Algorithm 781: Generating Hilbert's Space-Filling Curve by Recursion

GREG BREINHOLT and CHRISTOPH SCHIERZ
Swiss Federal Institute of Technology

An efficient algorithm for the generation of Hilbert's space-filling curve is given. The algorithm implements a recursive procedure that involves simple integer operations and quickly converges to the set of points that make the Hilbert curve. The algorithm is elegant, short, and considerably easier to implement than previous recursive and nonrecursive algorithms and can be efficiently implemented in all programming languages that have integer operations and allow recursion. The fundamental Hilbert shape (a line joining the four corners of a square) is represented by two variables with values of either 0 or 1. This coding technique could be successfully applied to the generation of other regular space-filling curves, such as the Peano curve.

Categories and Subject Descriptors: D.3.2 [**Programming Languages**]: Language Classifications—*C*; I.3.3 [**Computer Graphics**]: Picture/Image Generation—*line and curve generation*

General Terms: Algorithms

Additional Key Words and Phrases: Recursion

1. INTRODUCTION

Although it was Peano [1890] that produced the first space-filling curves, it was Hilbert [1891] who first popularized their existence and gave an insight into their generation. Space-filling curves are commonly used to reduce a multidimensional problem to a one-dimensional problem; the curve is essentially a linear transversal of the discrete multidimensional space. Hilbert curves belong to the class of *FASS* curves; an acronym for space-**f**illing, self-**a**voiding, **s**imple, and self-**s**imilar [Prusinkiewicz and Lindenmayer 1990]. FASS curves can be thought of as finite, self-avoiding approximations of curves that pass through all points of a square. The Hilbert curve is a particular form of the FASS curve that scans a $2^m \times 2^m$ array of points while never maintaining the same direction for more

Authors' address: Institute for Hygiene and Applied Physiology, Swiss Federal Institute of Technology, Zurich, CH-8092, Switzerland; email: breinholt@computer.org.
Permission to make digital/hard copy of part or all of this work for personal or classroom use is granted without fee provided that the copies are not made or distributed for profit or commercial advantage, the copyright notice, the title of the publication, and its date appear, and notice is given that copying is by permission of the ACM, Inc. To copy otherwise, to republish, to post on servers, or to redistribute to lists, requires prior specific permission and/or a fee.
© 1998 ACM 0098-3500/98/0600–0184 $5.00

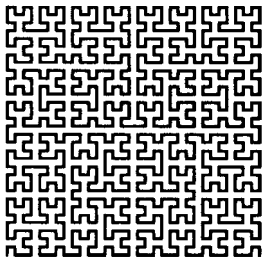

Fig. 1. Hilbert curve filling an area of 32 × 32 points.

than three consecutive points. Figure 1 shows an example of a Hilbert curve.

Different methods of curve generation have been suggested: a byte-oriented technique [Butz 1971], an arithmetization of the curve [Sagan 1992], L-system formalization with turtle interpretation [Prusinkiewicz et al. 1991], and fractal curve generation [Ohno and Ohyama 1991; Peitgen et al. 1992]. A simple algorithm for the generation of the nodal points of the Hilbert curve has also been derived by Sagan [1994], though when these points are linked they produce only approximating polygons that are not a "true" representation of the Hilbert curve.

Though space-filling curves were discovered over a century ago their use has been sporadic but varied. Interesting examples of their use have been in mathematics and data structures [Butz 1968; 1969; Gotsman and Lindenbaum 1994; Asano et al. 1995], traveling salesman problems [Gao and Steele 1994; Norman and Moscato 1995], image manipulation and analysis [Stevens et al. 1983; Kamata et al. 1995; Agranov and Gotsman 1995; Lamarque and Robert 1996; Giordana and Pieczynski 1997], digital halftoning [Witten and Neal 1982; Velho and Gomes 1991; Zhang and Webber 1993], pattern and texture analysis [Abend et al. 1965; Lee and Hsueh 1994], cryptology [Matias and Shamir 1987; Bertilsson et al. 1989], and data compression [Bially 1969; Moghaddam et al. 1991].

Most of the techniques for curve generation are interpretations of Hilbert's original suggestion to continually divide a plane into four parts, each of these parts into four parts, and so on, calculating the necessary plotting points as the division proceeds. This continual dividing of the plane continues until the required curve resolution is obtained, i.e., in a typical recursive procedure. The exact method to determine the points was not given by Hilbert (who worked before the invention of computers), and this has lead to the many different programming code solutions to the problem [Goldschlager 1981; Cole 1983; Willen and Wyvill 1983].

This article presents a simple computer-based algorithm that can quickly and efficiently generate the points of the Hilbert curve using the simplest recursive technique.

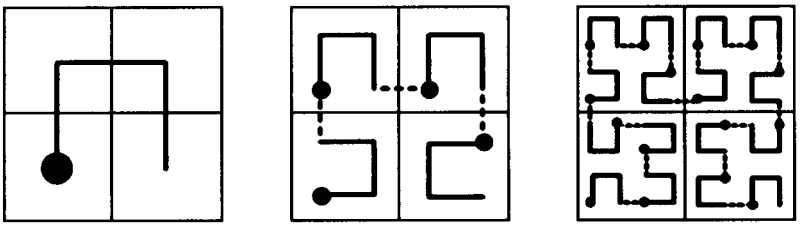

Fig. 2. Deconstructed unit shapes for different curve resolutions. The circles represent the start points.

2. IMPLEMENTATION

The Hilbert curve can be deconstructed into a set of unit shapes (Figure 2), the relative position and rotation of each defined by its sequential position in the curve generation. As the resolution of the curve increases, more unit shapes are required for its description, but the principle remains true to Hilbert's original proposition of dividing each part into smaller parts.

As the curve progresses, the rotation of each unit shape follows a pattern that must be efficiently coded to describe both the rotation of the unit shape and its start and end points, so that the points of the curve can be plotted in the correct position and the correct order. The scheme chosen codes the rotation of each unit shape into two variables i1 and i2, which represent the starting point and end point respectively (Figure 3).

Variable i1 is given the value 0 when the starting point is in the lower left corner of the unit shape, and the value 1 when the starting point is in the upper right corner. Variable i2 is given the value 0 when the end point of the unit shape is in the lower right corner, and the value 1 when the end point is in the upper left corner. Figure 4 illustrates this coding system when applied to four of the possible orientations of the unit shape used to construct the curve.

The method is implemented in a recursive procedure that modifies its calling parameters as it converges to the set of points on the curve. For example, to generate a curve that fills a 64×64 point area, the following is the first call: $Hilbert(0,0,64,0,0)$.

3. IMPLEMENTATION RESULTS

The procedure works by recursively calling itself, modifying its parameters with each call, until it reaches the smallest unit shape for the next section of the curve to be plotted. This occurs by repeatedly halving the parameter *lg* (representing the side length of the unit shape) until a unit value is achieved, and then the next four points on the curve are plotted.

The number of calls to the procedure can be easily calculated:

$$Number\ Calls = W^2 + \frac{1}{3}(W^2 - 1)$$

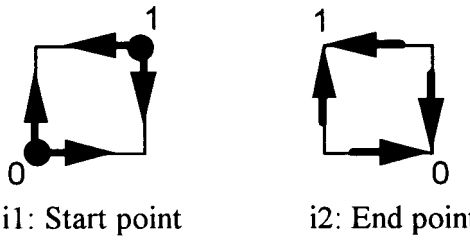

Fig. 3. Coding system to describe the start and end points of the unit shape.

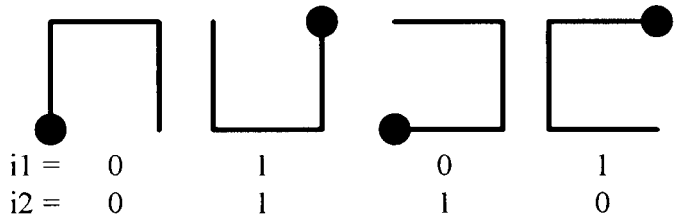

Fig. 4. Coding system applied to four orientations of the unit shape.

$W =$ curve width (2^m; e.g., 1, 2, 4, 8, 16, 32, 64, ...)

1st term : calls that actually produce a point on the curve.

2nd term : due to recursive calling to find the desired points.

This gives the ratio

$$\frac{\text{Points}}{\text{Calls}} = \frac{3W^2}{4W^2 - 1} \rightarrow 75\% \text{ for } W >= 8.$$

At curve widths greater than 8, the algorithmic calling efficiency (number of points generated/number of calls to the recursive procedure) approximates to 75%. This is a relatively high ratio and demonstrates an efficient recursive procedure.

As noted by one of the reviewers of this article, the recursive algorithm could be modified so that the last recursive calls (i.e., those with $lg = 1$) directly produce the next four points, rather than calling the *Hilbert* function again. This would greatly enhance the efficiency of the algorithm, but the performance gain will not be so great during runtime, as the extra recursive calls made in the original algorithm are relatively time inexpensive. This optimizing technique will speed the performance of the algorithm, but detracts slightly from its clarity, and so the algorithm given is the simplest, while not the most efficient.

4. SUMMARY

A recursive algorithm has been introduced for the generation of the points that make the Hilbert curve. The algorithmic calling efficiency (number of points generated/number of calls to the recursive procedure) settles at 75% for curve widths greater than 8, showing a relatively high efficiency. Though the algorithm requires many calls to generate the full curve (one third more calls than the number of points to be generated), each call involves only simple integer operations that are quickly processed on modern computers. The procedure can be implemented in all languages that have integer operations and allow recursion (C, PASCAL, JAVA etc.).

The same technique could be applied to other space-filling curves by identifying their unit shape, describing the points that make this unit shape, then developing a simple recursive procedure to generate all the points that make this curve from the unit shape. Another way to describe the process, as in the Hilbert curve, is that the square space is divided into a 2×2 array, and then further divided until the required resolution of curve is obtained. It would be possible to divide the same square space into a 3×3 array (or 4×4, or higher) and then by a simple modification of the recursive algorithm (requiring more calls for larger arrays), produce the required points. This would enable the construction of other forms of space-filling curve, such as the Peano curve.

ACKNOWLEDGMENTS

I would like to sincerely thank the reviewers of this article, who all contributed greatly, and their diligence in finding improvements and insights into possible extensions, have improved this article and provided material for further work.

REFERENCES

ABEND, K., HARLEY, T. J., AND KANAL, L. N. 1965. Classification of binary random patterns. *IEEE Trans. Inf. Theor. IT-11*, 4 (Oct.), 538–544.

AGRANOV, G. AND GOTSMAN, C. 1995. Algorithms for rendering realistic terrain image sequences and their parallel implementation. *Visual Comput. 11*, 9, 455–464.

ASANO, T., RANJAN, D., ROOS, T., WELZL, E., AND WIDMAYER, P. 1995. Space filling curves and their use in the design of geometric data structures. In *Proceedings of the Conference on Theoretical Informatics* (LATIN '95). 36–48.

BERTILSSON, M., BRICKELL, E., AND INGEMARSSON, I. 1989. Cryptanalysis of video encryption based on space-filling curves. In *Advances in Cryptology—EUROCRYPT '89*. 403–411.

BIALLY, T. 1969. Space-filling curves: Their generation and their application to bandwidth reduction. *IEEE Trans. Inf. Theor. IT-15*, 6 (Oct.), 658–664.

BUTZ, A. R. 1968. Space filling curves and mathematical programmming. *Inf. Control 12*, 314–330.

BUTZ, A. R. 1969. Convergence with Hilbert's space filling curve. *J. Comput. Syst. Sci. 3*, 128–146.

BUTZ, A. R. 1971. Alternative algorithm for Hilbert's space-filling curve. *IEEE Trans. Comput. C-20* (Apr.), 424–426.

COLE, A. J. 1983. A note on space filling curves. *Softw. Pract. Exper. 13*, 12 (Dec.), 1181–1189.

GAO, J. AND STEELE, J. M. 1994. General spacefilling curve heuristics and limit theory for the traveling salesman problem. *J. Complexity 10*, 2 (June), 230–245.

GIORDANA, N. AND PIECZYNSKI, W. 1997. Estimation of generalized multisensor hidden Markov chains and unsupervised image segmentation. *IEEE Trans. Pattern Anal. Mach. Intell. 19*, 5, 465–475.

GOLDSCHLAGER, L. M. 1981. Short algorithms for space-filling curves. *Softw. Pract. Exper. 11*, 1 (Jan.), 99–100.

GOTSMAN, T. AND LINDENBAUM, M. 1994. On the metric properties of discrete space-filling curves. In *Proceedings of the 12th IAPR International Conference on Pattern Recognition*. IEEE Computer Society Press, Los Alamitos, CA, 98–102.

HILBERT, D. 1891. Über die stetige Abbildung einer Linie auf ein Flächenstück. *Math. Ann. 38* (Mar.), 459–460.

KAMATA, S., KAWAGUCHI, E., AND NIIMI, M. 1995. An interactive analysis method for multidimensional images using a Hilbert curve. *Syst. Comput. Japan 26*, 3, 83–92.

LAMARQUE, C.-H. AND ROBERT, F. 1996. Image analysis using space-filling curves and 1D wavelet bases. *Pattern Recognit. 29*, 8, 1309–1322.

LEE, J.-H. AND HSUEH, Y.-C. 1994. Texture classification method using multiple space filling curves. *Pattern Recognit. Lett. 15*, 12 (Dec.), 1241–1244.

MATIAS, Y. AND SHAMIR, A. 1987. A video scrambling technique based on space filling curves. In *Advances in Cryptology—CRYPTO '89*. 398–416.

MOGHADDAM, B., HINTZ, K. J., AND STEWART, C. V. 1991. Space-filling curves for image compression. In *Proceedings of the SPIE*. 414–421.

NORMAN, M. G. AND MOSCATO, P. 1995. The Euclidean traveling salesman problem and a space-filling curve. *Chaos Solitons Fractals 6*, 389–397.

OHNO, Y. AND OHYAMA, K. 1991. A catalog of symmetric self-similar space-filling curves. *J. Recreat. Math. 23*, 3, 161–174.

PEANO, G. 1890. Sur une Courbe qui Remplit Toute une Aire Plane. *Math. Ann. 36*, 157–160.

PETIGEN, H.-O., JÜRGENS, H., AND SAUPE, D. 1992. *Chaos and Fractals*. Springer-Verlag, New York, NY.

PRUSINKIEWICZ, P. AND LINDENMAYER, A. 1990. *The Algorithmic Beauty of Plants*. Springer-Verlag, New York, NY.

PRUSINKIEWICZ, P., LINDENMAYER, A., AND FRACCHIA, F. D. 1991. Synthesis of space-filling curves on the square grid. In *Fractals in the Fundamental and Applied Sciences*, Peitigen, H.-O., Henriques, J. M., and Pendo, L. F., Eds. Elsevier Sci. Pub. B. V., Amsterdam, The Netherlands, 341–366.

SAGAN, H. 1992. On the geometrization of the Peano curve and the arithmetization of the Hilbert curve. *J. Math. Educ. Sci. Tech. 23*, 3, 403–411.

SAGAN, H. 1994. *Space-Filling Curves*. Springer-Verlag, New York, NY.

STEVENS, R. J., LEHAR, A. F., AND PRESTON, F. H. 1983. Manipulation and presentation of multidimensional image data using the Peano Scan. *IEEE Trans. Pattern Anal. Mach. Intell. PAMI-5*, 5 (Sept.), 520–526.

VELHO, L. AND GOMES, J. D. M. 1991. Digital halftoning with space filling curves. *SIGGRAPH Comput. Graph. 25*, 4 (July), 81–90.

WITTEN, I. A. AND NEAL, R. M. 1982. Using Peano curves for bilevel display of continuous-tone images. *IEEE Comput. Graph. Appl. 2*, 3 (May), 47–52.

WITTEN, I. A. AND WYVILL, B. 1983. On the generation and use of space-filling curves. *Softw. Pract. Exper. 13*, 6 (June), 519–525.

ZHANG, Y. AND WEBBER, R. E. 1993. Space diffusion: An improved parallel halftoning technique using space-filling curves. In *Conference Proceedings on Computer Graphics* (SIGGRAPH 93, Anaheim, CA, Aug. 1–6). ACM Press, New York, NY, 305–312.

Received: June 1996; revised: October 1996 and July 1997; accepted: September 1997