MVA '96 IAPR Workshop on Machine Vision Applications, November. 12-14, 1996, Tokyo, Japan

Run-time Behavior of Contour Tracing Algorithms for Document Conversion

Paul C K Kwok* and Lok C Wong
Department of Electrical & Electronic Engineering
The University of Hong Kong

Abstract

In this paper, we present the run-time behavior of contour tracing algorithms for processing large documents, in a contemporary workstation environment. The performance criteria examined are processing time, memory usage, and the number of page faults.

A new contour tracing algorithm based on Capson's algorithm is proposed and it is compared with other algorithms. It is observed that for the set of document images under test, the new algorithm consumes the least amount of memory, has the smallest number of page faults and is slightly faster than Capson's algorithm.

1 Introduction

The digitization and conversion of line drawing documents has become an important activity in many organizations from an architect's office to utility companies. The motivation comes from the increasing rental cost and fire/water risks associated with the storage of hard copy documents accumulated by companies over the years. Up to now, automated conversion systems could not compete favorably with manual digitization and vectorization systems, in terms of quality, reliability and cost. Many document conversion projects have been contracted to China to take advantage of the low labor cost.

1.1 Line drawing conversion.

An automated conversion system [1] for line drawing documents consists of (i) a scanner with scanning and thresholding software, (ii) contour tracing program, (iii) vectorization program, to be followed optionally by high level recognition software, before encoding onto an output file. Each one of these components need to be carefully designed and fine tuned. This paper addresses the second component: contour tracing. Choosing the right contour tracing algorithm is important to ensure a low cost and reliable system.

Thus the role of contour tracing algorithms have shifted from one of preprocessing in pattern recognition tasks to one of document conversion. The amount of data involved in a pattern recognition task is insignificant by today's standard, when compared with the amount of main memory that is available is a typical contemporary workstation. On the other hand, in a document conversion task, the amount of image data depend on the size of the document and the resolution of the scanning device. The image data may occupy or even exceed all the available main memory found in the computer.

Despite the intensive research effort by academia on parallel computing and the parallelization of image processing algorithms, parallel computers are hard to come by in a production system. In the real world, the most cost effective platform to use for document conversion is a PC or a workstation. We have therefore chosen to evaluate contour tracing algorithms in a workstation environment.

Apart from the usual performance criteria of processing time and memory usage, the amount of page faults experienced by a contour tracing algorithm when processing a large bitmap is also an important performance indicator.

2. Contour Tracing Algorithms

Contour tracing is the front end software component whose input is the bitmap file from the scanner. The input file is usually compressed in some way, e.g. run-length encoded in Tiff format. Contour tracing provides further data reduction and outputs a chain coded file. There are two broad approaches to contour tracing. The first is serial and is based on Pavlidis's [2] algorithm. It requires random access to the bitmap. The other approach is non-sequential and is represented by Capson's algorithm [3]. Capson's approach examines two successive scan lines at a time. Random access to the bitmap is not required but it creates an intermediate dynamic linked structure whose size depends on the

* Address: Pokfulam Road, Hong Kong.
E-mail: kwok@hkueee.hku.hk

complexity of the image. A post-processing step is necessary to assemble the chain codes by traversing the linked structure. Locality of reference is poor during the traversal and it leads to many page faults.

A two pass algorithm, based on Capson's approach was proposed by Kwok [4] with an aim to reduce the number of page faults. During the first pass, the size of the resulting chain code output file and its structure are determined. Memory is allocated to accommodate the chain codes. In the second pass, the chain code is assembled directly onto the allocated memory space.

3. Table look-up method

In this paper, a further refinement to the algorithm in [4] is proposed to incorporate a table look-up method in the generation of chain codes.

Capson's paper gave a listing of the pseudo-code of the algorithm. It includes a series of nested IF-THEN-ELSE constructs to classify the adjacency relationships between two successive scan lines. It was discovered that in most cases, classification cannot proceed until the deepest conditional statement is reached. An alternative classification scheme is presented in this paper. It involves the assembly of a 3x2 window representing the relationship between a run in the current scan line and those in the previous scan line. After constructing the window, a table look-up procedure is used in the classification step.

The table look-up is indexed by the 3x2 window (6 bits), together with 2 extra bits to cater for open contours touching the top and bottom of the bitmap. Thus there are 256 entries in the table, but only 75 entries are defined. The other entries will not occur at all.

4. Evaluation of algorithms

Four algorithms are evaluated in this paper: Pavlidis, Capson, Kwok, and Kim *et. al.* [5]. Kim *et. al.* 's algorithm is included as a representative algorithm of the late 1980's. Kwok's algorithm is the two pass algorithm described in [4] with the table look-up method outlined in the last section. The evaluation platform is a Sparc Workstation with 128 MB main memory. It is a multi-tasking, multi-programming environment so only a fraction of the main memory is allocated to the contour tracing program. The programs are coded in C.

In a multi-tasking, multi-programming environment, the run-time statistics may vary slightly when the program is run a second time using the same data set. The tests are repeated a number of times to get an average performance indices. Care is taken so that every time the program is run, the memory has been flushed so that the program and data are loaded

4.1 Test images.

The test images are a collection of drawings found in an architect's office. These include maps, plans, views of buildings, etc. They are typically A3 size drawings digitised at 300 dpi, with about 35M pixel per image. The images are threshold by the scanner to become a binary image. If an algorithm requires keeping a copy of the image in the main memory, a packed data structure is used.

Figure 1 shows two of the many different types of architectural drawings (Note: Due to the reprographic process, these are low quality reproduction of the original high quality scanned images). On the left is a building. The drawing is 93% sparse, i.e., with only 7% foreground pixels. It has 6534 contours, 3416 of which with lengths less than 50 pixels. In the test set, most images have between 4% to 22% foreground pixels. In these images, over 50% of the contours have a length of less than 50 pixels.

4.2 Memory management.

The implementation of the algorithms include memory management module to request memory from the system whenever the need arises. The algorithms are tested for different sizes of memory (blocks) allocated per request, ranging from 64 bytes to 4K bytes. It was found that for all the algorithms, there is a general but insignificant decrease (less than one percent) in processing time as the block size is increased.

4.3 Memory usage.

Figure 2 shows the usage of memory for the four algorithms for 10 images of similar sizes, representing different types of drawings found in the architect's office. Pavlidis's algorithm consumes the largest amount of the memory, because of the need to store the entire bitmap. All the other algorithms make use of a dynamic workspace. Kwok's algorithm consumes the least amount of memory.

4.3 Number of page faults.

The number of non-compulsory page faults is the lowest for Kwok's algorithm, followed by Pavlidis's, Kim's, Capson, in this order. The number of page faults for Kwok's algorithm is one half that of Pavlidis's algorithm.

4.4 Processing time.

Processing time includes both user CPU time as well as system CPU time. It is smallest for Kwok's algorithm, which is slightly faster than Capson's for most cases; about 6 times faster than Kim's; and 17 times faster than Pavlidis's (see Figure 3).

4.5 Complexity of images.

The relationship between the performance of algorithms and the complexity of the images is established. It was discovered that in all the algorithms tested, with the exception of Pavlidis's algorithm, the total processing time is approximately proportional to the number of contours in the image. The processing time for Pavlidis's algorithm is almost constant for a given bitmap size.

5. Conclusion

Experimental results have demonstrated that different approaches to contour tracing have different run-time behavior. It was found that the serial approach requires the most memory and runs the slowest. On the other hand, the two pass Kwok's algorithm proposed in this paper has significant improvement over other algorithms in its use of memory, while maintaining the high speed of non-sequential algorithm, with the help of the table look-up method.

6. References

[1] P.C.K. Kwok and T.J. Turner. "Raster to vector conversion in a map interpretation system", *Proceedings of IAPR International Workshop on Machine Vision Applications*, IAPR, Tokyo, November, 1990, 165 - 168.

[2] T. Pavlidis. *Algorithms for graphics and image processing*, Computer Science Press, Rockville, Md, 1982.

[3] D.W. Capson. "An improved algorithm for the sequential extraction of boundaries from a raster scan", *Computer Vision, Graphics and Image Processing 28*, 1984, 109-125.

[4] P.C.K. Kwok. "Contour tracing of large bitmaps", *Proceedings of Asian Conference on Computer Vision '93*, Osaka, 1993, 309-312.

[5] S. Kim, J. Lee and J. Kim. "A new chain-coding algorithm for binary images using run-length codes", *Computer Vision, Graphics and Image Processing 41*, 1988, 114-128.

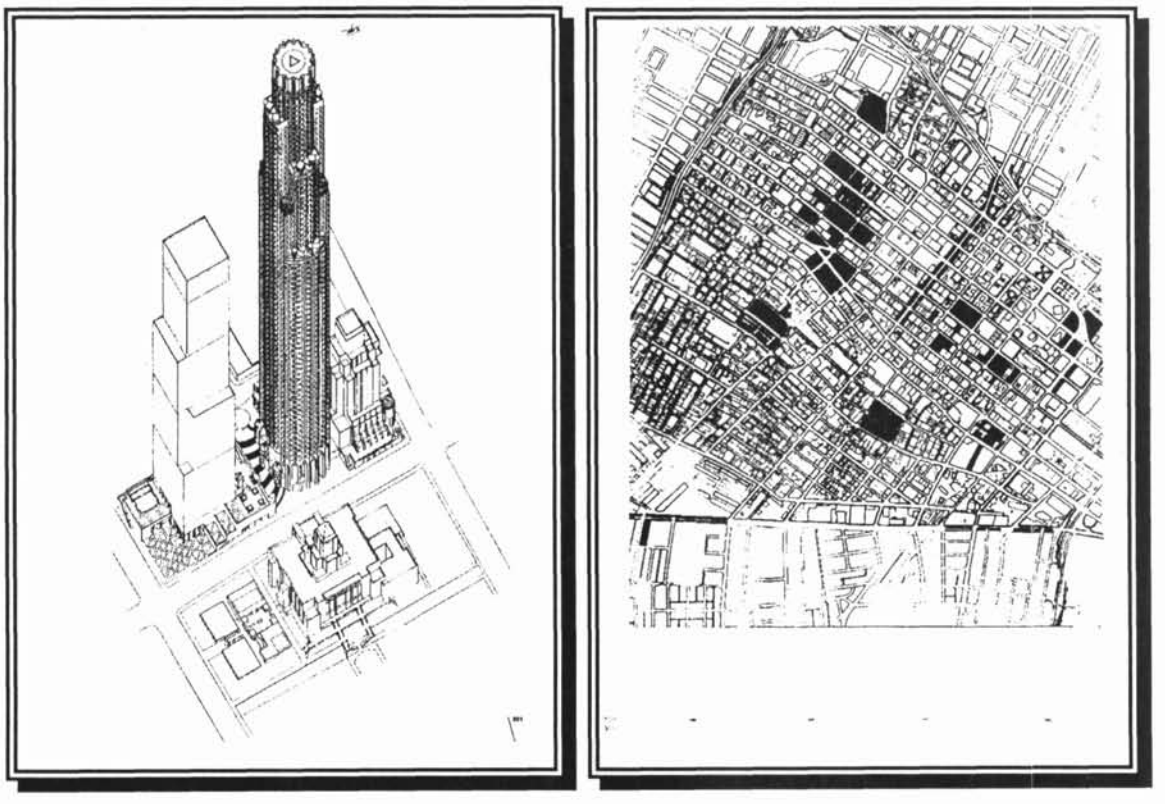

Figure 1. Two of the many types of drawings found in an architect's office. On the left: the aerial view of a building. On the right: The plan of the lots in a area.

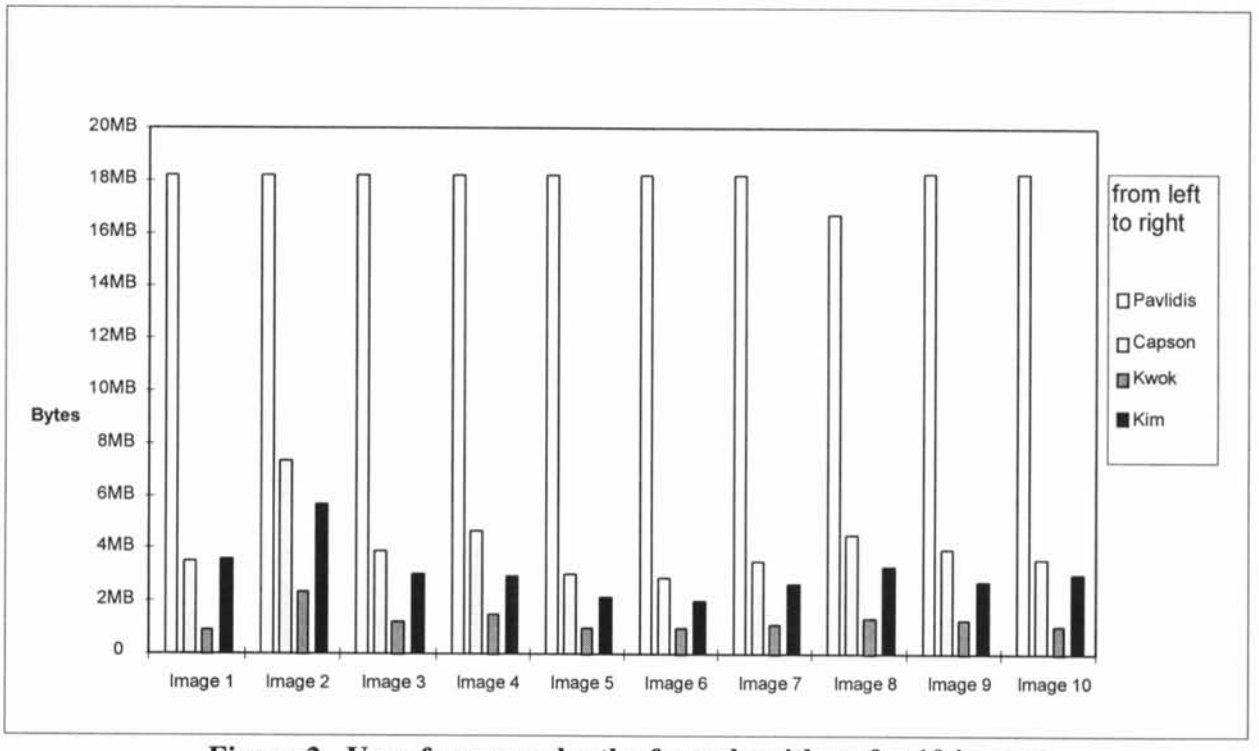

Figure 2. Use of memory by the four algorithms for 10 images.
(The bars from left to right are: Pavlidis, Capson, Kwok, Kim.)

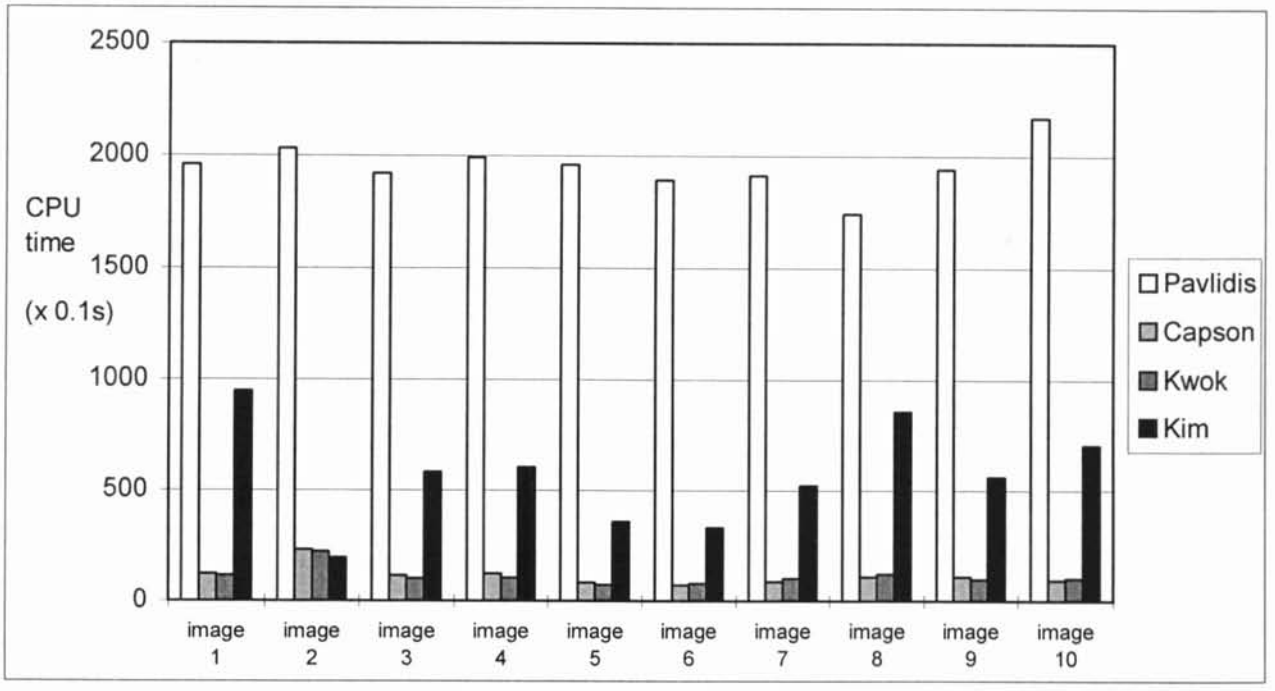

Figure 2. Processing times for the ten images.
(The bars from left to right are: Pavlidis, Capson, Kwok, Kim.)