A Two-step Algorithm and its Parallelization for the Generation of Minimum Containing Rectangles for Document Image Segmentation

Shamik Sural, NIIT Limited, 6B Pretoria Street, Calcutta 700 071, India.
P.K.Das, Dept. of Computer Science & Engineering, Jadavpur University, Calcutta 700 032, India.

Abstract

In document processing, segmentation is done to uniquely identify each foreground connected region of an image by specifying its minimum containing rectangle (MCR). MCR is the rectangle with minimum dimensions that completely encloses a geometric pattern. In this paper, we present a two-step MCR detection algorithm and its parallelization method. The first step determines the boundary of each connected component in a document image. This reduces resource requirements and speeds up the subsequent rectangle detection step. The rectangle detection step determines MCRs of the connected components from the detected boundaries. A comparison is made between a single-step and the two-step approaches of MCR detection. Both the boundary detection and the rectangle detection steps are parallelized and implemented on transputers to reduce the total processing time.

Keywords : Document image processing, Segmentation, Minimum containing rectangle, Parallel processing, Transputer.

1. Introduction

Determination of Minimum Containing Rectangles (MCR) of connected regions is required for both top-down and bottom-up approaches to document image segmentation [4-6,8]. Top-down is a knowledge-based approach in which a document image is recursively segmented into regions and sub-regions. A Run Length Smearing Algorithm (RLSA) is often used to connect black pixels which are separated by less than a threshold number of white pixels to generate a smeared image of the document [7]. The MCRs of the smeared regions are then determined for identification of the individual blocks. Bottom-up, on the other hand, is a data driven approach in which the first step is to identify the individual connected components like characters and images. Each connected component is represented by its MCR. There are, however, a few other techniques which follow a hybrid approach to document segmentation.

Most of the document processing applications require a real-time response. High-end sequential computers are normally used in an attempt to achieve this speed of computation. However, it is our observation, that, the problem of document image processing is inherently parallel in nature. Processing time can be greatly reduced by exploiting this parallelism in different stages.

In this paper, we propose a two-step approach for the determination of MCRs of connected document regions. In the next section, we describe the two-step MCR detection algorithm. Parallelization of the boundary detection and the rectangle detection steps are discussed in section 3. In section 4, we present the implementation results on a transputer network and draw conclusions from our work.

2. Two-step MCR detection algorithm

In the first step of MCR detection, the connected region boundaries are determined. We propose the following algorithm for boundary detection from a two-tone document image.

BEGIN ALGORITHM
FOR i = 1 TO no_of_scan_line
  FOR j = 1 TO no_of_pxl_per_scan_line
    IF pxl[i,j] = 1 AND pxl[i,j-1] = 0 *SET* pxl[i,j] = 0
    ELSE
    IF pxl[i,j] = 0 AND pxl[i,j-1] = 1 *SET* pxl[i,j-1] = 0
END ALGORITHM

Black pixels (represented by 1's) on each white-black horizontal transition of the image are replaced by white pixels (represented by 0's) at this stage and a new bitmap is generated. The above algorithm is repeated for the new bitmap, along each pixel position for all scan lines. Black pixels on each white-black transition, both horizontal and vertical, of the original image are replaced by white pixels and a transformed bitmap is generated. The pixel values of the original and the transformed bitmaps are then logically EXCLUSIVE-ORed to generate a boundary bitmap.

Once the boundary bitmap is generated, the number of pixels in the image is greatly reduced. We then detect the bounding box of each pattern boundary. The following

Fifth International Conference on Document Analysis and Recognition (ICDAR99), Bangalore, India, 1999, pp. 173-176

recursive algorithm is proposed to determine the MCRs from a boundary bitmap generated in the last step.

BEGIN ALGORITHM

```
FOR i = 1 TO no_of_scan_line
  FOR j = 1 TO no_of_pxl_per_scan_line
    IF pxl[i,j] = 1
      SET MAX_X = MIN_X = i
      SET MAX_Y = MIN_Y = j
      CALL PROC MCR(i,j)
PROC MCR (cur_x, cur_y)
BEGIN PROCEDURE
 FOR 8 ngbr_pxl OF pxl[cur_x, cur_y]
    IF ngbr_pxl = 1
      IF ngbr_pxl_x > MAX_X
        SET MAX_X= ngbr_pxl_x
      IF ngbr_pxl_x < MIN_X
        SET MIN_X = ngbr_pxl_x
      IF ngbr_pxl_y > MAX_Y
        SET MAX_Y = ngbr_pxl_y
      IF ngbr_pxl_y < MIN_Y
        SET MIN_Y = ngbr_pxl_y
      SET ngbr_pxl = 0
      CALL PROC MCR(ngbr_pxl_x, ngbr_pxl_y)
END PROCEDURE
```

END ALGORITHM

This algorithm uses local information around each black pixel to extend the MCR for the connected component. A number of other boundary detection and region labelling algorithms are available in the literature [1]. The proposed two-step algorithm has the advantage that it can detect any arbitrary shape and size of connected component boundary. Even in the presence of multiple pixels, none of the pixels is traversed more than once. The suggested rectangle detection algorithm does not involve searching and propagation, which is found to be time consuming. We later show how the proposed algorithm is also efficiently implemented on parallel processors.

The boundary detection step reduces the number of black pixels in each connected region. The rectangle detection algorithm, therefore, runs faster and requires considerably less system memory. However, the two-step approach works efficiently if the reduction in the number of black pixels is substantial in the boundary detection step. We make a performance comparison between the two-step approach and a single step approach in which MCRs are detected directly.

The total time for MCR detection in the single step method is $T_1 = t_r Ps$, where P is the total number of pixels and s is the fraction of the total number of pixels in the document image that are black. t_r is the average time for rectangle detection of a black pixel. The boundary detection time, T_{bd}, depends on the number of connected regions, n, in the document and is given by $T_{bd} = t_b n$, where t_b is the average time for boundary detection of a connected region. Assuming all the n connected regions to be square in shape and equal in size, the time for rectangle detection from the boundary bitmap is $T_{rd} = t_r 4n\sqrt{(Ps/n)}$. Here each square is of size $\sqrt{(Ps/n)} x \sqrt{(Ps/n)}$. The total time for MCR detection in the two step method is $T_2 = t_b n + t_r 4n\sqrt{(Ps/n)}$. The ratio of time required in the two-step and the single step method is $T_2/T_1 = (t_b/t_r)[n/(Ps)] + 4\sqrt{(n/Ps)}$. Thus, the improvement in total time for MCR detection is dependent both on the efficiency of the algorithm and the nature of the document. For a specific implementation of the algorithms, if the number of black pixels in the document increases keeping the number of connected regions unchanged, the ratio T_2/T_1 decreases, signifying a better performance of the two-step approach. Thus, for a larger document or for a document with higher number of black pixels, a two-step approach is better. Also, the performance of the two-step approach is better if there are a small number of connected regions, each containing a large number of black pixels. The reduction in the number of black pixels can then be done effectively by the pre-processing step. Thus, the performance of the two-step approach is expected to be better for MCR detection of run length smeared images than for bottom-up connected component analysis. The time required for a single-processor implementation of the MCR detection algorithm using the single-step and the two-step approaches for a scanned image and a smeared image is shown in table I.

Table I MCR detection performance in the single-step and the two-step approaches

No. of Step	Image Type	Boundary Detection Time (msec)	Rect. Detection Time (msec)	MCR Detection Time (msec)	Reduction (msec)
One	Original	------	19488	19488	------
One	Smeared	------	28828	28828	------
Two	Original	6792	11367	18159	1328
Two	Smeared	6679	9990	16669	12159

3. Parallelization of the MCR detection algorithm

We use a Multiple Instruction Multiple Data (MIMD) architecture [3] using transputers as parallel processors for MCR detection from document images. These processors execute multiple instructions in parallel on different data sets to reduce the total computation time. However, additional time is required for initial distribution and collection of data from all the processors. Inter-processor communication time is also required for data exchange during computation. If the computation to communication

time ratio is high, the effect of inter-processor communication overhead is less. A higher number of parallel processors can then be used to solve a problem [2].

For the boundary detection problem, we use a data partitioning in which there is an overlap of one scan line of pixels at each data set boundary to avoid the communication time overhead. Each processor can then detect all the pixels in its data set without any inter-processor communication during computation. The overhead of transferring one extra scan line of pixels for each data set boundary during initial data distribution is negligible.

The rectangle detection step is also parallelized by partitioning the image and submitting to a number of processors. However, the partitioning is done in such a way that no connected region overlaps two data sets since post-processing of pixels on the data set boundaries is difficult. Initially, the entire bitmap is partitioned into data sets, each with almost equal number of pixels. Each boundary is then checked to see if it is "clean". For a "clean" boundary, the scan line does not contain any black pixel. If a boundary is not clean, scan lines are searched both above and below the nominal boundary to find a clean scan line. When a clean scan line is found, the corresponding data set boundary is shifted there. The search is done for a certain maximum number of scan lines for each boundary. If a clean boundary is not found for a particular data set, it is added to the data set of the neighbouring processor.

3.1. Boundary detection time

For the data partitioning discussed above, the number of pixels allotted to the p^{th} processor is :

$$X(p) = P/N + 1 \quad \text{For } p = 1 \text{ and } N$$
$$= P/N + 2 \quad \text{For } p = 2,3,....,N-1 \qquad (1)$$

The number of pixels in a document image is usually very high compared to the number of processors so that,

$$X(p) = P/N \quad \text{For } p = 1,2,....N \qquad (2)$$

Since the N processors run in parallel, the total computation time for boundary detection is $T_{comp}(N)=t_b n/N$. Here we consider uniform distribution of connected regions in the document.

Let t_s be the setup time for data transfer and t_x be the inter-processor byte transfer time. Then, the communication time for data distribution to the p^{th} processor is

$$t_d(p) = t_s + t_x(P_h/8)(P_v/N + 1) \quad \text{For } p = 1 \text{ and } N$$
$$= t_s + t_x(P_h/8)(P_v/N + 2) \quad \text{For p= 2,3,....,N-1} \qquad (3)$$

P_h and P_v are the number of pixels per scan line and number of scan lines, respectively. Since most of the bitmap file formats store pixels along each scan line as bytes for two-tone images, a factor of eight is used to arrive at the number of bytes from the number of pixels. Again, since the number of pixels is high compared to the number of processors,

$$t_d(p) = t_s + t_x(P_h/8)(P_v/N) \quad \text{For } p = 1,2,...., N \qquad (4)$$

The total time for sequential data distribution to N processors is $t_{dd}(N)= Nt_s + t_x P/8$. Similarly, the data collection time at the end of computation is $t_{dc}(N)= Nt_s + t_x P/8$. Thus, the total communication time for the boundary detection step is $T_{comm}(N)= 2(Nt_s + t_x P/8)$.

The total time for boundary detection, $T_{bd}(N)$, is $T_{bd}(N)= t_b n/N + 2(Nt_s + t_x P/8)$. The number of processors for which the total boundary detection time is minimized, is $N_{min} = \sqrt{[(t_b n)/(2t_s)]}$. We see that if the setup time t_s increases, N_{min} decreases due to communication overhead. On the other hand, if the number of connected regions, or the average boundary detection time goes up, N_{min} increases. This implies that if the computation time is high compared to the communication overhead, a larger number of processors can be used to achieve a better response time.

3.2. Rectangle detection time

Let m_t be the maximum number of tries made for detection of a clean scan line and t_c be the average time required to check whether a scan line is clean. We make the worst case assumption that for each data set boundary, a clean scan line is obtained only after searching m_t number of scan lines both above and below the nominal boundary. We further assume that there exists at least one data set for which the upper boundary is shifted by m_t scan lines above as well as m_t scan lines below the nominal boundary. The computation time for clean scan line determination is $T_c(N)= t_c 2Nm_t$.

The computation time for rectangle detection for the processor with maximum number of pixels is $T_r(N)= (P/N + 2m_t)st_r$. The total computation time for rectangle detection, therefore, is $T_{comp}(N)=t_c 2Nm_t + (P/N + 2m_t)st_r$.

The communication time required for data distribution and data collection are the same as that for boundary detection and is given by $T_{comm}(N)=2(Nt_s + t_x P/8)$.

The total time for rectangle detection including the search time for detecting "clean" scan lines is $T_{rd}(N)= t_c 2Nm_t + (P/N + 2m_t)st_r + 2(Nt_s + t_x P/8)$. The number of processors for which the total rectangle detection time is optimized is $N_{min} = \sqrt{[(t_r sP)/2(t_s + t_c m_t)]}$. We see that if the setup time increases, N_{min} decreases due to increase in communication overhead. On the other hand, if the number of black pixels or the average rectangle detection time goes up, N_{min} increases. We also see that the time for finding clean scan lines for each data set boundary is a processing overhead. Decreasing the maximum number of tries reduces the search time but, in turn, increases the probability of improper load balancing among processors.

This value is, therefore, set after a large number of test runs.

3.3. MCR detection time

The total MCR detection time in the two-step approach with N processors running in parallel is $T_{MCR}(N) = A + 2N(2t_s + m_t t_c) + (nt_b + sPt_r)/N$, where $A = t_x P/2 + 2m_t st_r$ is a constant independent of N.
The number of processors for which the total MCR detection time is optimized, is $N_{min} = \sqrt{[(nt_b + sPt_r)/2(2t_s + m_t t_c)]}$. The interpretation of the above result is similar to that given for boundary detection and rectangle detection steps.

4. Implementation results and conclusions

We have implemented the two-step MCR detection algorithm on a T800 transputer-based system using OCCAM. The speedup variation for boundary detection, rectangle detection and complete MCR detection algorithms with the number of processors are plotted in figures 1, 2 and 3, respectively. We see that the boundary detection time is reduced almost by a factor of the number of processors. While the absolute time will depend on the processor used, an important observation is that the total computation time is high compared to the total communication time. This implies that one can go for a higher degree of parallelization to improve performance.

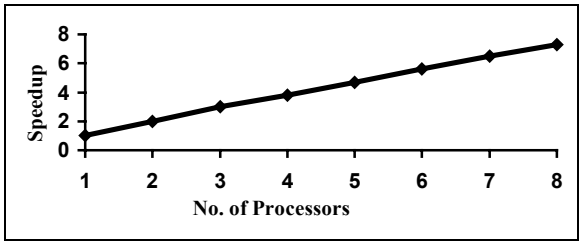

Figure 1. Speedup variation of the boundary detection algorithm

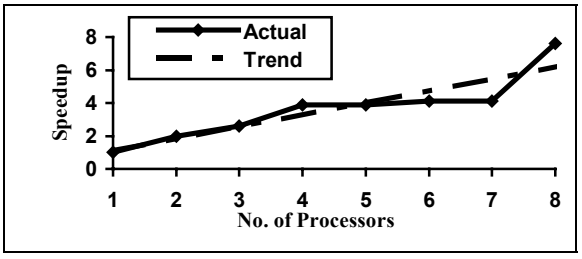

Figure 2. Speedup variation of the rectangle detection algorithm

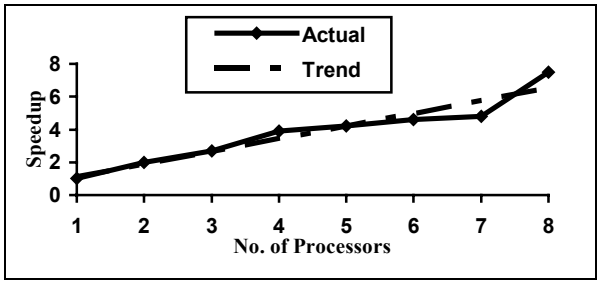

Figure 3. Speedup variation of the complete MCR detection algorithm

References

1. Chaudhuri, B.B. and D.Datta Majumder, "*Two-tone image processing and recognition*", Wiley Eastern, 1993, pp. 101-114.
2. Das, P. K. "Programming techniques and message passing mechanism in distributed memory computer system", *Proc. National Seminar on Parallel Computer Systems and Their Applications, Calcutta*, Oct. 1990, pp. 75-92.
3. Fox, G. C. et al, *Solving problems on concurrent processors*, Prentice Hall, USA, 1988, pp. 17-38.
4. Lu, Y. "Machine printed character segmentation - an overview", *Patt. Recogn.*, vol. 28, no. 1, 1995, pp. 67-80.
5. Nagy, G. "Teaching a computer to read", *Proc. 11th Int. Conf. Patt. Recogn. (ICPR)*, Aug. 1992, pp. 225-229.
6. Tang, Y. Y. and C. Y. Suen, "Document structures : A survey", *International J. Patt. Recogn. and Artificial Intelligence (IJPRAI)*, vol. 8, no. 5, 1994, pp. 1081-1111.
7. Wong, K. Y., R. G. Casey and F.M. Wahl, "Document analysis system", *IBM J. Res. Development*, vol. 26, no. 6, 1982, pp. 647-656.
8. Zlatopolsky, A.A. "Automated document segmentation", *Patt. Recogn. Lett.*, vol. 15, no. 7, 1994, pp. 699-704.