1]} \subset \{ \ ls^{r^{[2]}-1} ls^{r^{[2]}} \ \}^{[1]}$
and reduces to	$\{ \ \mathbf{s} \ \}^{[1]} \subset \{ \ \mathbf{l} \ s^{r^{[2]}} \ \}^{[1]}$

Therefore the lemma is shown. □

Lemma A.0.4 *For any order i, $\{ \ sl \ \}^{[[i]]} \not\subset \{ \ ll \ \}^{[[i]]}$.*

Proof : By Lemma A.0.2 we have that $\{ \ sl \ \}^{[[i]]} \not\subset \{ \ ls \ \}^{[[i]]}$. By Lemma A.0.3 we have that $\{ \ s \ \}^{[[i]]} \subset \{ \ l \ \}^{[[i]]}$ therefore $\{ \ ls \ \}^{[[i]]} \subset \{ \ ll \ \}^{[[i]]}$. Therefore $\{ \ sl \ \}^{[[i]]} \not\subset \{ \ ll \ \}^{[[i]]}$. □

Lemma A.0.5 *For any order i, $\{ \ l \ \}^{[[i]]} \subset \{ \ ss \ \}^{[[i]]}$ and $\{ \ l \ \}^{[[i]]} \subset \{ \ sl \ \}^{[[i]]}$ iff $i > 1$ and $t^{[i]} = 0$.*

Proof : For order 1, the lemma is obvious from the geometry of the runs of order 1. For order 2, we will again consider the two possible shapes.

For shape 1: ls^+,	$\{ \ l \ \}^{[2]} \subset \{ \ ss) \ \}^{[2]}$
which implies	$\{ \ ls^{r^{[2]}} \ \}^{[1]} \subset \{ \ ls^{r^{[2]}-1} ls^{r^{[2]}}) \ \}^{[1]}$
and reduces to	$\{ \ \mathbf{s} \ \}^{[1]} \subset \{ \ \mathbf{l} \ s^{r^{[2]}}) \ \}^{[1]}$

From Lemma A.0.3 this case is shown. For shape 3, assume that $\{ \ l \ \}^{[[2]]} \subset \{ \ ss \ \}^{[[2]]}$.

For shape 3: l^+s,	$\{ \ l \ \}^{[2]} \subset \{ \ ss \ \}^{[2]}$
which implies	$\{ \ l^{r^{[2]}} s \ \}^{[1]} \subset \{ \ l^{r^{[2]}-1} s l^{r^{[2]}} s \ \}^{[1]}$
and reduces to	$\{ \ \mathbf{ls} \ \}^{[1]} \subset \{ \ \mathbf{sl} \ l^{r^{[2]}-1} s \ \}^{[1]}$

From Lemma A.0.2 this is not possible. Therefore, for order 2, $\{ \ l \ \}^{[[2]]} \not\subset \{ \ ss \ \}^{[[2]]}$ if $t^{[2]} = 1$. A similar result can be obtained for $\{ \ l \ \}^{[[i]]} \subset \{ \ sl \ \}^{[[i]]}$.

For order $i > 2$, consider the four possible cases. For shapes 3 and 2, assume $\{\ l\ \}^{[[i]]} \subset \{\ sl\ \}^{[[i]]}$.

For shape 3: l^+s,	$\{\ l\ \}^{[i]} \subset \{\ sl\ \}^{[i]}$
which implies	$\{\ l^{r^{[i]}}s\ \}^{[i-1]} \subset \{\ l^{r^{[i]}-1}sl^{r^{[i]}}s\ \}^{[i-1]}$
and reduces to	$\{\ \mathbf{ls}\ \}^{[i-1]} \subset \{\ \mathbf{sl}\ l^{r^{[i]}-1}s\ \}^{[i-1]}$

By Lemma A.0.2, this is not possible.

For shape 2: s^+l,	$\{\ l\ \}^{[i]} \subset \{\ sl\ \}^{[i]}$
which implies	$\{\ s^{r^{[i]}}l\ \}^{[i-1]} \subset \{\ s^{r^{[i]}-1}ls^{r^{[i]}}l\ \}^{[i-1]}$
and reduces to	$\{\ \mathbf{sl}\ \}^{[i-1]} \subset \{\ \mathbf{ls}\ s^{r^{[i]}-1}l\ \}^{[i-1]}$

By Lemma A.0.2, this is not possible.

For shape 1: ls^+,	$\{\ l\ \}^{[i]} \subset \{\ sl\ \}^{[i]}$
which implies	$\{\ ls^{r^{[i]}}\ \}^{[i-1]} \subset \{\ ls^{r^{[i]}-1}ls^{r^{[i]}}\ \}^{[i-1]}$
and reduces to	$\{\ \mathbf{s}\ \}^{[i-1]} \subset \{\ \mathbf{l}\ s^{r^{[i]}}\ \}^{[i-1]}$

By Lemma A.0.3 this case is shown.

For shape 0: sl^+,	$\{\ l\ \}^{[i]} \subset \{\ sl\ \}^{[i]}$
which implies	$\{\ sl^{r^{[i]}}\ \}^{[i-1]} \subset \{\ sl^{r^{[i]}-1}sl^{r^{[i]}}\ \}^{[i-1]}$
and reduces to	$\{\ \mathbf{l}\ \}^{[i-1]} \subset \{\ \mathbf{sl}\ l^{r^{[i]}-1}\ \}^{[i-1]}$

To answer the question of the containment of l in sl, we need to consider the runs of order $i-2$. For shape $s^{[i]} = 0$ we have $t^{[i]} = 0$ and $t^{[i-1]} = 0$. Therefore the possible shapes of the order $i-1$ runs are shapes 0 and 1.

For shape 1: ls^+,	$\{\ l\ \}^{[i-1]} \subset \{\ sl\ \}^{[i-1]}$
which implies	$\{\ ls^{r^{[i-1]}}\ \}^{[i-2]} \subset \{\ ls^{r^{[i-1]}-1}ls^{r^{[i-1]}}\ \}^{[i-2]}$
and reduces to	$\{\ \mathbf{s}\ \}^{[i-2]} \subset \{\ \mathbf{l}\ s^{r^{[i-1]}}\ \}^{[i-2]}$

By induction, this is shown.

For shape 0: sl^+,	$\{\ l\ \}^{[i-1]} \subset \{\ sl\ \}^{[i-1]}$
which implies	$\{\ sl^{r^{[i-1]}}\ \}^{[i-2]} \subset \{\ sl^{r^{[i-1]}-1}sl^{r^{[i-1]}}\ \}^{[i-2]}$
and reduces to	$\{\ \mathbf{l}\ \}^{[i-2]} \subset \{\ \mathbf{sl}\ l^{r^{[i-1]}-1}\ \}^{[i-2]}$

For shape $s^{[3]} = 0$ we have $t^{[3]} = 0$ and $t^{[2]} = 0$ and therefore the shape at order 2 is $s^{[2]} = 1 : ls^+$. By induction this case and the lemma are shown.

$\square$

Lemma A.0.6 *For any order i, $\{\ ss\ \}^{[[i]]} \not\subset \{\ ls\ \}^{[[i]]}$.*

Proof : For order 1, the lemma is obvious from the geometry of the runs of order 1. For order 2, assume that $\{\ ss\ \}^{[[2]]} \subset \{\ ls\ \}^{[[2]]}$.

For shape 1: ls^+,	$\{\ ss\ \}^{[2]} \subset \{\ ls\ \}^{[2]}$
which implies	$\{\ ls^{r^{[2]}-1} ls^{r^{[2]}-1}\ \}^{[1]} \subset \{\ ls^{r^{[2]}} ls^{r^{[2]}-1}\ \}^{[1]}$
and reduces to	$\{\ \mathbf{ls}\ s^{r^{[2]}-1}\ \}^{[1]} \subset \{\ \mathbf{sl}\ s^{r^{[2]}-1}\ \}^{[1]}$

By Lemma A.0.2 this case is shown by contradiction.

For shape 3: l^+s,	$\{\ ss\ \}^{[2]} \subset \{\ ls\ \}^{[2]}$
which implies	$\{\ l^{r^{[2]}-1} s l^{r^{[2]}-1} s\ \}^{[1]} \subset \{\ l^{r^{[2]}} s l^{r^{[2]}-1} s\ \}^{[1]}$
and reduces to	$\{\ \mathbf{sl}\ l^{r^{[2]}-1} s\ \}^{[1]} \subset \{\ \mathbf{ls}\ l^{r^{[2]}-1} s\ \}^{[1]}$

This case is shown by contradiction.

For higher order runs, $i > 1$, assume that $\{\ ss\ \}^{[[i]]} \subset \{\ ls\ \}^{[[i]]}$ and consider the sequences of runs of order $i-1$ for each of the possible shapes.

For shape 0: sl^+,	$\{\ ss\ \}^{[i]} \subset \{\ ls\ \}^{[i]}$
which implies	$\{\ sl^{r^{[i]}-1} sl^{r^{[i]}-1}\ \}^{[i-1]} \subset \{\ sl^{r^{[i]}} sl^{r^{[i]}-1}\ \}^{[i-1]}$
and reduces to	$\{\ \mathbf{sl}\ l^{r^{[i]}-1}\ \}^{[i-1]} \subset \{\ \mathbf{ls}\ l^{r^{[i]}-1}\ \}^{[i-1]}$

This case is shown by contradiction.

For shape 1: ls^+,	$\{\ ss\ \}^{[i]} \subset \{\ ls\ \}^{[i]}$
which implies	$\{\ ls^{r^{[i]}-1} ls^{r^{[i]}-1}\ \}^{[i-1]} \subset \{\ ls^{r^{[i]}} ls^{r^{[i]}-1}\ \}^{[i-1]}$
and reduces to	$\{\ \mathbf{ls}\ s^{r^{[i]}-1}\ \}^{[i-1]} \subset \{\ \mathbf{sl}\ s^{r^{[i]}-1}\ \}^{[i-1]}$

This case is shown by contradiction.

For shape 2: s^+l,	$\{\ ss\ \}^{[i]} \subset \{\ ls\ \}^{[i]}$
which implies	$\{\ s^{r^{[i]}-1} ls^{r^{[i]}-1} l\ \}^{[i-1]} \subset \{\ s^{r^{[i]}} ls^{r^{[i]}-1} l\ \}^{[i-1]}$
and reduces to	$\{\ \mathbf{ls}\ s^{r^{[i]}-1} l\ \}^{[i-1]} \subset \{\ \mathbf{sl}\ s^{r^{[i]}-1} l\ \}^{[i-1]}$

This case is shown by contradiction.

For shape 3: l^+s, $\{\ ss\ \}^{[i]} \subset \{\ ls\ \}^{[i]}$
which implies $\{\ l^{r^{[i]}-1}sl^{r^{[i]}-1}s\ \}^{[i-1]} \subset \{\ l^{r^{[i]}}sl^{r^{[i]}-1}s\ \}^{[i-1]}$
and reduces to $\{\ \mathbf{sl}\ l^{r^{[i]}-1}s\ \}^{[i-1]} \subset \{\ \mathbf{ls}\ l^{r^{[i]}-1}s\ \}^{[i-1]}$

By Lemma A.0.2 this is not possible and the lemma is shown. □

Lemma A.0.7 *For any order i, $\{\ ss\ \}^{[[i]]} \not\subset \{\ ll\ \}^{[[i]]}$.*

Proof : By Lemma A.0.6 we have that $\{\ ss\ \}^{[[i]]} \not\subset \{\ ls\ \}^{[[i]]}$. By Lemma A.0.3 we have that $\{\ s\ \}^{[[i]]} \subset \{\ l\ \}^{[[i]]}$ therefore $\{\ ls\ \}^{[[i]]} \subset \{\ ll\ \}^{[[i]]}$. Therefore $\{\ ss\ \}^{[[i]]} \not\subset \{\ ll\ \}^{[[i]]}$. □

Lemma A.0.8 *For any order i, $\{\ ls\ \}^{[[i]]} \not\subset \{\ sss\ \}^{[[i]]}$.*

Proof : From Lemma A.0.5 we know that $\{\ l\ \}^{[[i]]} \subset \{\ ss\ \}^{[[i]]}$ iff $i > 1$ and the type is 0. For all other cases the lemma is shown. We only need consider order $i > 1$ and the shapes possible when $t^{[i]} = 0$, shape 0: sl^+ and shape 1: ls^+.

For order 2, only shape 1 is possible. Assume that $\{\ ls\ \}^{[[i]]} \subset \{\ sss\ \}^{[[i]]}$.

For shape 1: ls^+, $\{\ ls\ \}^{[2]} \subset \{\ sss\ \}^{[2]}$
which implies $\{\ ls^{r^{[2]}}ls^{r^{[2]}-1}\ \}^{[1]} \subset \{\ ls^{r^{[2]}-1}ls^{r^{[2]}-1}ls^{r^{[2]}-1}\ \}^{[1]}$
and reduces to $\{\ \mathbf{sl}\ s^{r^{[2]}-1}\ \}^{[1]} \subset \{\ \mathbf{ls}\ s^{r^{[2]}-2}ls^{r^{[2]}-1}\ \}^{[1]}$

From Lemma A.0.2 we know that this is not possible by contradiction. Therefore $\{\ ls\ \}^{[[i]]} \not\subset \{\ sss\ \}^{[[i]]}$ for this case.

For higher order runs, $i > 1$, let us look at the sequences of runs of order $i-1$ for each of the possible shapes. Assume that $\{\ ls\ \}^{[[i]]} \subset \{\ sss\ \}^{[[i]]}$.

For shape 0: sl^+, $\{\ ls\ \}^{[i]} \subset \{\ sss\ \}^{[i]}$
which implies $\{\ sl^{r^{[i]}}sl^{r^{[i]}-1}\ \}^{[i-1]} \subset \{\ sl^{r^{[i]}-1}sl^{r^{[i]}-1}sl^{r^{[i]}-1}\ \}^{[i-1]}$
and reduces to $\{\ \mathbf{ls}\ l^{r^{[i]}-1}\ \}^{[i-1]} \subset \{\ \mathbf{sl}\ l^{r^{[i]}-2}sl^{r^{[i]}-1}\ \}^{[i-1]}$

This case is shown by contradiction.

For shape 1: ls^+,	$\{\ ls\ \}^{[i]} \subset \{\ sss\ \}^{[i]}$
which implies	$\{\ ls^{r^{[i]}} ls^{r^{[i]}-1}\ \}^{[i-1]} \subset \{\ ls^{r^{[i]}-1} ls^{r^{[i]}-1} ls^{r^{[i]}-1}\ \}^{[i-1]}$
and reduces to	$\{\ \mathbf{sl}\ s^{r^{[i]}-1}\ \}^{[i-1]} \subset \{\ \mathbf{ls}\ s^{r^{[i]}-2} ls^{r^{[i]}-1}\ \}^{[i-1]}$

This case is shown by contradiction and so is the lemma. □

Lemma A.0.9 *For any order i, $\{\ ls\ \}^{[[i]]} \not\subset \{\ ssl\ \}^{[[i]]}$.*

Proof : By Lemma A.0.8 we have that $\{\ ls\ \}^{[[i]]} \not\subset \{\ sss\ \}^{[[i]]}$. By Lemma A.0.3 we have that $\{\ s\ \}^{[[i]]} \subset \{\ l\ \}^{[[i]]}$ therefore $\{\ sss\ \}^{[[i]]} \subset \{\ ssl\ \}^{[[i]]}$. Therefore $\{\ ls\ \}^{[[i]]} \not\subset \{\ ssl\ \}^{[[i]]}$. □

Lemma A.0.10 *For any order i, $\{\ ll\ \}^{[[i]]} \not\subset \{\ sss\ \}^{[[i]]}$.*

Proof : By Lemma A.0.3 we have that $\{\ ls\ \}^{[[i]]} \subset \{\ ll\ \}^{[[i]]}$. By Lemma A.0.8 we have that $\{\ ls\ \}^{[[i]]} \not\subset \{\ sss\ \}^{[[i]]}$ therefore $\{\ ll\ \}^{[[i]]} \subset \{\ sss\ \}^{[[i]]}$. Therefore $\{\ ls\ \}^{[[i]]} \not\subset \{\ ssl\ \}^{[[i]]}$. □

Lemma A.0.11 *For any order i, $\{\ ll\ \}^{[[i]]} \not\subset \{\ ssl\ \}^{[[i]]}$.*

Proof : By Lemma A.0.10 we have that $\{\ ll\ \}^{[[i]]} \not\subset \{\ sss\ \}^{[[i]]}$. By Lemma A.0.3 we have that $\{\ s\ \}^{[[i]]} \subset \{\ l\ \}^{[[i]]}$ therefore $\{\ sss\ \}^{[[i]]} \subset \{\ ssl\ \}^{[[i]]}$. Therefore $\{\ ll\ \}^{[[i]]} \not\subset \{\ ssl\ \}^{[[i]]}$. □

Lemma A.0.12 *For any order i, $\{\ ll\ \}^{[[i]]} \not\subset \{\ sls\ \}^{[[i]]}$.*

Proof : From Lemma A.0.5 we know that $\{\ l\ \}^{[[i]]} \subset \{\ sl\ \}^{[[i]]}$ iff $i > 1$ and the type is 0. For all other cases the lemma is shown. We only need consider order $i > 1$ and the shapes possible when $t^{[i]} = 0$, shape 0: sl^+ and shape 1: ls^+.

For order 2, only shape 1 is possible. Assume that $\{\ ls\ \}^{[[i]]} \subset \{\ sss\ \}^{[[i]]}$.

For shape 1: ls^+,	$\{\ ll\ \}^{[2]} \subset \{\ sls\ \}^{[2]}$
which implies	$\{\ ls^{r^{[2]}} ls^{r^{[2]}}\ \}^{[1]} \subset \{\ ls^{r^{[2]}-1} ls^{r^{[2]}} ls^{r^{[2]}-1}\ \}^{[1]}$
and reduces to	$\{\ \mathbf{sl}\ s^{r^{[2]}}\ \}^{[1]} \subset \{\ \mathbf{ls}\ s^{r^{[2]}-1} ls^{r^{[2]}-1}\ \}^{[1]}$

From Lemma A.0.2 we know that this is not possible therefore $\{\ ls\ \}^{[[i]]} \not\subset \{\ sss\ \}^{[[i]]}$ for this case by contradiction.

For higher order runs, $i > 1$, let us look at the sequences of runs of order $i-1$ for each of the possible shapes. Assume that $\{\ ls\ \}^{[[i]]} \subset \{\ sss\ \}^{[[i]]}$.

For shape 0: sl^+,	$\{\ ll\ \}^{[i]} \subset \{\ sls\ \}^{[i]}$
which implies	$\{\ sl^{r^{[i]}}sl^{r^{[i]}}\ \}^{[i-1]} \subset \{\ sl^{r^{[i]}-1}sl^{r^{[i]}}sl^{r^{[i]}-1}\ \}^{[i-1]}$
and reduces to	$\{\ \mathbf{ls}\ l^{r^{[i]}}\ \}^{[i-1]} \subset \{\ \mathbf{sl}\ l^{r^{[i]}-1}sl^{r^{[i]}-1}\ \}^{[i-1]}$

This case is shown by contradiction.

For shape 1: ls^+,	$\{\ ll\ \}^{[i]} \subset \{\ sls\ \}^{[i]}$
which implies	$\{\ ls^{r^{[i]}}ls^{r^{[i]}}\ \}^{[i-1]} \subset \{\ ls^{r^{[i]}-1}ls^{r^{[i]}}ls^{r^{[i]}-1}\ \}^{[i-1]}$
and reduces to	$\{\ \mathbf{sl}\ s^{r^{[i]}}\ \}^{[i-1]} \subset \{\ \mathbf{ls}\ s^{r^{[i]}-1}ls^{r^{[i]}-1}\ \}^{[i-1]}$

This case is shown by contradiction and so is the lemma. □

Lemma A.0.13 *For any order i, $\{\ ll\ \}^{[[i]]} \not\subset \{\ sll\ \}^{[[i]]}$.*

Proof : By Lemma A.0.12 we have that $\{\ ll\ \}^{[[i]]} \not\subset \{\ \ \}^{[[i]]}$. By Lemma A.0.3 we have that $\{\ s\ \}^{[[i]]} \subset \{\ l\ \}^{[[i]]}$ therefore $\{\ sls\ \}^{[[i]]} \subset \{\ sll\ \}^{[[i]]}$. Therefore $\{\ ll\ \}^{[[i]]} \not\subset \{\ sll\ \}^{[[i]]}$. □

Appendix B

The Correspondance between the Run-Based and Bresenham's Line Drawing Algorithms.

B.1 Bresenham's Line Drawing Algorithm

In Chapter 5 we made some additions to our definition of the run based structure of the intersection pattern at order 0 to define an order 0 run-based line drawing algorithm. These additional definitions were

$$m^{[0]} = p^{[1]} = a \tag{B.1}$$

$$n^{[0]} = p^{[0]} - m^{[0]} = d - a \tag{B.2}$$

$$b_0^{[0]} = b - n^{[0]} \tag{B.3}$$

The value $n^{[0]}$ is subtracted from the initial numerator value to align the decision variable to zero. Therefore if $b_j^{[0]} < 0$ the next pixel is $(x+1, y+1)$, if not, the next pixel is $(x+1, y)$.

Algorithm B.1 The Order 0 Line Drawing Algorithm.

$\Delta x = x_1 - x_0$
$\Delta y = y_1 - y_0$

```
m^{[0]} = 2Δy
n^{[0]} = 2(Δx − Δy)
b_0^{[0]} = b − n^{[0]}

setPixel(x, y)
while x < x_1
    x = x + 1
    if b_j^{[0]} < 0
        y = y + 1
        b_{j+1}^{[0]} = b_j^{[0]} + m^{[0]}
    else
        b_{j+1}^{[0]} = b_j^{[0]} − n^{[0]}
    setPixel(x, y)
```

The algorithm is identical to Bresenham's line drawing algorithm [15], except for some sign changes. Bresenham's algorithm is detailed in Algorithm B.2.

Algorithm B.2 Bresenham's Line Drawing Algorithm

```
Δa = x_1 − x_0
Δb = y_1 − y_0
∇_1 = 2Δb − Δa
Δ_0 = 2(Δb − Δa)
Δ_1 = 2Δb
x = x_0;
y = y_0;

setPixel(x, y)
while x < x_1
    x = x + 1
    if ∇_i < 0
        ∇_{i+1} = ∇_i + Δ_0
    else
        ∇_{i+1} = ∇_i + Δ_1
        y = y + 1
    setPixel(x, y)
```

B.2 Bresenham's Run Length Slice Algorithm

While the order 0 run-based algorithm and Bresenham's original line drawing algorithm are identical, the run length slice algorithm shares properties of the run-based algorithms of orders 1 and 2.

Algorithm B.3 The Run Length Slice Line Drawing Algorithm

```
ΔA = x_1 − x_0
ΔB = y_1 − y_0
∇B = ΔA − ΔB
Q = ⌊ΔA / ∇B⌋
R = ∇B − QΔA
M = ⌊Q / 2⌋

if Q is odd
    N = R + ∇B
else
    N = R

H_0 = M
if N = 0
    H_b = M − 1
else
    H_b = M

x = 0
y = 0
Δ_0 = 2R
Δ_1 = 2(R − ∇B)
∇_1 = N + Δ_1
count = ∇B − 1

{Slope < 1/2}
if ΔB < ∇B
    {Output initial run length}
    while H_O > 0
        setPixel(x, y)
        x = x + 1
        H_0 = H_0 − 1
    y = y + 1

    {Intermediate run length loop}
    while count > 0
        if ∇_i < 0
            H_i = Q − 1
            ∇_{i+1} = ∇_i + Δ_0
        else
            H_i = Q
            ∇_{i+1} = ∇_i + Δ_1
        {Output run length}
        while H_i > 0
            setPixel(x, y)
```

```
            $x = x + 1$
            $H_i = H_i - 1$
          $y = y + 1$
        $count = count - 1$

    {Output final run length}
    while $H_b > 0$
        $setPixel(x, y)$
        $x = x + 1$
        $H_b = H_b - 1$

{Slope $> \frac{1}{2}$}
else
    {Output initial run slice}
    while $H_O > 0$
        $setPixel(x, y)$
        $x = x + 1$
        $y = y + 1$
        $H_0 = H_0 - 1$
    $setPixel(x, y)$
    $x = x + 1$

    {Intermediate run length loop}
    while $count > 0$
        if $\nabla_i < 0$
            $H_i = Q - 1$
            $\nabla_{i+1} = \nabla_i + \Delta_0$
        else
            $H_i = Q$
            $\nabla_{i+1} = \nabla_i + \Delta_1$
        {Output run slice}
        while $H_i > 0$
            $setPixel(x, y)$
            $x = x + 1$
            $y = y + 1$
            $H_i = H_i - 1$
        $setPixel(x, y)$
        $x = x + 1$
        $count = count - 1$

    {Output final run length}
    while $H_b > 0$
        $setPixel(x, y)$
        $x = x + 1$
        $y = y + 1$
        $H_b = H_b - 1$
    $setPixel(x, y)$
```

Bresenham's definition of the run length slice was based around the slope of $\frac{1}{2}$. From his results this condition was defined as $\Delta A < 2\nabla B$ or $d < 2(d-a)$ which is equivalent to $a < d-a$ or $\Delta B < \nabla B$.

For the slope $\frac{a}{d} < \frac{1}{2}$ Bresenham's run length slice algorithm is identical to the run-based algorithm of order 1 except for some sign changes and a change in the calculation of the initial and final run lengths. To illustrate our case we define the following values

$$\begin{aligned}
p^{[0]}/2 &= \Delta A \\
p^{[1]}/2 &= \Delta B \\
p'^{[1]} &= \nabla B \\
r'^{[1]} &= Q \\
n'^{[1]} &= R \\
m'^{[1]} &= \nabla B - R \\
r_0'^{[1]} &= H_0, M \\
r_p'^{[1]} &= H_b
\end{aligned}$$

It should be noted that in the algorithms we have presented we have $\gcd(p^{[0]}, p^{[1]}) = 2$. This was to ensure that b is an integer and therefore at a numerator value of zero in the numerator sequence, a corner intersection occurs. This point equates to when there are two pixels that are equally close to the original line. If the parameters are defined such that $\gcd(p^{[0]}, p^{[1]}) = 1$, we have $b_j^{new[i]} = \left\lfloor \frac{b_j^{[i]}}{2} \right\rfloor$, there is no relative change in the numerator values and the algorithm remains valid.

When the slope $\frac{a}{d} < \frac{1}{2}$ we have $p^{[1]} < p^{[0]} - p^{[1]}$ and therefore $p'^{[1]} = p^{[1]}/2$, $n'^{[1]} = n^{[1]}/2$, $m'^{[1]} = m^{[1]}/2$, $r'^{[1]} = r^{[1]}$, $r_0'^{[1]} = r_0^{[1]} - 1$ and $r_p'^{[1]} = r_p^{[1]} - 1$.

The initial value of the decision variable in the run length slice algorithm is ∇_1. If $r'^{[1]} = Q$ is odd we have

$$\begin{aligned}
N &= \nabla B + R \\
&= p'^{[1]} + n'^{[1]} \\
\nabla_1 &= N + 2R - 2\nabla B
\end{aligned}$$

$$
\begin{aligned}
&= p'^{[1]} + n'^{[1]} - 2m'^{[1]} \\
&= 2n'^{[1]} - m'^{[1]} && \text{(B.4)}
\end{aligned}
$$

If $r'^{[1]}$ is even we have

$$
\begin{aligned}
N &= R \\
&= n'^{[1]} \\
\nabla_1 &= N + 2R - 2\nabla B \\
&= n'^{[1]} - 2m'^{[1]} && \text{(B.5)}
\end{aligned}
$$

For setting the intermediate run lengths, for the order 1 algorithm, we have if $r^{[i]}$ is odd,

$$
\begin{aligned}
b_1^{[1]} &= \frac{m^{[1]}}{2} - n^{[1]} \\
&= m'^{[1]} - 2n'^{[1]} \\
&= -\nabla_1 && \text{(B.6)}
\end{aligned}
$$

If $r^{[i]}$ is even,

$$
\begin{aligned}
b_1^{[1]} &= m^{[1]} + \frac{n^{[1]}}{2} \\
&= 2m'^{[1]} + n'^{[1]} \\
&= -\nabla_1 && \text{(B.7)}
\end{aligned}
$$

When the slope $\frac{a}{d} > \frac{2}{3}$ the run slice algorithm is identical to the run-based algorithm of order 2 of type 1. If the slope $\frac{a}{d} > \frac{1}{2}$, we have $p^{[1]} > p^{[0]} - p^{[1]}$ and $p'^{[1]} = (p^{[0]} - p^{[1]})/2$. We also know that $r^{[1]} = 1$ and therefore $n^{[1]} = p^{[0]} - p^{[1]}$. For the run slice algorithm to behave like the run-based algorithm of order 2 we need to find the situation when

$$
r'^{[1]} = r^{[2]} = \left\lfloor \frac{p^{[1]}}{p^{[2]}} \right\rfloor = \left\lfloor \frac{p^{[1]}/2}{p'^{[1]}} \right\rfloor = \left\lfloor \frac{p^{[1]}}{p^{[0]} - p^{[1]}} \right\rfloor = \left\lfloor \frac{p^{[1]}}{n^{[1]}} \right\rfloor
$$

If order 2 type is 1 we have $p^{[2]} = n^{[1]}$ and

$$
\begin{aligned}
n^{[1]} &< m^{[1]} \\
p^{[0]} - p^{[1]} &< p^{[1]} - (p^{[0]} - p^{[1]}) \\
\frac{p^{[1]}}{p^{[0]}} &> \frac{2}{3}
\end{aligned}
$$

Therefore the run slice algorithm will behave exactly as the order 2 type 1 run based algorithm when the slope of the line $\frac{a}{d} > \frac{2}{3}$.

When the slope $\frac{1}{2} < \frac{a}{d} < \frac{2}{3}$ to show the order 2 type is 0 we have $r^{[1]} = 1$ and

$$\begin{aligned} n^{[1]} &> m^{[1]} \\ p^{[0]} - p^{[1]} &> p^{[1]} - (p^{[0]} - p^{[1]}) \\ \frac{p^{[1]}}{p^{[0]}} &< \frac{2}{3} \end{aligned}$$

In this situation the number of long runs of order 1 outnumber the short runs. Therefore the value of $r'^{[1]} = Q = 1$. In this case Bresenham sets the contiguous runs of length 2 separately except for the last. The last single run of length 2 and the intermediate run of length 1 are set together. Therefore the run slice algorithm in this range of the slope has a behaviour between the run-based algorithms of orders 1 and 2. If there are a lot more long runs in the line, the behaviour of the algorithm is more like order 1, as the long runs are set individually. The more short runs in the line, the closer the algorithm approaches the order 2 algorithm, as sets of short and long runs are set together. Therefore the run length slice algorithm has an identical behaviour to the the order 1 algorithm when the line consists of all runs of length 2 or the slope is $\frac{1}{2}$. The algorithm behaves exactly as the order 2 algorithm when the line consists of an equal number of runs of length 1 and 2 or the slope is $\frac{2}{3}$.

Dividing the run length slice algorithm into two sections based on the slope of $\frac{1}{2}$ allows us to use the primitives we defined for the run-based algorithms.

Algorithm B.4 The Adapted Run Length Slice Line Drawing Algorithm

$p_0 = p^{[0]}/2 = x_1 - x_0$
$p_1 = p^{[1]}/2 = y_1 - y_0$
$p'^{[1]} = \min(p_1, p_0 - p_1)$
$r'^{[1]} = \left\lfloor \frac{p_0}{p'^{[1]}} \right\rfloor$
$n'^{[1]} = p_0 - r'^{[1]} p'^{[1]}$
$m'^{[1]} = p'^{[1]} - n'^{[1]}$
$r_p'^{[0]} = r_0'^{[1]} = \left\lfloor \frac{r'^{[1]}}{2} \right\rfloor$
$count = p'^{[1]}$
$x = x_0$
$y = y_0$

$\Delta_0 = 2n'^{[1]}$
$\Delta_1 = -2m'[1]$
if $r'^{[1]}$ is odd
    $\nabla_1 = \Delta_0 - m'^{[1]}$
else
    $\nabla_1 = n'^{[1]} + \Delta_1$

if $p_1 \leq p_0 - p_1$ {The slope $\leq \frac{1}{2}$}
    $r_0'^{[1]} = r_0'^{[1]} + 1$
    $r_p'^{[1]} = r_p'^{[1]} + 1$
    $setRunsOrder1(x, y, r_0'^{[1]})$
    $x = x + r_0'^{[1]}$
    $y = y + 1$
    while $(count = count - 1) > 0$
        if $\nabla_i < 0$
            $setRunOrder1(x, y, r'^{[1]})$
            $x = x + r'^{[1]}$
            $y = y + 1$
            $\nabla_{i+1} = \nabla_i + \Delta_0$
        else
            $setRunOrder1(x, y, (r'^{[1]} + 1))$
            $x = x + (r'^{[1]} + 1)$
            $y = y + 1$
            $\nabla_{i+1} = \nabla_i + \Delta_1$
        $setRunOrder1(x, y, r_p'^{[1]})$

else if $r'^{[1]} == 2$ {The slope $< \frac{2}{3}$}
    $setPixel(x, y)$
    $x = x + 1$
    $y = y + 1$

    while $(count = count - 1) > 0$
        if $\nabla_i < 0$
            $setRunOrder1(x, y, 2)$
            $x = x + 2$
            $y = y + 1$
            $\nabla_{i+1} = \nabla_i + \Delta_0$
        else
            $setPixel(x, y)$
            $x = x + 1$
            $y = y + 1$
            $setRunOrder1(x, y, 2)$
            $x = x + 2$
            $y = y + 1$
            $\nabla_{i+1} = \nabla_i + \Delta_1$
        $setPixel(x, y)$

else {The slope $> \frac{2}{3}$}

$setRunOrder1(x, y, 2)$

$x = x + 2$

$y = y + 1$

while $(count = count - 1) > 0$

    if $\nabla_i < 0$

        $setRunOrder2 - Shape3(x, y, r'^{[1]})$

        $x = x + r'^{[1]} + 1$

        $y = y + r'^{[1]}$

        $\nabla_{i+1} = \nabla_i + \Delta_0$

    else

        $setRunOrder2 - Shape3(x, y, r'^{[1]} + 1)$

        $x = x + r'^{[1]} + 2$

        $y = y + r'^{[1]} + 1$

        $\nabla_{i+1} = \nabla_i + \Delta_1$

    $setRunOrder2 - Shape3(x, y, r'^{[1]}_p)$

B.3 Analysis

B.3.1 Run Length Slice Algorithm

Table B.1: Operator counts for the run length slice algorithm.

Operator	Counts
Assignment	$4p'^{[1]} + 14$
Additive	$4p'^{[1]} + 5.5$
Comparison	$2p'^{[1]}$
Memory	$1.5p'^{[1]} + 2.5$
Division	1
Multiplication	1
Shift	3
Bitwise logical	1

B.3.2 Adapted Run Length Slice Algorithm

Table B.2: Operator counts for the adapted run length slice algorithm.

Operator	Counts
Assignment	$4p'^{[1]} + 13.5$
Additive	$4p'^{[1]} + 7$
Comparison	$2p'^{[1]} - 1$
Memory	$p'^{[1]} + 1$
Division	1
Multiplication	1
Shift	3
Bitwise logical	1

Appendix C

Summary.

Run length	$r^{[i]} = \left\lfloor \frac{p^{[i-1]}}{p^{[i]}} \right\rfloor$	
Structural parameters	$p^{[0]} = d$	
	$p^{[1]} = a$	
	$p^{[i]} = m^{[i]} + n^{[i]}$	$= \min(m^{[i-1]} + n^{[i-1]})$
Order 1	$n^{[1]} \equiv p^{[0]} \bmod p^{[1]}$	$= p^{[0]} - r^{[1]}p^{[1]}$
	$m^{[1]} = p^{[1]} - n^{[1]}$	$= (r^{[1]} + 1)p^{[1]} - p^{[0]}$
Order $i > 1$, type $t^{[i]} = 0$	$p^{[i]} = n^{[i-1]} < m^{[i-1]}$	
	$m^{[i]} = p^{[i-1]} \bmod p^{[i]}$	$= p^{[i-1]} - r^{[i]}p^{[i]}$
	$n^{[i]} = p^{[i]} - m^{[i]}$	$= (r^{[i]} + 1)p^{[i]} - p^{[i-1]}$
Order $i > 1$, type $t^{[i]} = 1$	$p^{[i]} = m^{[i-1]} < n^{[i-1]}$	
	$n^{[i]} = p^{[i-1]} \bmod p^{[i]}$	$= p^{[i-1]} - r^{[i]}p^{[i]}$
	$m^{[i]} = p^{[i]} - n^{[i]}$	$= (r^{[i]} + 1)p^{[i]} - p^{[i-1]}$
Type and Shape	$s^{[i]} = 0$: sl^+	if $t^{[i]} = 0$ and $t^{[i-1]} = 0$
	$s^{[i]} = 1$: ls^+	if $t^{[i]} = 0$ and $t^{[i-1]} = 1$
	$s^{[i]} = 2$: s^+l	if $t^{[i]} = 1$ and $t^{[i-1]} = 0$
	$s^{[i]} = 3$: l^+s	if $t^{[i]} = 1$ and $t^{[i-1]} = 1$

Calculation of the Numerator Sequence

$$\begin{aligned} b_j^{[i]} &\equiv b_{j-1}^{[i]} + m^{[i]} \equiv b_0^{[i]} + jm^{[i]} \pmod{p^{[i]}} \\ &\equiv b_{j-1}^{[i]} - n^{[i]} \equiv b_0^{[i]} - jn^{[i]} \pmod{p^{[i]}} \end{aligned}$$

For order 1
$$\begin{aligned} b_{j+1}^{[1]} &= b_j^{[1]} - n^{[1]} && \text{if the } j\text{-th run is short} \\ &= b_j^{[1]} + m^{[1]} && \text{if the } j\text{-th run is long} \end{aligned}$$

Order $i > 1$, type $t^{[i]} = 0$
$$\begin{aligned} b_{j+1}^{[i]} &= b_j^{[i]} + m^{[i]} && \text{if the } j\text{-th run is short} \\ &= b_j^{[i]} - n^{[i]} && \text{if the } j\text{-th run is long} \end{aligned}$$

Order $i > 1$, type $t^{[i]} = 1$
$$\begin{aligned} b_{j+1}^{[i]} &= b_j^{[i]} - n^{[i]} && \text{if the } j\text{-th run is short} \\ &= b_j^{[i]} + m^{[i]} && \text{if the } j\text{-th run is long} \end{aligned}$$

Generation of the Intersection Pattern

For order 1
$$\begin{aligned} r_j^{[1]} &= r^{[1]} && \text{if } b_j^{[1]} > b_{j+1}^{[1]} \\ &&& \text{or } b_j^{[1]} \geq n^{[1]} \\ &&& \text{or } b_{j+1}^{[1]} < m^{[1]} \\ &= r^{[1]} + 1 && \text{if } b_j^{[1]} < b_{j+1}^{[1]} \\ &&& \text{or } b_j^{[1]} < n^{[1]} \\ &&& \text{or } b_{j+1}^{[1]} \geq m^{[1]} \end{aligned}$$

Order $i > 1$, type $t^{[i]} = 0$
$$\begin{aligned} r_j^{[i]} &= r^{[i]} && \text{if } b_j^{[i]} < b_{j+1}^{[i]} \\ &&& \text{or } b_j^{[i]} < n^{[i]} \\ &&& \text{or } b_{j+1}^{[i]} \geq m^{[i]} \\ &= r^{[i]} + 1 && \text{if } b_j^{[i]} > b_{j+1}^{[i]} \\ &&& \text{or } b_j^{[i]} \geq n^{[i]} \\ &&& \text{or } b_{j+1}^{[i]} < m^{[i]} \end{aligned}$$

Order $i > 1$, type $t^{[i]} = 1$
$$\begin{aligned} r_j^{[i]} &= r^{[i]} && \text{if } b_j^{[i]} > b_{j+1}^{[i]} \\ &&& \text{or } b_j^{[i]} \geq n^{[i]} \\ &&& \text{or } b_{j+1}^{[i]} < m^{[i]} \\ &= r^{[i]} + 1 && \text{if } b_j^{[i]} < b_{j+1}^{[i]} \\ &&& \text{or } b_j^{[i]} < n^{[i]} \\ &&& \text{or } b_{j+1}^{[i]} \geq m^{[i]} \end{aligned}$$

Initial run length

For order $i > 0$ $\qquad r_0^{[i]} = 0$ if $b_0^{[i-1]} < p^{[i]}$

For order 1, $\qquad r_0^{[i]} = \left\lceil \frac{p^{[i-1]} - b_0^{[i-1]}}{p^{[i]}} \right\rceil$

For order $i > 1$, type $t^{[i]} = 0$ $\qquad r_0^{[i]} = \left\lfloor \frac{b_0^{[i-1]}}{p^{[i]}} \right\rfloor$

For order $i > 1$, type $t^{[i]} = 1$ $\qquad r_0^{[i]} = \left\lceil \frac{p^{[i-1]} - b_0^{[i-1]}}{p^{[i]}} \right\rceil$

Initial partial numerator value

For order 1 $\qquad b_0^{[1]} \equiv b_0^{[0]} + p^{[1]} r_0^{[1]} \pmod{p^{[0]}}$

For order 1, $a \geq d - a$ $\qquad b_0^{[1]} = b_0^{[0]} - r_0^{[1]} p^{[1]}$

For order 1, $a < d - a$ $\qquad b_0^{[1]} = b_0^{[0]} + r_0^{[1]} p^{[1]} - p^{[0]}$

For order $i > 1$

$$\begin{aligned} b_0^{[i]} &\equiv b_0^{[i-1]} + m^{[i-1]} r_0^{[i]} \pmod{p^{[i-1]}} \\ &\equiv b_0^{[i-1]} - n^{[i-1]} r_0^{[i]} \pmod{p^{[i-1]}} \end{aligned}$$

Order $i > 1$, type 0 $\qquad b_0^{[i]} = b_0^{[i-1]} - r_0^{[i]} p^{[i]}$

Order $i > 1$, type 1 $\qquad b_0^{[i]} = b_0^{[i-1]} + r_0^{[i]} p^{[i]} - p^{[i-1]}$

Final partial run length

Order 1

$$\begin{aligned} r_p^{[1]} &= r^{[1]} - r_0^{[i]} - 1 && \text{if } b_{p^{[i]}-1}^{[1]} > b_0^{[1]} \\ &= r^{[1]} - r_0^{[i]} && \text{if } b_{p^{[i]}-1}^{[1]} < b_0^{[1]} \end{aligned}$$

Order $i > 1$, type 0

$$\begin{aligned} r_p^{[i]} &= r^{[i]} - r_0^{[i]} - 1 && \text{if } b_{p^{[i]}-1}^{[i]} < b_0^{[i]} \\ &= r^{[i]} - r_0^{[i]} && \text{if } b_{p^{[i]}-1}^{[i]} > b_0^{[i]} \end{aligned}$$

Order $i > 1$, type 1

$$\begin{aligned} r_p^{[i]} &= r^{[i]} - r_0^{[i]} - 1 && \text{if } b_{p^{[i]}-1}^{[i]} > b_0^{[i]} \\ &= r^{[i]} - r_0^{[i]} && \text{if } b_{p^{[i]}-1}^{[i]} < b_0^{[i]} \end{aligned}$$

Appendix D

Glossary.

We concentrate of those symbols used repeatedly without immediate definition.

$\mathcal{R}$ The set of real numbers.

$\mathcal{Z}$ The set of integers.

$\mathcal{N}$ The set of natural numbers, $1, 2, \ldots$.

$\lfloor\ \rfloor$ The floor function, $a = \lfloor a + \rho \rfloor$ where $a \in \mathcal{Z}$ and $0 \leq \rho < 1$.

$\lceil\ \rceil$ The ceiling function, $a + 1 = \lceil a + \rho \rceil$ where $a \in \mathcal{Z}$ and $0 < \rho \leq 1$.

$\subset$ The subset function.

$\subseteq$ The subset or equality function.

gcd( ) The greatest common divisor of the numbers presented.

min( ) The minimum value of the numbers presented.

ℓ Used to denote a continuous line.

$\mathcal{L}$ The lattice, defined as the integer domain $\mathcal{Z} \times \mathcal{Z}$.

$\mathcal{L}_{(x,y)}$ A bounded frame of the lattice from $(0,0)$ to (x,y).

$\mathcal{P}_\ell$ Used to denote the intersection pattern of the line ℓ and the lattice.

c_j The j-th element in the chain code of the digitised line.

α The real or irrational slope of a line.

β The real or irrational intercept of a line.

a The numerator of the rational slope of a line.

b The numerator of the rational intercept of a line.

d The denominator of the rational slope or intercept of a line.

g Typically used to represent the greatest common divisor of a and d unless otherwise defined.

h, j, k Positive integers.

u, v, w, x, y, z Positive integers used in Chapter 4.

u A rational number. Typically $0 \leq u < 1$

ρ An irrational number. Typically $0 \leq \rho < 1$.

P_n The n-th partial numerator of a continued fraction.

Q_n The n-th partial denominator of a continued fraction.

f_x, f_y Positive integers used to define frame size.

R The set of pixels defined by a sequence of runs.

L A set of lines.

i The order of a run, run parameter, or run–based construct. Defines the level of the run within the hierarchy of the run structure of the intersection pattern.

i_{max} The maximum order of the run structure of an intersection pattern.

s A symbol representing a short run of order i.

l A symbol representing a long run of order i.

$b_j^{[i]}$ The j-th value of the numerator sequence.

$c^{[i]}$ The composition of a run of order i. Calculated from the values of $t^{[i-1]}$ and $t^{[i]}$.

$m^{[i]}$ The number of long or short runs of order i in the intersection pattern depending on type.

$n^{[i]}$ The number of long or short runs of order i in the intersection pattern depending on type.

$p^{[i]}$ The total number of runs of order i.

$t^{[i]}$ The type of order i. Calculated from the values of $m^{[i-1]}$ and $n^{[i-1]}$. Only defined for order $i > 1$.

$r^{[i]}$ The length of a short run of order i.

$r_j^{[i]}$ The length of the j-th run of order i in the intersection pattern.

$s^{[i]}$ The shape of a run of order i. Calculated from the values of $t^{[i-1]}$ and $t^{[i]}$.

$u^{[i]}$ The number of long runs of order i in the intersection pattern.

$v^{[i]}$ The number of short runs of order i in the intersection pattern.

$x^{[i]}$ The x-coordinate position of a run of order i.

$y^{[i]}$ The y-coordinate position of a run of order i.

Bibliography

[1] J. Amanatides and A. Woo. A fast voxel traversal algorithm for ray tracing. *EUROGRAPHICS '87 Amsterdam*, pages 3–10, August 1987.

[2] T. A. Anderson and C. E. Kim. Representation of digital line segments and their preimages. In *Proceedings of the 7th International Conference on Pattern Recognition*, IEEE Publ. 84CH2046-1, pages 501–504. IEEE, IEEE, 1984.

[3] T. A. Anderson and C. E. Kim. Representation of digital line segments and their preimages. *Computer Graphics and Image Processing*, 30:279–288, 1985.

[4] E. Angel and D. Morrison. Speeding up Bresenham's algorithm. *IEEE Computer Graphics and Applications*, 11(6):16–17, November 1991.

[5] C. Arcelli and A. Massarotti. On the parallel generation of straight digital lines. *Computer Graphics and Image Processing*, 7(1):67–83, February 1978.

[6] J. Arvo and D. Kirk. A survey of ray tracing techniques. In A. S. Glassner, editor, *An Introduction to Ray Tracing*, pages 201–262. Academic Press, 1989.

[7] P. G. Bao and J. G. Rokne. Quadruple-step line generation. *Computers and Graphics*, 13:461–469, 1989.

[8] J. Barros and H. Fuchs. Generating smooth 2-D monocolor line drawings on video displays. *Computer Graphics (SIGGRAPH '79 Proceedings)*, 13(3):260–269, August 1979.

[9] K. Belser. Comment on "an improved algorithm for the generation of non-parametric curves". *IEEE Transactions on Computers*, C-25:103, 1976.

[10] N. E. Berg. Considerations in a general purpose line drawing package. In J. S. Summers, editor, *Conference Proc. Lasers in Graphics, Electronic Publishing in the 80's (2 vols)*, volume 2, pages 433–446. Dunn Technology, 1984.

[11] G. Bongiovanni, F. Luccio, and A. Zorat. The discrete equation of the straight line. *IEEE Transactions on Computers*, pages 310–313, 1975.

[12] J. Boothroyd and P.A. Hamilton. Exactly reversible plotter paths. *Australian Computer Journal*, 2(1):20–21, 1970.

[13] R. Borges. Line algorithms for raster displays rescued from round-off errors. *Computers and Graphics*, 15(2):155–160, 1991.

[14] J. E. Bresenham. An incremental algorithm for digital plotting. *ACM National Conference*, August 1963.

[15] J. E. Bresenham. Algorithm for computer control of a digital plotter. *IBM Systems Journal*, 4(1):25–30, July 1965.

[16] J. E. Bresenham. Incremental line compaction. *The Computer Journal*, 25(1):116–120, 1982.

[17] J. E. Bresenham. Run length slice algorithm for incremental lines. In R. A. Earnshaw, editor, *Fundamental Algorithms for Computer Graphics*, volume F17 of *NATO ASI*, pages 59–104. Springer-Verlag, 1985.

[18] J. E. Bresenham. Ambiguities in incremental line rastering. *IEEE Computer Graphics and Applications*, 7(5):31–43, May 1987.

[19] J. E. Bresenham. Something old, new, borrowed, and blue. *IEEE Computer Graphics and Applications*, 10(5):42–44, September 1990.

[20] J. E. Bresenham, D. G. Grice, and S. C. Pi. Run length slices for incremental lines. *IBM Technical Disclosure Bulletin*, 22-8B:3744–3747, January 1980.

[21] R. Brons. Linguistic methods for the description of a straight line on a grid. *Computer Graphics and Image Processing*, 3(1):48–62, March 1974.

[22] R. Brons. Theoretical and linguistic methods for describing straight lines. In R. A. Earnshaw, editor, *Fundamental Algorithms for Computer Graphics*, volume F17 of *NATO ASI*, pages 19–57. Springer-Verlag, 1985.

[23] A. M. Bruckstein. The self–similarity of digital straight lines. In *Proceedings of the 10th International Conference on Pattern Recognition*, volume B, pages 485–490, 1990.

[24] A. M. Bruckstein. Self–similarity properties of digital straight lines. *Contemporary Mathematics*, 119:1–20, 1990.

[25] V. Caglioti. On the uncertainty of straight lines in digital images. *Graphical Models and Image Processing*, 55(4):255–270, July 1993.

[26] G. Casciola. Basic concepts to accelerate line algorithms. *Computers and Graphics*, 12(3/4):489–502, 1988.

[27] C. M. Castle and M. L. V. Pitteway. An application of euclid's algorithm to drawing straight lines. In R. A. Earnshaw, editor, *Fundamental Algorithms for Computer Graphics*, volume F17 of *NATO ASI*, pages 135–139. Springer-Verlag, 1985.

[28] C. M. Castle and M. L. V. Pitteway. An efficient structural technique for encoding 'best-fit' straight lines. *The Computer Journal*, 30:168–175, 1987.

[29] R. L. T. Cederberg. Chain-link coding and segmentation for raster scan devices. *Computer Graphics and Image Processing*, 10:224–234, July 1979.

[30] R. L. T. Cederberg. A new method for vector generation. *Computer Graphics and Image Processing*, 9:183–195, February 1979.

[31] Y. L. Chang and J. K. Aggarwal. Representing and estimating 3–D lines. *Pattern Recognition*, 28:1181–1190, 1995.

[32] S. Chattopadhyay and P. P. Das. A new method of analysis for discrete straight lines. *Pattern Recognition Letters*, 12:747–755, 1991.

[33] P.-E. Danielsson. Converting a curve to right-angled increments. *Nord. Tidsk. Informationsbehandling (BIT)*, 3:213–221, 1963.

[34] P.-E. Danielsson. Incremental curve generation. *IEEE Transactions on Computers*, C-19:783–793, September 1970.

[35] G. H. Dao and J. J. Watters. Run slice line draw engine with enhanced clipping techniques. US Patent 5600769, February 1997. Compaq Computer Corporation.

[36] G. H. Dao and J. J. Watters. Run slice line draw engine with enhanced processing capabilites. US Patent 5627957, May 1997. Compaq Computer Corporation.

[37] G. H. Dao and J. J. Watters. Run slice line draw engine with non-linear scaling capabilities. US Patent 5657435, August 1997. Compaq Computer Corporation.

[38] G. H. Dao and J. J. Watters. Run slice line draw engine with non-linear shading capabilities. US Patent 5611029, March 1997. Compaq Computer Corporation.

[39] G. H. Dao and J. J. Watters. Run slice line draw engine with shading capabilities. US Patent 5617524, April 1997. Compaq Computer Corporation.

[40] G. H. Dao and J. J. Watters. Run slice line draw engine with stretching capabilities. US Patent 5627956, May 1997. Compaq Computer Corporation.

[41] G. H. Dao, J. J. Watters, and R.W. Spurlock. Method and apparatus for displaying a parametric curve on a video display. US Patent 5604852, February 1997. Compaq Computer Corporation.

[42] G.H. Dao. Run slice line draw engine with enhanced line configurations. US Patent 5714986, February 1998. Compaq Computer Corporation.

[43] L. Dorst. The accuracy of the digital representation of a straight line. In R. A. Earnshaw, editor, *Fundamental Algorithms for Computer Graphics*, volume F17 of *NATO ASI*, pages 141–152. Springer-Verlag, 1985.

[44] L. Dorst. *Discrete Straight Line Segments: Parameters, Primitives and Properties*. Ph.d. thesis, Offsetdrukkerij, Kanters BV, June 1986.

[45] L. Dorst and R. P. W. Duin. Spirograph theory: A framework for calculations on digitized straight lines. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, PAMI-6(5):632–639, September 1984.

[46] L. Dorst and A. W. M. Smeulders. The estimation of parameters of digital straight line segments. In *Proceedings of the 6th International Conference on Pattern Recognition*, pages 601–603, 1982.

[47] L. Dorst and A. W. M. Smeulders. Discrete straight line segments:Parameters, primitives and properties. *Contemporary Mathematics*, 119:45–62, 1990.

[48] R. A. Earnshaw. Line generation for incremental and raster devices. *Computer Graphics*, 11:199–205, Summer 1977.

[49] R. A. Earnshaw. Line tracking for incremental plotters. *Computer Journal*, 23:46–52, February 1980.

[50] R. Endl and M. Sommer. Classification of ray-generators in uniform subdivisions and octrees for ray tracing. *Computer Graphics Forum*, 13(1):3–19, 1994.

[51] Euclid. *Elements*, volume 7. 300 B.C.

[52] D. E. Field. *Algorithms for drawing simple geometric objects on raster devices.* Ph.d. thesis, Princeton U., 1983.

[53] O. Figueiredo and J.-P. Reveilles. New results about 3-d digital lines. In *Proceedings of SPIE*, pages 98–108, 1996.

[54] W. R. Franklin. Evaluation of algorithms to display vector plots on raster devices. *Computer Graphics and Image Processing*, 11:377–397, December 1979.

[55] H. Freeman. On the encoding of arbitrary geometric configurations. *IRE Transactions*, EC-102:260–268, June 1961.

[56] H. Freeman. Boundary encoding and processing. In B. S. Lipkin and A. Rosenfeld, editors, *Picture Processing and Psychopictorics*, pages 241–266. Academic Press, 1970.

[57] H. Freeman. Algorithm for generating a digital straight line on a triangular grid. *IEEE Transactions on Computers*, C-28:150–152, February 1979.

[58] H. Freeman and J. A. Saghri. Comparative analysis of line-drawing modeling schemes. *Computer Graphics and Image Processing*, 12:203–223, March 1980.

[59] A. Fujimoto and K. Iwata. Accelerated ray tracing. In Tosiyasu Kunii, editor, *Computer Graphics: Visual Technology and Art (Proceedings of Computer Graphics Tokyo '85)*, pages 41–65, New York, 1985. Springer Verlag.

[60] A. Fujimoto, T. Tanaka, and K. Iwata. Arts: Accelerated ray-tracing system. *IEEE Computer Graphics & Applications*, pages 16–26, April 1986.

[61] K. Y. Fung, T. M. Nicholl, and A. K. Dewdney. A run-length slice line drawing algorithm without division operations. *Computer Graphics Forum (EUROGRAPHICS '92 Proceedings)*, 11(3):267–277, September 1992.

[62] M. Gaafar. Convexity verification, block-chords, and digital straight lines. *Computer Graphics and Image Processing*, 6:361–370, August 1977.

[63] I. Garantini and H. H. Atkinson. Ray tracing an octree: Numerical evaluation of the first intersection. *Computer Graphics Forum*, 12(4):199–120, 1993.

[64] P.L. Gardner. Modification of Bresenham's algorithm for displays. *IBM Technical Disclosure Bulletin*, pages 1595–1596, October 1975.

[65] M. D. Gibbs. Angled vector generation. *IBM Technical Disclosure Bulletin*, 21(5):2041–2044, October 1978.

[66] G. W. Gill. N–step incremental straight-line algorithm. *IEEE Computer Graphics and Applications*, 14(3):66–67, 1994.

[67] A. S. Glassner. Space subdivision for fast ray tracing. *IEEE Computer Graphics & Applications*, 4(10):15–22, October 1984.

[68] P. Graham and S. Iyengar. Double– and triple–step incremental linear interpolation. *IEEE Computer Graphics and Applications*, 14(3):49–50, 1994.

[69] P. Graham, S. S. Iyengar, and S.-Q. Zheng. An efficient line drawing algorithm for parallel machines. *Lecture Notes in Computer Science*, 654:113–134, 1992.

[70] P. Graham, S. S. Iyengar, and S.-Q. Zheng. Improved recursive bisection line drawing algorithms. *CANDG: Computers & Graphics*, 19, 1995.

[71] G. H. Hardy and E. M. Wright. *An Introduction to the Theory of Numbers*. Oxford University Press, 1960.

[72] S. K. Hoo. Accelerated Bresenham algorithm. *IBM Technical Disclosure Bulletin*, 18(4):1075–1077, September 1975.

[73] S. H. Y. Hung. On the straightness of digitial arcs. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, PAMI-7:203–215, 1985.

[74] S. H. Y. Hung and T. Kasvand. On the chord property and its equivalences. In *Proceedings of the 7th International Conference on Pattern Recognition*, pages 116–119, 1984.

[75] D. Jevans and B. Wyvill. Adaptive voxel subdivision for ray tracing. *Proceedings of Graphics Interface '89*, pages 164–172, June 1989.

[76] B. Jordan, W. Lennon, and B. Holm. An improved algorithm for the generation of nonparametric curves. *IEEE Transactions on Computers*, C-22(12):1052, December 1973.

[77] S. A. T. M. Khalid and M. Kaykobad. Efficient line algorithm. *Midwest Symposium on Circuits and Systems*, 3:1280–1282, 1996.

[78] A. N. Khovanskii, editor. *The Application of Continued Fractions*. P. Noordhoff Ltd., 1963.

[79] C. E. Kim. On cellular straight line segments. *Computer Graphics and Image Processing*, 18:369–391, 1982.

[80] C. E. Kim. Three–dimensional digital line segments. *IEEE Transactions on Pattern Analysis Machine Intelligence*, PAMI-5:231–234, 1983.

[81] H. Klaasman. Some aspects of the accuracy of the approximated position of a straight line on a square grid. *Computer Graphics and Image Processing*, 4:225–235, 1975.

[82] D. Knuth. *The Art of Computer Programming*, volume II. Academic Press, 1969.

[83] V. A. Kovalevsky. New definition and fast recognition of digital straight line segments and arcs. In *Proceedings of the 10th International Conference on Pattern Recognition*, volume B, pages 31–34, 1990.

[84] W. G. Kropatsch and H. Tockner. Detecting the straightness of digital curves in O(N) steps. *Computer Graphics and Image Processing*, 45:1–21, 1989.

[85] Y. P. Kuzmin. Ray traversal of spatial structures. *Computer Graphics Forum*, 13(4):223–227, 1994.

[86] J. Landsdown. Drawing the line. *Computer Bulletin*, 2(4), December 1986.

[87] H. C. Lee and K. S. Fu. Using the FFT to determine digital straight line chain codes. *Computer Graphics and Image Processing*, 18:359–368, April 1982.

[88] S. X. Li and M. H. Leow. Analysis and modeling of digitized straight–line segments. In *Proceedings of the 9th International Conference on Pattern Recognition*, pages 294–296, 1988.

[89] M. Lindenbaum and J. Koplowitz. A new parametrization of digital straight lines. *IEEE Transactions on Pattern Analysis Machine Intelligence*, PAMI-13:847–852, 1991.

[90] T. Lindgren and J. Weber. Measuring the quality of antialiased line drawing algorithms. In Michael F. Cohen, Claude Puech, and Francois Sillion, editors, *Fourth Eurographics Workshop on Rendering*, pages 157–174. Eurographics, June 1993. held in Paris, France, 14–16 June 1993.

[91] M. Loceff. A new approach to high-speed computer graphics: The line. *Computer (IEEE)*, 13:56–58, 60, 62–63, 65–66, June 1980.

[92] M. G. Luby. Grid geometries which preserve properties of euclidean geometry: A study of graphics line drawing algorithms. In R. A. Earnshaw, editor, *Theoretical Foundations of Computer Graphics and CAD*, volume F40 of *NATO ASI*, pages 397–432. Springer-Verlag, 1988.

[93] M. D. McIlroy. Number theory in computer graphics. In *PSAM: Proceedings of the 46th Symposium in Applied Mathematics*, 1992.

[94] G. M. Megson and D. J. Evans. More on systolic line drawing. *Parallel Computing*, 18(3):355–358, March 1992.

[95] R. A. Melter and A. Rosenfeld. New views of linearity and connectedness in digital geometry. *Pattern Recognition Letters*, 10:9–16, 1989.

[96] R. A. Melter, I. Stojmenovic, and J. Zunic. A new characterisation of digital lines by least square fits. *Pattern Recognition Letters*, 14:83–88, 1993.

[97] R. A. Metzger. Computer generated graphical segments in a raster display. In *AFIPS Conference Proceedings*, pages 161–172, 1969.

[98] O. Neugebauer. *The Exact Sciences in Antiquity.* Princeton University Press, 1952.

[99] O. Nevalainen and M. Vesterinen. Solving the order of line drawing. *Angew. Inf.*, 20:15–19, 1978.

[100] A. T. Pang. Line-drawing algorithms for parallel machines. *IEEE Computer Graphics and Applications*, 10(5):54–59, September 1990.

[101] J. R. Parker. Extracting vectors from raster images. *Computers and Graphics*, 12:75–79, 1988.

[102] S. Pham. Digital straight segments. *Computer Graphics and Image Processing*, 36:10–30, 1986.

[103] M. L. V. Pitteway. The relationship between Euclid's algorithms and run-length encoding. In R. A. Earnshaw, editor, *Fundamental Algorithms for Computer Graphics*, volume F17 of *NATO ASI*, pages 105–112. Springer-Verlag, 1985.

[104] M. L. V. Pitteway and A. J. R. Green. Bresenham's algorithm with run line coding. *The Computer Journal*, 25(1):114–115, 1982.

[105] J. Ramot. Nonparametric curves. *IEEE Transactions on Computers*, C-25:103–104, 1976.

[106] J. R. Rankin. Recursive bisection line algorithm. *Computers and Graphics*, 15(1):1–8, 1991.

[107] G. B. Reggiori. Digital computer transformations for irregular line drawings. Technical Report 403-22, New York University, April 1972.

[108] W. Rienecker and G. Triftshauser. A vector generator for interactive graphics displays. *Elektronik*, 27:95–97, June 1978.

[109] J. G. Rokne, B. Wyvill, and X. Wu. Fast line scan-conversion. *ACM Trans. Graph.*, 9:376–388, 1992.

[110] C. Ronse. A simple proof of Rosenfeld's characterization of digital straight line segments. *Pattern Recognition Letters*, 3:323–326, 1985.

[111] A. Rosenfeld. Digital straight line segments. *IEEE Transactions on Computers*, C-23(12):1264–1269, December 1974.

[112] A. Rosenfeld. Digital geometry: Introduction and bibliography. Technical Report University of Maryland Computer Science Technical Report 3743, University of Maryland, Department of Computer Science, February 1997.

[113] A. Rosenfeld and C. E. Kim. How a digital computer can tell whether a line is straight. *American Mathematical Monthly*, 89:230–235, 1982.

[114] J. Rothstein and C. F. R. Weiman. Parallel and sequential specification of a context sensitive language for straight lines on grids. *Computer Graphics and Image Processing*, 5:106–124, 1979.

[115] F. Rubin. Generation of nonparametric curves. *IEEE Transactions on Computers*, C-25:103, 1976.

[116] Y. M. Sharaiha and P. Garat. A compact chord property for digital arcs. *Pattern Recognition*, 26:799–803, 1993.

[117] Y. M. Sharaiha and P. Garat. Digital straightness and the skeleton property. *Pattern Recognition Letters*, 16:417–423, 1995.

[118] R. Shoucri, R. Benesch, and S. Thomas. Note on the determination of digital straight line from chain codes. *Computer Vision, Graphics and Image Process*, 29(1):133–139, January 1985.

[119] T. R. H. Sizer, editor. *The Digital Differential Analyser*. Chapman and Hall, 1968.

[120] M. Slater. Tracing a ray through uniformly subdivided n–dimensional space. *The Visual Computer*, 9:39–46, 1992.

[121] J. Spackman and P. Willis. The SMART navigation of a ray through an oct-tree. *Computers and Graphics*, 15(2):185–194, 1991.

[122] R. F. Sproull. Using program transformations to derive line-drawing algorithms. *ACM Transactions on Graphics*, 1:259–273, October 1982.

[123] R. Steinbrueggen. Line-drawing on a discrete grid. Technical Report TUM - 18610, Muenchen, 1986.

[124] F. G. Stockton. Algorithm 162 XMOVE PLOTTING. *Communications of the ACM*, 6(4), 1963.

[125] I. Stojmenovic and R. Tosic. Digitization schemes and the recognition of digital straight lines, hyperplanes, and flats in arbitrary dimensions. *Contemporary Mathematics*, 119:197–212, 1991.

[126] Q. F. Stout. Drawing straight lines with a pyramid cellular automation. *Information Processing Letters*, 15:233–237, 1982.

[127] Y. Suenaga, T. Kamae, and T. Kobayashi. A high-speed algorithm for the generation of straight lines and circular arcs. *IEEE Transactions on Computers*, C-28:728–736, October 1979.

[128] K. Sung. A dda octree traversal algorithm for ray tracing. In *Eurographics '91*, pages 11–23. North-Holland, 1991.

[129] J. R. Thompson. Straight lines and graph plotters. *The Computer Journal*, 7:227, 1964.

[130] T. Thong. A symmetric linear algorithm for line segment generation. *Computers and Graphics*, 6(1):15–17, 1982.

[131] J. V. Tucker and S. M. Eker. Specification, derivation and verification of concurrent line drawing algorithms and architectures. In R. A. Earnshaw, editor, *Theoretical Foundations of Computer Graphics and CAD*, volume F40 of *NATO ASI*, pages 449–516. Springer-Verlag, 1988.

[132] M. L. P. van Lierop, C. W. A. M. van Overveld, and H. M. M. van de Wetering. Line rasterization algorithms that satisfy the subset line property. *Computer Graphics and Image Processing*, 41:210–228, 1988.

[133] A. M. Vossepoel and A. W. M. Smeulders. Vector code probabilitity and metrication error in the representation of straight lines of finite length. *Computer Graphics and Image Processing*, 20:347–364, 1982.

[134] F. L. Weng and L. D. Wu. Covering property and *–chain code. In *Proceedings of the 8th International Conference on Pattern Recognition*, pages 1133–1135, 1986.

[135] T. Whitted. An improved illumination model for shaded display. *Communications of the ACM*, 23(6):343–349, June 1980.

[136] M. C. Whitton. Memory design for raster graphics displays. *IEEE Computer Graphics and Applications*, 4(3):48–65, 1984.

[137] C. M. Williams. Bounded straight-line approximation of digitized planar curves and lines. *Computer Graphics and Image Processing*, 16:370–381, August 1981.

[138] G. Woetzel. A fast and economic scan-to-line-conversion algorithm. *Computer Graphics (SIGGRAPH '78 Proceedings)*, 12(3):125–129, August 1978.

[139] W. E. Wright. Parallelization of Bresenham's line and circle algorithms. *IEEE Computer Graphics and Applications*, 10(5):60–67, September 1990.

[140] W. E. Wright. Parallel algorithms for generating the raster representation of straight lines and circles. *Journal of Parallel and Distributed Computing*, 11:170–173, 1991.

[141] L. Wu. On the chain code of a line. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, PAMI-4:347–353, 1982.

[142] X. Wu and J. G. Rokne. Double-step incremental generation of lines and circles. *Computer Graphics and Image Analysis*, 37(4):331–334, March 1987.

[143] B. Wyvill. Symmetric double step line algorithm. In A. Glassner, editor, *Graphics Gems*, pages 101–104. Academic Press, 1990.

[144] R. Yagel, D. Cohen, and A. Kaufman. Discrete ray tracing. *IEEE Computer Graphics and Applications*, pages 19–28, September 1992.

[145] C. Yao and J. Rokne. Bi-directional incremental linear interpolation. *Computers and Graphics*, 20(2):295–305, 1996.

[146] J. Yuan and C. Y. Suen. An optimal algorithm for detecting straight lines in chain codes. In *Proceedings of the 11th International Conference on Pattern Recognition*, volume C, pages 692–695, 1992.

[147] J. Yuan and C. Y. Suen. An optimal O(N) algorithm for identifying line segments from a sequence of chain codes. *Pattern Recognition*, 28:635–646, 1995.