Sādhanā Vol. 29, Part 3, June 2004, pp. 329–341. © Printed in India

Document image mosaicing: A novel approach

G HEMANTHA KUMAR, P SHIVAKUMARA*, D S GURU and P NAGABHUSHAN

Department of Studies in Computer Science, Manasagangotri, University of Mysore, Mysore 570 006, India
*e-mail: hudempsk@yahoo.com

MS received 28 April 2003; revised 22 July 2003

Abstract. There are situations where it is not possible to capture large documents with the given imaging media such as scanners or copying machines in a single stretch because of their inherent limitations. This results in capture of a large document in terms of split components of a document image. Hence, the need is to mosaic the split components into the original and put together the document image.

In this paper, we present a novel and simple approach to mosaic two split images of a large document based on pixel value matching. The method compares the values of pixels in the columns of split images to identify the common or overlapping region (OR) in them, which helps in mosaicing of split images of a large document.

Keywords. Pixel matching; overlapping region; document image mosaicing.

1. Introduction

Many a time, it may not be possible to capture the complete image of a large document in a single exposure as most image-capturing media work with documents of definite size. In such cases, the document has to be scanned part by part producing split images. Thus, document image analysis and processing require mosaicing of the split images to obtain a complete final image of the document. Hence, document image mosaicing is the process of merging split images that are obtained by scanning different parts of single large document image with some sort of overlapping region (OR) to produce a single and complete image of the document. In this paper, we consider only two split images to obtain the final mosaiced image.

Several researchers have addressed different methods for obtaining the final image from its split parts. Schutte & Vossepoe (1995) and Szeliski (1996) have described the usage of flat bed scanners to capture large utility maps. The method selects the control points in different utility maps to find the displacement required for shifting from one map to the next. These control points are found from the pair of edges common to both the maps. However, the process requires human intervention to mask out the region not common to both the split images in image mosaicing.

Zappala *et al* (1997) and Peleg & Gee (1997) have worked on document image mosaicing (DIM). A feature-based approach through estimation of the motion from point correspondence

is proposed. They have exploited domain knowledge, instead of using generic corner features, to extract a more organized set of features. The exhaustive search adopted is computationally expensive because of the rotation of an image employed during matching. In addition, the method demands 50% OR in the split images to produce the mosaic image. However, the approaches are limited to only text documents and are prone to failure in case of general documents containing pictures. However, in practice, a typical document contains text, pictures and tables.

An automatic mosaicing process for split document images containing both texts and pictures based on a correlation technique is proposed by Whichello & Yan (1997) and Burt & Adelson (1983). Here, the correlation technique is used to determine the position of the best match in the split images. However, accuracy is lost at the edges of the images. Moreover, the correlation of two images of normal size is computationally very expensive. In order to find a solution, additional constraints like *a priori* knowledge is introduced. Here, the sequence in which the images were captured and their placement (generally, referred as image sequencing) is known. A template-matching procedure is used to search for the ORs present in the split document images. Usually, the template-matching procedure is a time-consuming method. In addition, this approach assumes that the printed text lies on straight and horizontal baselines, which is not always possible in many pragmatic applications.

Hence, in this paper, in order to circumvent the above mentioned drawbacks and also to work on all types of documents, a novel and simple approach is presented. The overlapping region helps in mosaicing two split images. Thus, the crux of the work is determining the OR in the split images efficiently and is achieved by comparing the values of pixels in the split images. The advantage of this approach is that no exhaustive search is required. The proposed method has the worst case complexity $O(n^3)$ where n is the number of columns in the split images. However, the method works based on assumption that the OR is present at the right end of the split image 1 and at the left end of the split image 2.

This paper is organized into four sections. Section 2 describes the proposed methodology to mosaic two split images. Section 3 explains the time complexity analysis of the proposed algorithms. Experimental results are reported in § 4 and conclusions are set out in § 5.

2. Proposed methodology

In this section, a novel approach based on comparing the values of pixels in the columns of the split images to mosaic them in order to produce a single, large document image is presented. Mosaicing is achieved by identifying the positions of OR in the split images. The OR are obtained by comparing the values of pixels in the columns of the split images. If the columns match, then the pointers j (where j is the pointer to columns in the split images) of both split images 1 and 2 are incremented by one. If there is no match, the pointer j of the split image 1 is moved to the next column, while the pointer j of the image 2 remains unaltered. This procedure is repeated till an overlapping region is found. The method works based on the assumption that the OR is present at the right and the left ends of the split images 1 and 2 respectively.

The proposed method is presented in three subsections. Subsection 2.1 presents a unidirectional method to mosaic two split components with $O(n^2)$ time complexity. Subsection 2.2 describes a bidirectional method to obtain the mosaiced image with time complexity $O(n^2)$. An improved unidirectional algorithm to mosaic the split images with time complexity of $O(n^3)$ is given in § 2.3.

2.1 *Algorithm 1: Unidirectional scanning*

Here we present an algorithm to obtain a mosaiced image from its split images that has the time complexity $O(n^2)$ based on comparing the values of pixels in each column of the split images of a large document. The algorithm compares all pixel values in the column of split image 1 with all pixel values in the column of split image 2. If the entire column vector of split images 1 and 2 matches, the pointers j which keep track of positions of columns in the split images, are moved to the next column. This procedure is repeated till the overlapping region is found in the split images. If the entire column vector does not match, then the pointer j of split image 1 is moved to the next column, while the pointer j of image 2 remains unaltered. This is possible because of the assumption that the OR is present at the right and left ends of split images 1 and 2 respectively. However, the algorithm fails to identify the actual OR when the split image 1 contains more OR when compared to that in image 2 (see figure 8 in § 4). Figure 1 shows the process of obtaining a mosaiced image from two split images S_1 and S_2. In which n and m represent number of columns and number rows of split images. Pointers i and j of S_1 and S_2 respectively keep track of the positions of the rows and columns of images. The pointer j of S_1 is moved towards right as shown in figure 1 to find OR by comparing with the jth column of S_2. If the entire columns of S_1 and S_2 being compared match, then the pointers j of S_1 and j of S_2 are moved to the next column incrementing by one. Otherwise, only the pointer j of S_1 is moved to next column towards right. The pointer j of S_2 remains in the first column position till it gets matched. The algorithm terminates when the pointer j of S_1 reaches n.

Algorithm 1

Input: Split images 1 (S_1) and 2 (S_2).

Output: Mosaiced image.

Method: Step 1 – For each column (j) of S_1 and S_2

For each row (i) of S_1 and S_2

If $(S_1(i, j) = S_2(i, j))$

$i = i + 1$, in both S_1 and S_2 (the pointer i varies from 1 to m and j varies from 1 to n. Here, S_1 and S_2 are of size $m \times n$ where m and n represent number of rows and column)

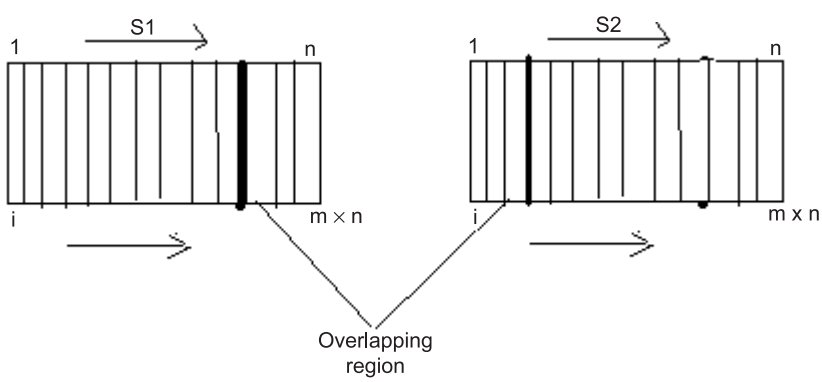

Figure 1. Mosaicing of two split images with $O(n^2)$ time complexity.

If whole column matches then CM = true if end

Else $j = j + 1$ in S_1 and, i and j of S_2 are again initialized to 0 (first column) (here, i and j of S_2 remains in the first column but j of S_1 moves to next column)
If end.

For ends.

Step 2 – If $(i = m)$ and (CM = true) then $j = j + 1$ in both S_1 and S_2 (here, if whole column matches then increment both the pointers in S_1 and S_2)
Else
For ends

Step 3 – If $(j = n)$ and (CM = true) then OR = 1 (if pointer j pointing to column in S_1 reaches n with some matches (CM = true) then overlapping is found)
Else OR = 0 (overlapping region is not found)

Step 4 – If OR = 1
Translate the split image 2 such that the overlapping regions of both the images match each other with respect to their coordinates.

Else no overlapping region exists

Algorithm 1 ends.

2.2 *Algorithm 2 – Bidirectional scanning*

In this section, another method which overcomes the drawback of algorithm 1, but has the same time complexity, is presented. Algorithm 2 is thus just an extension of previous algorithm 1. The method scans the split images from right to left as well as left to right, whereas in algorithm 1 scanning of the image takes place only from left to right to identify the overlapping region in the split images (see figure 2). Algorithm 2 finds the distance (D_1) between the first column and the starting matching area, while scanning left to right as in the case of algorithm 1. Similarly, it computes the distance (D_2) from the first column of the split image 1 to the beginning position of OR while scanning the split images from right to left. This is shown in figure 2. The actual overlapping region starts from position D_2 which is at smaller distance when compared to D_1 from the right end of split image 1. However, algorithm 2 fails to handle the situation shown in figure 3. In this case, actual match starts from 4th column of

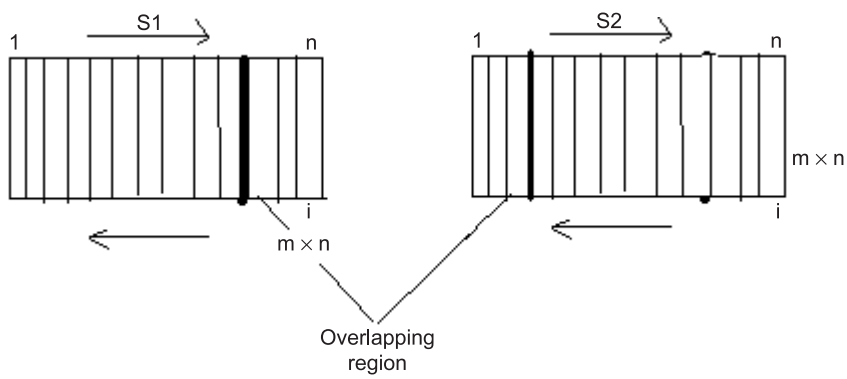

Figure 2. Mosaicing of two split images with $O(n^2)$ time complexity by back tracking.

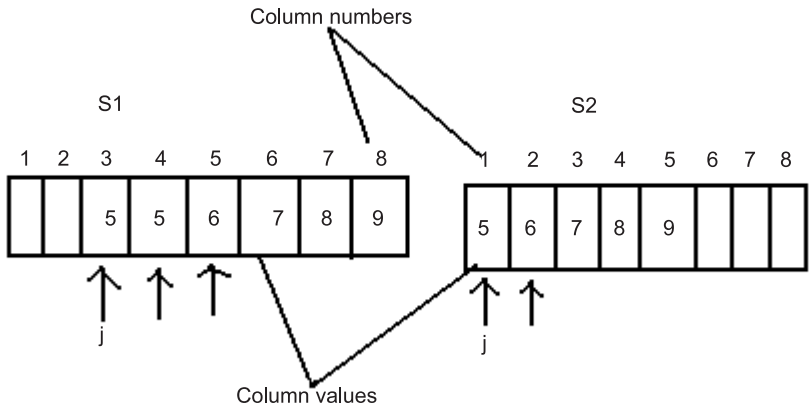

Figure 3. The situations where algorithm 2 fails.

S_1 but if we employ the algorithm 2 it gives no match though there is an OR in the S_1 and S_2. This is because the algorithm 2 matches 3rd column of S_1 with first column of S_2. Both the pointers are incremented by one since match exists. Now the pointer j of S_1 is pointing to 4th column of S_1 and the pointer j of S_2 is pointing to second column of S_2. There is no match in these positions. Then the pointer j of S_1 is moved to 6th column of S_1, meanwhile the pointer j of S_2 comes back to first column position. The algorithm continues to find match by incrementing the pointer j in S_1. This results in no OR in the split images. In figure 3, the values 5–9 are assumed to be values in the respective columns that means the whole column is matching. This conflict could be resolved by modifying algorithm 2 and is presented in § 2.3.

Algorithm 2

Input: Split images 1 (S_1) and 2 (S_2).

Output: Mosaiced image.

Method: Step 1 – Determine the distance (D_1) between the first column and the beginning position of OR in S_1 by employing algorithm 1.

Step 2 – Determine the distance (D_2) between the first column and the beginning position of OR in S_1 by employing algorithm 1 and by changing pointer positions i and j of S_1 start from the nth column in S_1, and i and j of S_2 starting from the nth column of S_2.

Step 3 – If ($D_2 < D_1$) then the D_2th position is the beginning position of actual OR in S_1.

Step 4 – Translate the split image 2 such that the overlapping regions of both the images match each other with respect to their coordinates.

Algorithm 2 ends.

2.3 *Algorithm 3 – An improved unidirectional scanning*

In this section we present an algorithm to mosaic split images. This algorithm works for all types of documents. The algorithm 3 uses algorithm 1 to mosaic split images for different cases. For each column the algorithm 3 invokes algorithm 1 to identify the OR in the split images. If the pointer of S_1 of invoked algorithm reaches end of split image 1 with OR then the

method terminates successfully. Otherwise, the method again invokes algorithm 1 from next column of the split image 1. This procedure is continued till it terminates successfully. If we consider the situation in Figure 3 where algorithm 3 invokes algorithm 1 from third column then the pointer j of S_1 moves to 5th column since match exist in third column. Meanwhile in S_2 the pointer j moves to 2nd column. Finally, the pointer j of S_1 reaches n without OR since no match exists. At this stage the number of matches becomes nil and the pointer j of S_1 reaches n of S_1. In this situation the algorithm 3 invokes the algorithm 1 from 4th column of S_1 and the pointer j of S_2 will move back to first column of S_2. Hence, the algorithm 3 successfully gives the results as the count (number of matches) is not equal to zero and the pointer j of S_1 reaches n of S_1. However, it requires $O(n^3)$ computational time in the worst case.

Algorithm 3

Input: Split image 1 (S_1) and Split image 2 (S_2)

Output: Mosaiced image

Method: Step 1 – For each column invoke algorithm 1 to identify the overlapping region in the split images.

(a) If the column of S_1 doesnot match with the column of S_2. Then note down the values of count and i.
(b) If count $\neq 0$ and $j = n$ then (where count is number of matches) overlapping region is found

Else go to next column

For end

Step 2 – Repeat step 1 till algorithm 1 satisfies the above conditions (b).
Step 3 – Translate the split image 2 such that the overlapping regions of both the images match each other with respect to their coordinates.

Algorithm 3 ends.

3. Complexity analysis

The major step involved in document image mosaicing is to identify the overlapping region in the split images efficiently. Hence the complexity of the proposed algorithm depends on complexity of the method of finding the overlapping region in the split images. In case of algorithm 1, the time complexity is $O(n^2)$ in the worst case because the pointer i pointing to row in S_1 runs from 1 to m of S_1 and the pointer j of S_1 pointing to column runs from 1 to n of S_1. Hence, the time complexity of algorithm 1 in the worst case which is $O(n \times m)$, and becomes $O(n^2)$ (since $n = m$), where n is the number of columns in the split images. The worst case is when there is no overlapping region in the split images. In this case, each and every element has to be considered for matching purpose to identify the overlapping region. Obviously, if the matrix is of size $n \times n$ then the time complexity of any algorithm using this matrix is of order $O(n^2)$. However, algorithm 1 fails when the OR in split image 1 is greater than in split image 2 (see figure 8 below).

In order to overcome the drawback of algorithm 1 we propose algorithm 2. But even this algorithm in the worst case has time complexity $O(n^2)$. This is because algorithm 2

Table 1. Comparative study of three methods with respect to an English document (figure 4).

Algorithms	Number of comparisons	Time complexity
Algorithm 1	2344	$O(n^2)$
Algorithm 2	4581	$O(n^2)$
Algorithm 3	63531	$O(n^3)$

employs algorithm 1 twice. That is, in order to find D_1 and D_2 (changing the direction of the pointers) algorithm 2 employs algorithm 1 twice. Hence the time complexity of algorithm 2 is $O(n^2) + O(n^2) = O(n^2)$ if we neglect the constant 2. However, algorithm 2 fails in the situation as shown in figure 3.

Hence algorithm 3 which overcomes the drawbacks of both algorithms 1 and 2 is presented. Algorithm 3 employs Algorithm 1 for each column of the split images, which is of time complexity $O(n^2)$. As algorithm 3 employs algorithm 1 n times in the worst case then the time complexity of the algorithm 3 becomes $n \times n \times n = O(n^3)$ (Aho *et al* 1974). This could be further reduced by modifying the algorithm (this however is beyond the scope of this paper). It is observed that from table 1, the number of comparisons increases as the complexity of the algorithm increases.

4. Experimental results

In this section, only a few out of many experimental results are given. For each algorithm we present different cases which ensure that our proposed algorithm works for any type of document irrespective of its content. In the following examples shown in figures 4–15, figure a and figure b are two input images. Figure c is the overlapping image and figure d is the final mosaiced image.

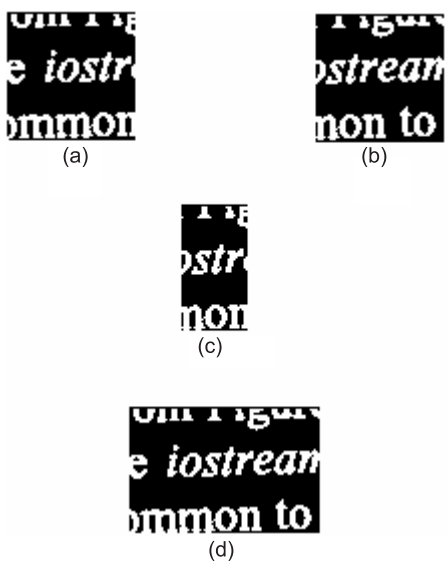

Figure 4. Text document image.

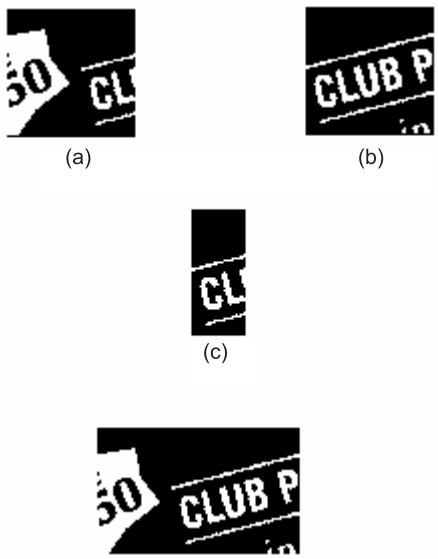

Figure 5. Text embedded within the lines.

Case 1: The experimental results of algorithm 1 do not conform to the expected results which are shown in figure 8. The input split image 1 (figure 8a) contains more OR compared to split image 2 (figure 8b). In such situations the proposed algorithm 1 fails to give the complete overlapping region present in both the images, which is the major drawback of this method. Hence, an improved version of this method is presented in case 2.

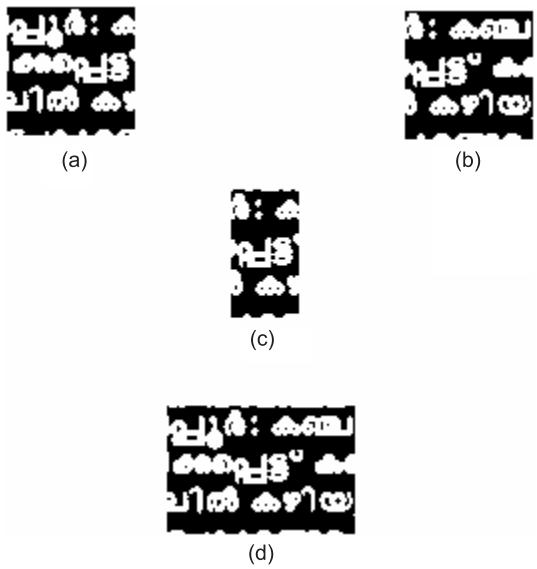

Figure 6. Malayalam text document image.

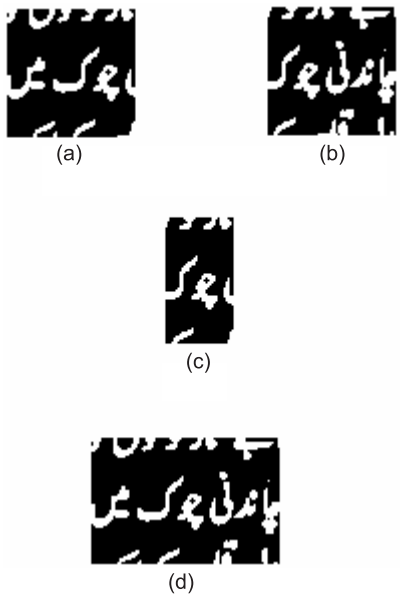

Figure 7. Urdu text document image.

Case 2: Algorithm 2 is an improved version of algorithm 1 which eliminates the drawbacks of algorithm 1. This algorithm works fine if the split image 1 contains more overlapping region as compared to split image 2 or vice versa, with the same time complexity. This is shown in figures 12 and 13. With reference to the previous case this algorithm works satisfactorily, in particular for the inputs shown in figure 8. In figure 12 we notice that the algorithm identifies the complete overlapping region in the split images. However, the algorithm fails in a situation as shown in figure 3.

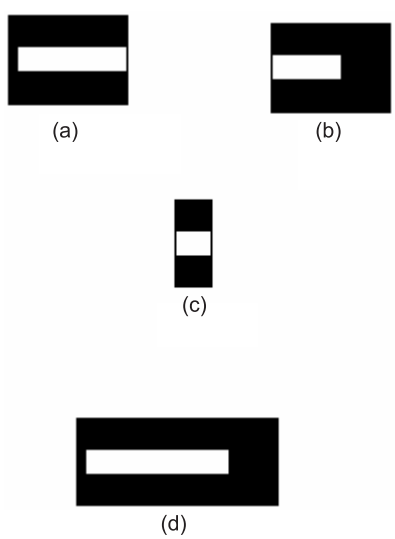

Figure 8. Synthetic document image 1.

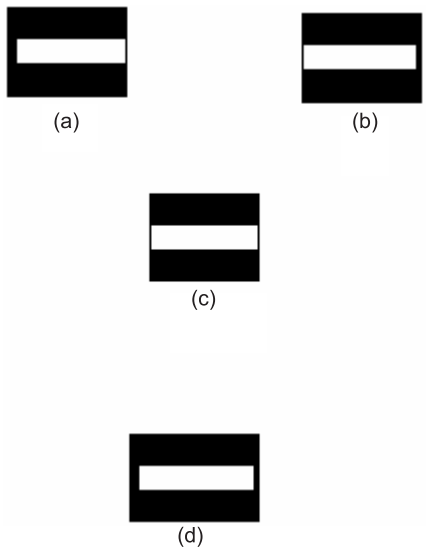

Figure 9. Synthetic document image 2.

Case 3: We present the experimental results of algorithm 3 which works on any type of document irrespective of its content and overcomes the drawbacks of both algorithms 1 and 2. For an instance, algorithm 3 identifies the complete overlapping region in the split images. This is shown in figures 14 and 15.

5. Conclusion

In this paper, a novel and simple approach for mosaicing of two split images of a large document to produce a single large document is presented. The approaches presented in this

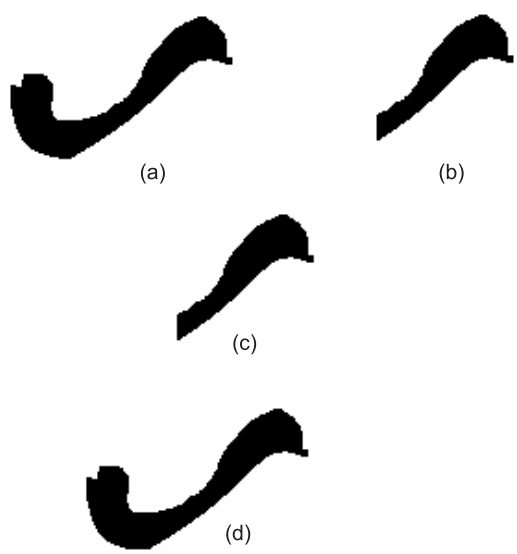

Figure 10. Irregular shaped image 1.

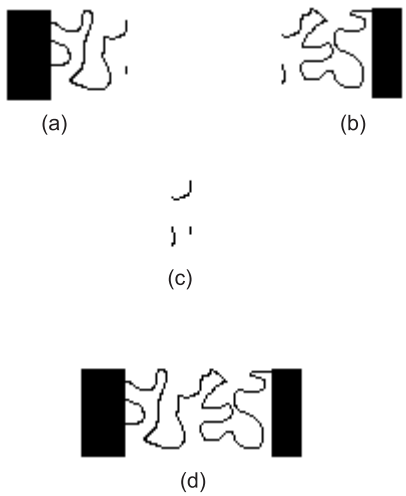

Figure 11. Irregular shaped document 2.

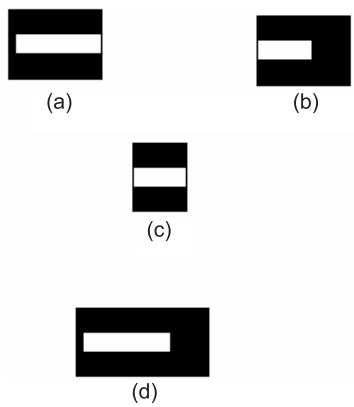

Figure 12. Synthetic document image 3.

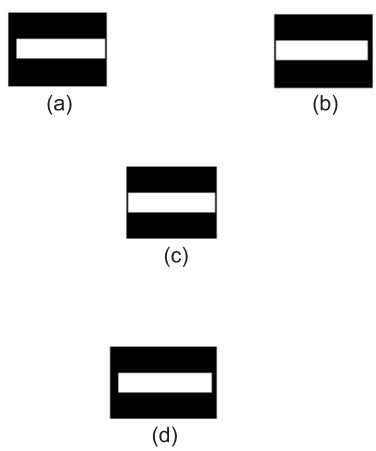

Figure 13. Synthetic document image 4.

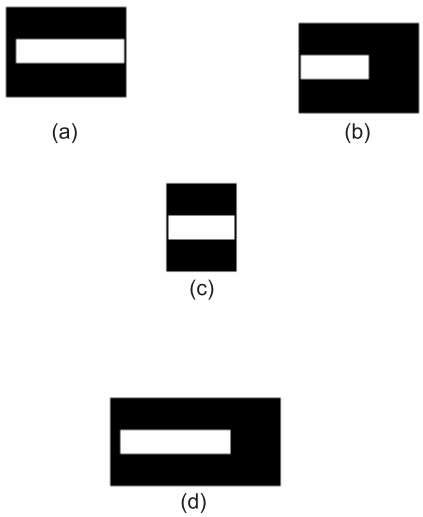

Figure 14. Synthetic document image 5.

paper match pixel levels in the split images to obtain a mosaiced image from its split images. The complexity of each algorithm is discussed. The proposed approaches assume that the overlapping region is present at the right and the left ends of split images 1 and 2 respectively. In addition to this, the method expects an overlapping region that is at least one pixel wide in the split images. The methods work fine for all types of documents but they consume time and fail if the sequence is missed.

One of the authors (PS) thanks the All India Council for Technical Education, New Delhi for a fellowship.

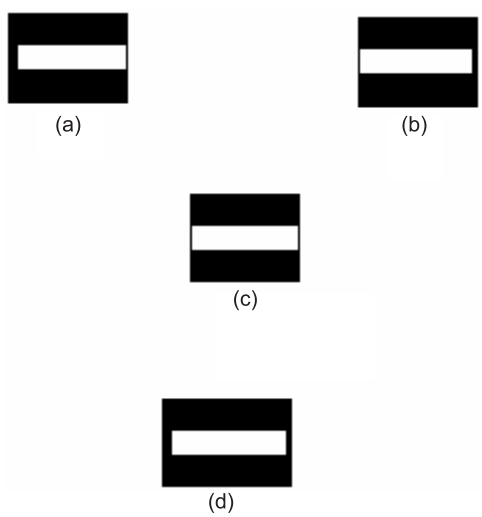

Figure 15. Synthetic document image 6.

References

Aho A V, Ullman J D, Hopcroft J E 1974 *The design and analysis of computer algorithms* (Cambridge, MA: Addison-Wesley)

Burt P, Adelson E 1983 A multiresolution spline with applications to image mosaics. *ACM Trans. Graphics* G-2: 217–236

Peleg S, Gee A H 1997 Virtual cameras using image mosaicing. Haifa Research Laboratory, Hebrew University, Jerusalem

Schutte K, Vossepoel A 1995 Accurate mosaicing of scanned maps of how to generate a virtual A_o scannar. *Proc. First Annual Conf. of the Advanced School for Computing and Imaging,* Heijen, the Netherlands, pp 353–359

Szelski R 1996 Video mosaic for virtual environments, *IEEE Comput. Graphics Appl.* 22–30

Whichello A P, Yan H 1997 Document image mosaicing, Science and Engineering Laboratory, Department of Electrical Engineering, University of Sydney, NSW 2006. www.bmva.ac.uk/bmvc/1997/papers/046/paper_046.html

Zappala A R, Gee A H, Taylor M J 1997 Document mosaicing. *Int. Proc. British Machine Vision Conference*, Colchester 2: 600–609