Compaction Techniques for Raster Scan Graphics using Space-filling Curves

A. J. COLE
Department of Computational Science, University of St Andrews, St Andrews, Fife, KY16 9SX

A method of scanning raster graphics pictures based on the use of Peano and Hilbert space-filling polygons is discussed. This leads to significant data reduction for transmission or storage of pictures. Quadtree encoding and a generalisation of this occurs as a special case. The aliasing problem in raster scan graphics is also alleviated if a space-filling curve scan is used to refresh the screen. A possible application to pattern recognition is briefly discussed.

Received July 1985

1. INTRODUCTION

The concept of a space-filling curve based on a continuous mapping of the unit line on to the unit square was introduced by Peano.[1] The proof used a base-three representation of all coordinates, although Peano indicated that corresponding results followed for any odd-number base, and also showed how the concept could be extended to space-filling curves in n-dimensions.

Hilbert[2] derived an alternative method of defining a space-filling curve as the limit of polygons enclosed in the unit square, using a fourfold repetition of successive polygons which correspond to a base 2 number representation. A similar result based on the Peano technique was given by Moore[3] to obtain a limiting polygon based on ninefold repetitions of successive polygons.

Recursive algorithms for drawing these and other space-filling curves have been given by Wirth,[4] Goldschlager[5] and Witten and Wyvill,[6] and Cole[7] has shown how Peano, Hilbert and Sierpinski[8] polygons can all be obtained recursively from a single point. Griffiths[9] discusses table-driven algorithms for generating space-filling curves.

All of these methods, apart from those of the original Peano paper, generate the points on the nth approximating polygon sequentially according to their position on the corresponding polygon. Cole[10, 11] has obtained explicit mappings between the first n non-negative integers and the n sequentially traversed vertices of any of the Peano or Hilbert polygons, including the generalisations of these polygons as suggested by Peano for his continuous curves. These results coincidentally lead to fast scanning algorithms for both Peano and Hilbert polygons. In addition, if any polygon is clipped it is possible to pass to the next point within the corresponding window by direct computation.

A raster graphics screen can be regarded as a finite rectangular array of pixels with integer coordinates. Thus any of the above polygonal types of suitable order may be used to scan either part or the whole of the screen using windowing if necessary. In this paper we will be concerned primarily with polygons which fit exactly into the selected part or whole of the screen. Techniques for scanning other rectangles and indeed other shapes will be discussed in a later paper.

The advantages of using space-filling polygons for this purpose arise from the fact that in general the curve passes through many points local to each other in two dimensions. In particular, the Hilbert polygons include quadtrees[12–15] as a particular case with no additional computation or complex data structure required to record or to scan them. The concept of quadtrees generalises in an interesting way corresponding to Peano and Hilbert polygons derived from any number base.

In addition, if the screen regeneration is carried out using a space-filling curve scan the aliasing problem is diminished, since pixel joins change direction very frequently, thus avoiding the linear alignment of conventional raster scanning.

2. TRANSFORMATIONS FOR PEANO POLYGONS

Cole[10] discusses the general problem of explicit mappings between sets of integers and vertices on n-dimensional Peano polygons derived from any odd-number base and gives formal proofs of the results. We are primarily interested in the two-dimensional case using base 3 integers although other odd-number bases may also be of interest. We give here a brief description of the idea behind these special cases. For further details the reader is referred to the above paper.

The first three Peano polygons P1, P2, P3 are shown in Fig. 1. By inspection of P2 it can be seen that P2 is obtained from P1 by placing suitable orientations of P1 in nine positions in the plane and joining their end points by horizontal or vertical line segments. Cole[7] noted that P1 could be generated similarly from a single point P0, since the polygon P1 has nine vertices if each of the

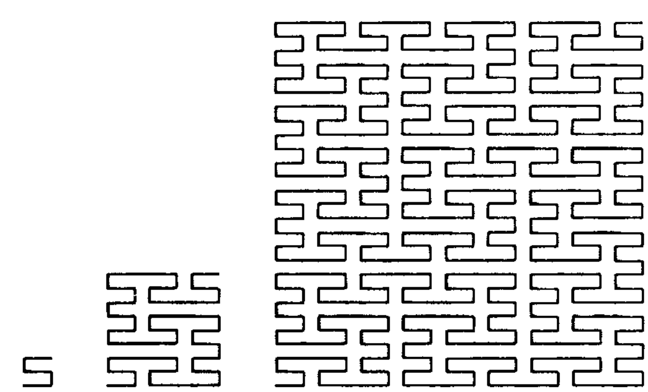

Figure 1

horizontal segments is broken by a vertex at its mid point. Exactly the same algorithm as for the higher-order cases can be used to generate P1 from P0, since rotation of a point leaves it unchanged.

Consider the Peano polygon Pn in two dimensions. Assuming as above that P0 has nine vertices, it follows that Pn has 3^{2n} vertices. We therefore need to define a one–one mapping between the integers $1, 2, 3, \ldots, 3^{2n}-1$ and the points $[0,0]$, $[1,0]$, $[2,0]$, $[2,1]$, ..., $[n-1, n-1]$ which are the successive vertices of Pn. The explicit mapping used is similar to that described by Peano but based on base 3 Gray code integers. Gray[12] discussed ways in which cyclic progressive number systems could be defined, and Cole[16] gave conversion rules and addition and multiplication tables for such systems. Essentially cyclic progressive-number systems are such that successive integers differ in only one digit. Although there are many ways of defining cyclic progressive-number systems the most common is as defined below and is, by convention, called the Gray code.

Suppose that

$$a = a_2, a_2, \ldots, a_m \quad (0 \leqslant a_i < n, i = 1, 2, \ldots, m)$$

is a base n non-negative integer and let

$$p_j = \left(\sum_{i-1}^{j} a_i\right) \bmod 2.$$

Then the Gray code odd base n integer corresponding to a is defined to be

$$a' = b_1 b_2 \ldots b_m$$

where

$$b_1 = a_1$$

and

$$b_i = \begin{cases} a_i & \text{if} \quad p_{i-1} = 0 \\ n-1-a_i & \text{if} \quad p_i = 1 \end{cases}$$

for $i = 2, 3, \ldots, m$.

Thus the first few base 3 Gray-code integers are 0, 1, 2, 12, 11, 10, 20, 21, 22, 122, 121, ...

For even-base Gray code integers the corresponding conversion rule is

$$b_1 = a_1$$

$$b_i = \begin{cases} a_i & \text{if} \quad a_{i-1} \quad \text{is even} \\ n-1-a_i & \text{if} \quad a_{i-1} \quad \text{is odd} \end{cases}$$

for $i = 2, 3, \ldots, m$.

Suppose now that

$$a = a_1 a_2 \ldots a_{2m} \quad (0 \leqslant a_i < 3, i = 1, 2, \ldots, 2m)$$

is a base 3 integer and

$$a' = b_1 b_2 \ldots b_{2m}$$

is the Gray code equivalent of a. Note that for both even- and odd-number bases $(a')' = a$. Let

$$x = b_2 b_4 b_6 \ldots b_{2m}$$

$$y = b_1 b_3 \ldots b_{2m-1}$$

Then the point (x, y) relative to integer Gray-code scale axes is the ath vertex on the Peano polygon Pm and conversely. Note that the leading digits of a, x and y may be zeros, so any pair of integers must be made up to the same length.

The formal proof of this result, which is given in Cole,[10] is dependent on the commutability of the operations of conversion to Gray code and reduced radix complementation. That is, if

$$a^* = c_1 c_2 \ldots c_m$$

were

$$c_i = n - 1 - a_i \quad (i = 1, 2, \ldots, m)$$

is the n reduced radix complement of a then

$$(a')^* = (a^*)'$$

This curious result is only true for odd-base numbers so, for example, a corresponding conversion from base 2 Gray code integers to successive points on Hilbert polygons does not hold.

As an example, the 50th point on P3 is found by converting 50 to base 3 giving 1212, with Gray-code equivalent 1012.

The required point now has Gray code coordinates (02,11), which also happens in this case to be the coordinates in standard base 3 integers. This result may be easily verified from the diagram for P2 in Fig. 1.

Similar results hold for any odd-based number system.

3. TRANSFORMATIONS FOR HILBERT POLYGONS

As has been indicated above, the obvious extension of the result given for Peano polygons is not valid when applied to even-base number systems. It is easy to see that the problem of non-commutability of conversion to cyclic progressive form and radix complementation arises for any even-number base, and consequently the method leading to the Peano result does not apply for any of the possible even-base cyclic progressive number systems. A less elegant, but computationally efficient method based effectively on table lookup is given in Cole.[11]

The method is closely related to the recursive derivation of the vertices of Hilbert polygons. For the simplest Hilbert polygons as shown in Fig. 2 we use a base 2 representation of the vertex number and its coordinates.

Consider each of the first 2^{2p} non-negative integers in binary form

$$a_1 a_2 \ldots a_{2p-1} a_{2p}$$

with all digits including leading zeros present. Since the Hilbert polygons fill each of the four quadrants of the enclosing square in anti-clockwise order the first pair of digits (a_1, a_2) uniquely determine the quadrant in which the corresponding vertex lies and therefore the most significant digits of that vertex. The process can now be repeated, but with the complication that there are four possible orientations for the new sub-polygons. We therefore need four tables corresponding to these four orientations each having four columns corresponding to the next pair of digits, the next x and y digits and the next table number. These four tables are displayed in Table 1.

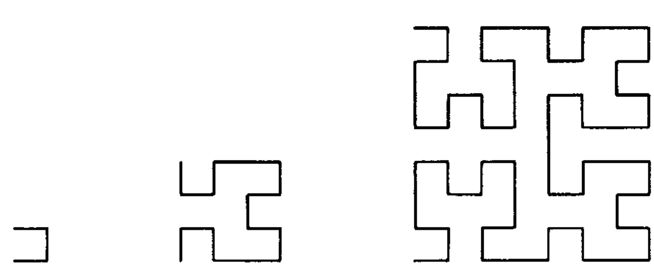

Figure 2

Table 1

Digit pair	x	y	Next table
	(1)		
00	0	0	2
01	1	0	1
10	1	1	1
11	0	1	4
	(2)		
00	0	0	1
01	0	1	2
10	1	1	2
11	1	0	3
	(3)		
00	1	1	4
01	0	1	3
10	0	0	3
11	1	0	2
	(4)		
00	1	1	3
01	1	0	4
10	0	0	4
11	0	1	1

Table 2

x, y pair	Integer pair	Next table
	(1)	
00	00	2
01	11	4
10	01	1
11	10	1
	(2)	
00	00	1
01	01	2
10	11	3
11	10	2
	(3)	
00	10	3
01	01	3
10	11	2
11	00	4
	(4)	
00	10	4
01	11	1
10	01	4
11	00	3

Similarly, given a vertex with coordinates (x, y), by considering the digit pair formed by taking corresponding digits from x and y it is possible to use successive table lookups to build the corresponding integer position of this vertex on the Peano polygon. The tables for this inverse operation are given in Table 2.

Corresponding tables can be built for any number base, both odd and even.

4. THE COMPACTION TECHNIQUE

Consider first a black-and-white raster graphics picture which can be enclosed in a square of pixels of side length n^p for some positive integers n and p. In practice it is computationally simpler to fix n and let p vary, but this is not essential. We return to this point later.

If n is even we choose the corresponding Hilbert polygon derived from base n numbers and similarly for Peano polygons when n is odd. In practice, if $n = 2$, then we use the appropriate Hilbert polygon as typified in Fig. 2, and if $n = 3$ then we use the appropriate Peano polygon as typified in Fig. 1. In either case, as we traverse the corresponding polygon we record the number of successive pixels without a colour change giving a sequence

$$c_1, c_2, \ldots, c_m$$

of segment lengths for pixels of the same colour. We need some way of indicating the colour associated with the first segment, and the simplest way to handle this is arbitrarily to fix the colour of the first segment to be white, say. If the first pixel is black then we simply record the first segment length as 0.

A typical compacted colour sequence for a square of side 8 could be

6 3 9 1 8 2 20 7 1 4 3

with meaning that the first 6 pixels are coloured white, the next 3 black and so on. Similarly the sequence

0 5 4 8 2 7 3 10 18 2 5

would have the first 5 pixels coloured black, the next 4 coloured white and so on. Note that the sum of segment lengths in both cases is 8^2.

The picture may now be reconstituted either in its original position or at any arbitrary starting point by scanning the corresponding polygon from the new starting point.

It is also necessary to record the frame size. If the picture fits exactly into a square of side n^p then it would be sufficient to record n and p, or if n is fixed, then p only. This is unnecessarily restricting since n^p increases rapidly with p. It is therefore better to record the actual lengths a, b of the sides of the smallest enclosing rectangle and to use these values to determine when the chosen space-filling polygon goes outside this rectangle. Since the mapping function is known explicitly for each integer and corresponding vertex pair it is possible to continue scanning at the next entry point to the enclosing rectangle. If this happens to have the same colour as the exit point then counting continues on the same segment. More efficient ways of handling this problem are to be discussed in a later paper.

The reason for obtaining good compaction by this technique is that, with blocks of colour, the number of segments per block is roughly proportional to half the number of points on the perimeter of the block rather than the total number of points contained within the block. When the space-filling curve enters the block the corresponding segment continues until it leaves the block, having used up two perimeter points which will not be traversed again. Sometimes the space-filling curve will only touch the perimeter at one point but these cases are balanced out by internal segments which touch the perimeter and go back inside, or follow the perimeter without change of colour.

A number of examples are discussed and a further compaction is defined in the results section of this paper.

In reconstituting black-and-white pictures in different parts of the screen it is useful either to be able to reproduce the picture exactly with each black or white pixel being exactly as in the original, or alternatively choosing one colour as the dominant one and leaving the other coloured pixels unchanged. In this way it is possible to superimpose silhouettes on an existing picture.

In colour graphics it is necessary to store the associated colour code with each segment. A method of doing this efficiently is discussed in the section below.

5. CODING

In real pictures a high proportion of the segments will be short, and it is therefore inefficient to use integer locations to store them. It is better to use a fixed number of bits which will be sufficient to record most segment lengths, together with a possible extension bit. Thus if a byte were to be used for this purpose 7 bits could be used to record possible segment lengths between 0 and 127, with the 8th bit being a flag to indicate when extension to subsequent bytes was required.

In colour coding, sufficient additional bits are needed to allow for all possible colour codes. In low-quality colour systems these could be packed into one byte along with the segment length, again with a flag to indicate subsequent extensions. In high-quality colour systems, where a large number of isolated pixels of different colours are likely, it is probably better to extend the colour code by a single bit which, if set to zero, indicates that the corresponding segment is of length 1 and otherwise that the associated segment length is recorded in the next byte or agreed number of bits. This solution gives an acceptable coding in the extreme cases where a large number of pixels differ in colour from all their neighbours.

6. QUADTREES

Quadtrees have been proposed and discussed by a number of authors as a method for compact pixel encoding.[12–15] Essentially the method is to divide an initial square repeatedly into four equal subsquares and to repeat this process until a subsquare is uniquely coloured. The collection of such subsquares is built into a tree structure which can then be compactly stored and reconstituted when required. Processing of quadtrees for certain raster graphics operations is also discussed by these authors. The principal disadvantages of this technique arise from the amount of scanning that has to be done to break down the pixels into coherent parts, the complexity of the algorithms to ensure efficient coding of the quadtrees and the fact that, in order to avoid serious complexity problems, the initial boundary of the picture needs to be a square with sides equal to a power of 2.

The quadtree compaction is obtained immediately as a consequence of a Hilbert scan, since the Hilbert polygon scans the whole space in successively larger squares. Each point is inspected once only, and in many cases adjacent squares are coalesced to form larger uni-coloured blocks. If exact quadtree representation for compaction is required, a minor modification to the scanning algorithm is all that is necessary. Further, the size of bounding rectangle is arbitrary, as has been indicated above.

Note that the above discussion relates to Hilbert polygons based on four successive subdivisions. Similar results follow with Peano polygons based on nine subdivisions and similarly for n^2 subdivisions, where the corresponding space-filling curve is of Peano or Hilbert type according to the parity of n.

7. SOME COMPACTION RESULTS

Table 3 summarises the results of applying Peano, Hilbert and normal linear (run-length encoding) compaction to the black-and-white pictures shown in Fig. 3–7. The quality of the diagrams is poor relative to normal reproduction since they have been reproduced using a dot-matrix printer to indicate the actual pixel colouring. Two numbers are given in each of the columns for Peano and Hilbert entries, being the number of segments required and the number of bytes required to store these segments using the compaction technique described

Table 3

Figure number	Peano segments	Peano bytes	Hilbert segments	Hilbert bytes	Linear segments
3	343	378	385	426	578
4	989	1070	967	1056	1248
5	2085	2165	2055	2141	2372
6	1939	2026	1633	1755	2376
7	1161	1223	1133	1190	1404

Table 4

Figure number	Peano	Hilbert	Linear	Minimum percentage of linear
3	245	251	578	42
4	683	603	1248	48
5	617	767	2372	26
6	1707	1427	2376	60
7	793	677	1404	48

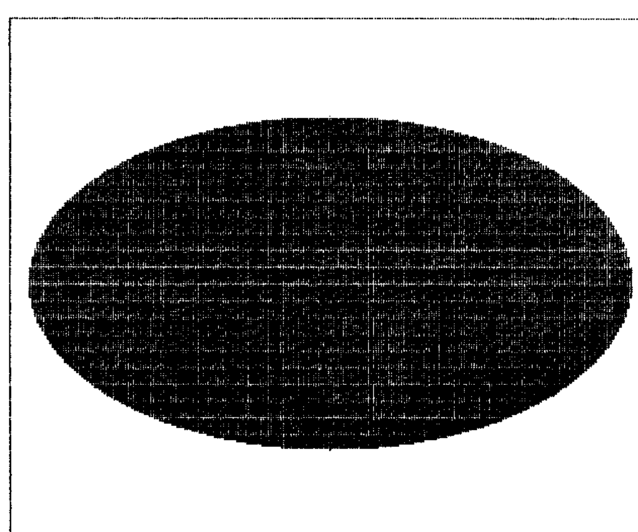

Figure 3

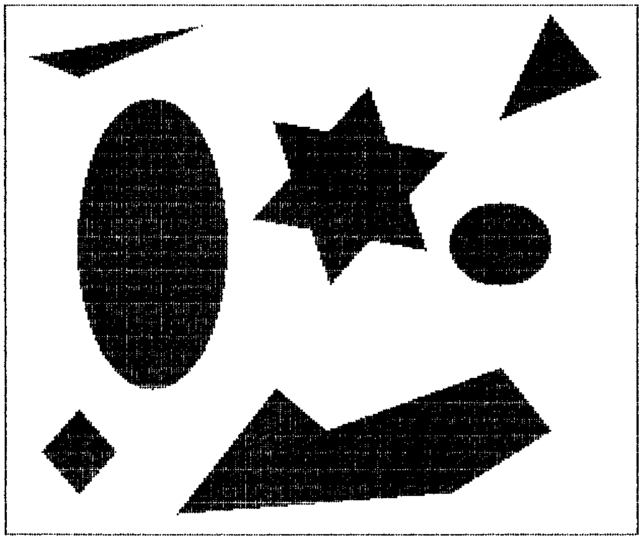

Figure 4

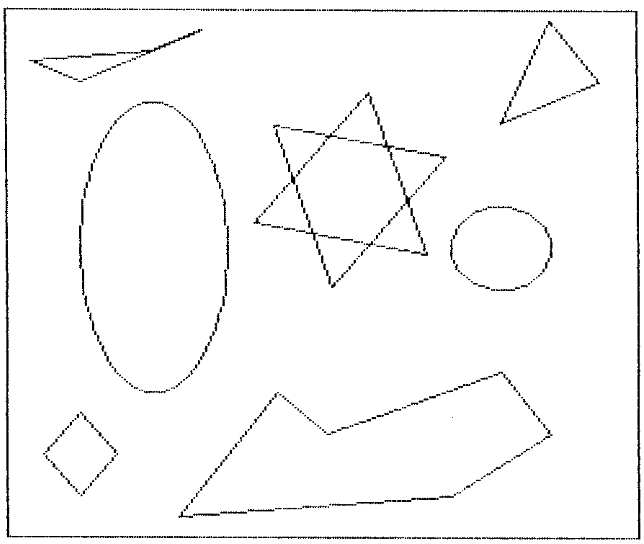

Figure 5

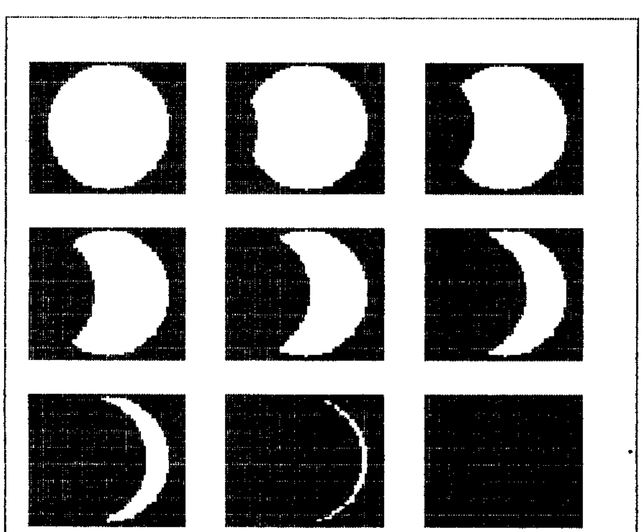

Figure 6

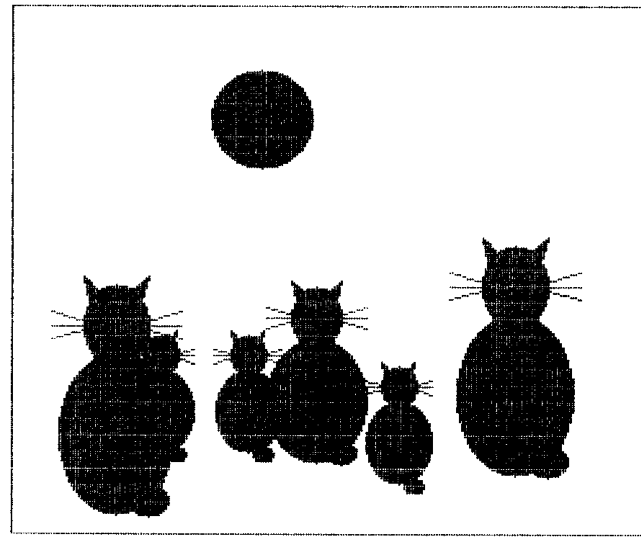

Figure 7

above. Any of the normal Huffman codings could also be used to further reduce these sets of numbers.

A further reduction with some loss of quality can be made by choosing a small integer n and replacing any three adjacent segment lengths a, n, b by $a+n+b$, which effectively changes the colour of the central segment to that of the surrounding segments. Table 4 summarises the results when segments of length 1 have been removed and similarly for Table 5 with segments of length 1, 2 and 3 successively removed. The quality of the pictures is illustrated in Figures 8, 9 and 10. Fig. 8 corresponds to the case in which segments of length 1 have been removed. In Fig. 9 the segments of length 1, 2 and 3 were

Table 5

Figure number	Peano	Hilbert	Linear	Minimum percentage of linear
7	1161	1133	1404	81
8	419	411	1404	29
9	353	309	1404	22

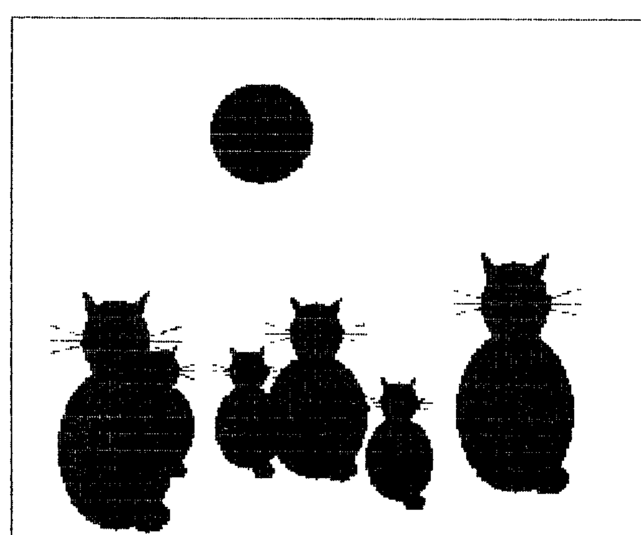

Figure 8

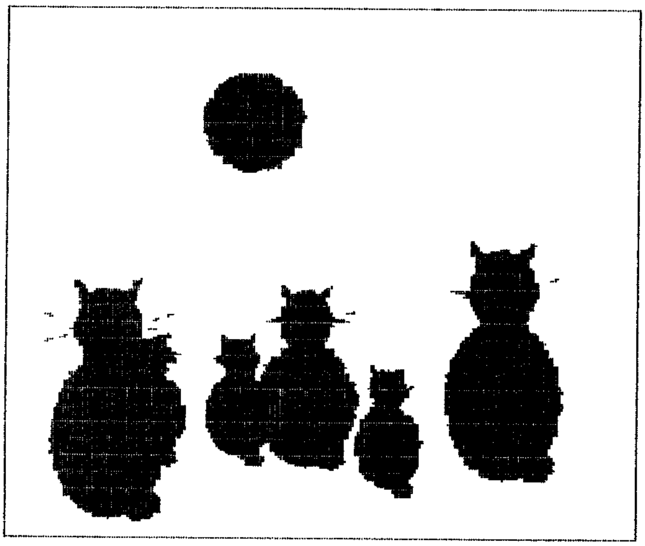

Figure 9

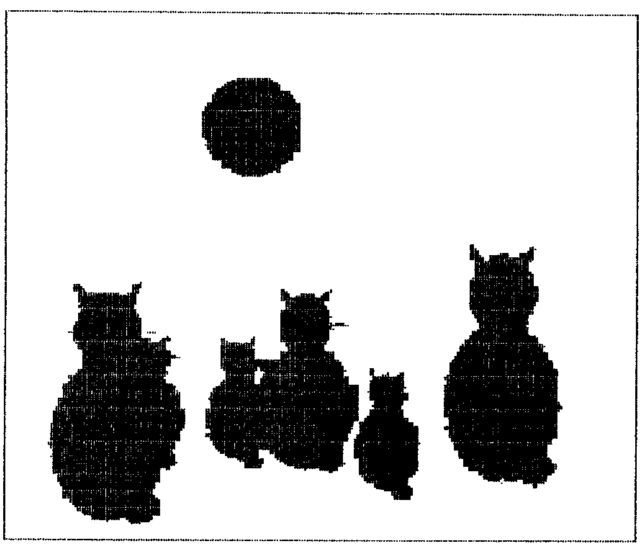

Figure 10

removed successively. In Fig. 10 the segments of length 1, 2 and 3 were removed simultaneously, with a significant loss of picture quality.

The reason why removal of segments of length 1 does not lead to greater picture distortion is that apart from truly isolated pixels the segments of length 1 are on the boundary of colour blocks and so it is only the boundaries which are, in general, distorted.

8. CONCLUSIONS

The use of space-filling curves in raster scan data compaction leads to data reduction of between 60 and 80% of the normal linear scan data. Quadtree representation is included as a special case. A simple reduction technique reduces the data to between 20 and 50% of the linear scan data without too serious reduction of picture quality.

There appears to be no special general advantage as between Peano and Hilbert scans, with one sometimes being marginally better than the other. In cases in which data reduction is important it might be advantageous to carry out both scans in parallel and to choose the smaller for transmission or storage. A leading code number could specify which algorithm should be used for reconstitution of the picture.

Computationally, both algorithms could be based on table lookup, but the direct Peano algorithm would probably be faster and more flexible if special hardware to carry out base 3 arithmetic was built.

The principal applications apart from data storage are in reduced data transmission and as an aid to pattern recognition systems. This follows from the localised scanning technique with a correspondingly simple identification of large blocks of similarly coloured pixels. For example, seven of the eight eclipses in Fig. 6 are immediately identified as being of likely interest from the segment data alone, and their limiting coordinates may also be easily calculated.

Acknowledgement

The author is indebted to the referee for helpful comments leading to some revision of this paper.

REFERENCES

1. G. Peano, Sur une courbe, qui remplit toute une aire plaine. *Mathematische Annalen* **36**, 157–160 (1890).
2. D. Hilbert, Ueber stetige abildung einer linie auf ein flachenstuck. *Mathematische Annalen* **38**, 459–460 (1891).
3. E. H. Moore, On certain crinkly curves. *Transactions of the American Mathematical Society* **1**, 72–90 (1900).
4. N. Wirth, *Algorithms + Data Structures = Programs*. Prentice-Hall, (1976).
5. L. M. Goldschlager, Short algorithms for space filling curves. *Software – Practice and Experience* **11**, 99–100 (1981).
6. I. H. Witten and B. Wyvill, On the generation and use of space filling curves. *Software – Practice and Experience* **6**, 519–525 (1983).
7. A. J. Cole, A note on space filling curves. *Software – Practice and Experience* **13**, 1181–1189 (1983).
8. W. Sierpinski, Sur une nouvelle courbe qui remplit toute une aire plaine. *Bulletin Academie Sciences Cracovie*, Series A, 462–478 (1912).
9. J. G. Griffiths, Table-driven algorithms for generating space-filling curves. *Computer-aided Design* **17**, 37–41 (1985).
10. A. J. Cole, A note on Peano polygons and Gray codes. *International Journal of Computer Mathematics*. **18**, 3–13 (1985).
11. A. J. Cole, Direct transformations between sets of integers and Hilbert polygons, *International Journal of Computer Mathematics*, submitted for publication (1985).
12. G. M. Hunter and K. Steiglitz, Linear transformation of pictures represented by quadtrees. *Computer Graphics and Image Processing* **10**, 289–296 (1979).
13. J. R. Woodwark, The explicit quadtree as a structure for computer graphics. *The Computer Journal* **25**, 235–238 (1982).
14. I. Gargantini, An effective way to represent quadtrees. *Communications of the ACM* **25**, 905–910 (1982).
15. M. A. Oliver and N. E. Wiseman, Operations on quadtree encoded images. *The Computer Journal* **26**, 83–91 (1983).
16. A. J. Cole, Cyclic progressive number systems. *Mathematical Gazette* **50**, 122–131 (1966).