Volume 2, Issue 5, May 2012 ISSN: 2277 128X
International Journal of Advanced Research in Computer Science and Software Engineering
Research Paper
Available online at: www.ijarcsse.com

Reconstruction of 2D Image Fragments

Richa Mishra
Assistant Professor,
Dept. of Computer Science,
GCET, Greater Noida.
richa36@gmail.com

Saurabh Tripathi
Student,
Dept. of Computer Science,
GCET, Greater Noida.
saurabhcs85@rediffmail.com

Prem Prakash Patel
Student,
Dept. of Computer Science,
GCET, Greater Noida.
prempatel2447@gmail.com

Abstract— **Digital images may be considered as collection of pixels. If a single image is divided into more than one part, then these subparts are treated as fragments for an image. Joining of 2D fragments of an in image means we have to reassemble these image's fragments. The joining of fragments to reconstruct images and objects is a problem associated in several applications, like archeology, medicine, art restoration, and forensics .We mainly focused on 2D Image Reconstruction by joining two 2D fragments. This approach is based on the information generated from the boundary. Local curvature is calculated to obtain the lines in order to find the angle between them to rotate the second fragment. Based on the information of the boundary comparison is done to obtain maximum matching parts among fragments. Finally longest matching parts can be joined to obtain single image. The techniques illustrated in this paper constitute the core of a more general method for reassembling *n* fragments we are developing.**

Keywords— **Archeology, Boundary pixels, Chain Code, Curvature, Rotation , Digital images, Fragments**

1. INTRODUCTION

The reassembly of fragments to recompose images and objects is a problem occurring in a number of fields: in archeology, in art restoration, and in other disciplines including forensics, computer - aided design, chemistry, and medicine. In this paper we present a novel method for reassembling two 2D fragments. We give only few preliminary considerations on the more general problem (on which we are currently working) of integrating *n* fragments to reconstruct a complete image.

We proposed the reconstruction method for two cases-
1. When the fragments are aligned.
2. When the fragments are non-aligned.

The outline of the proposed method for reassembling two fragments follows.
(1) Finding the boundary of the fragments
(2) Analysis of the two fragments and extraction of their representations. A fragment is represented by the sequence of its border pixels with associated information about coordinates.
(3) If the fragments are non- aligned, analyse the local curvature, the relative angle between the fragments and finally rotate the second fragment so that it is aligned to the first one.
(4) Comparison of the two sequences to identify the common sub sequences. The common subsequences represent the candidate matching portions of the borders.
(5) Calculation of the transformation (translation in both x and y co-ordinates) that brings the first fragment to join the second one along the best (longest) matching portion of the borders.

2. PROPOSED APPROACH FOR ALIGNED FRAGMENTS

The approach for the joining of the images when both fragments are aligned is following-
i. Find the boundary of the fragments.
ii. Find boundary array using chain code.
iii. Find longest common sub sequences using fragment matching algorithm.
iv. Join the two fragments according to these longest common sub sequences.

2.1. BOUNDARY OF THE FRAGMENTS

The very first approach is to find the boundary of the fragments. We can use so many of operators to do this like 'Robert ', 'Sobel', 'Prewitt', 'Compass Operators' , ' Canny ' etc. Canny operator is one the best operator in finding the boundary. The boundary pixels have intensity 1 and rest of the pixels in the fragments is 0.

2.2. FRAGMENT ANALYSIS (CHAIN CODE)

In his section, we find the boundary array using chain code. We calculate chain code of the fragment by moving from one boundary pixel to another. The next boundary pixel is located by searching among the eight neighbours of the current boundary pixel. The result is stored in a boundary array containing the co ordinates of every pixel belonging to the fragment's boundary.
Chain codes are used to represent the boundary of an object composed of pixels of regular cells by connected sequence of

straight-line segments of specified length and direction. The object is traversed in clockwise manner. As the boundary is traversed, the direction of each chain segments is specified using the following numbering scheme:

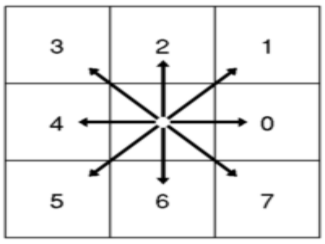

Fig 1. Chain Code directions

x

x-1, y-1	x, y-1	x+1, y-1
x-1, y	x,y	x+1,y
x-1, y+1	x, y+1	x+1, y+1

y

Fig2. Pixel positions in an image

The chain code provides a storage-efficient representation for the boundary of an object in a binary image. The chain code representation incorporates such pertinent information as the length of the boundary of the encoded object, its area, and moments. Chain codes lend to efficient calculation of certain curve parameters. Additionally, chain codes are invertible in that an object can be reconstructed from its chain code representation. The basic idea behind the chain code is that each boundary pixel of an object has an adjacent boundary pixel neighbour whose direction from the given boundary pixel can be specified by a unique number between 0 and 7 (8-connectivity neighbourhood).

ALGORITHM: CHAIN CODE

1. Initialize empty chain code array.
2. Choose a pixel with intensity 1.
3. Check the next pixel for8 connectivity
4. The next pixel with value 1 should not be the previous to previous pixel of the chain code.
5. Make entry in the chain code array and increment its pointer.
6. Do step3 to step5 until we reach to the starting point.

2.3. FRAGMENT MATCHING ALGORITHM

This section we describe fragment matching algorithm. Input for this algorithm consists of two border pixel arrays called shape1 and shape2 in the form of chain codes. Their outputs are start points and endpoints of the longest matching subsequences.

1. Obtain border arrays (for both shapes it should be in different directions i.e. if clockwise for the shape1 then it should be anticlockwise for shape 2).
2. Find long subsequences appearing in both shapes.
3. The outer loop cycles the different placements of arrays. Initially position is fig.3, then Shape2 is shifted by one position fig.4 and so on until the end of the loop fig. 5.
4. The number of evaluated placements is equal to the length of Shape1.
5. For each single placement, these arrays are compared element by element (pixel by pixel) from left to right. The total number of element comparisons for each placement is equal to the double of length of Shape 2.

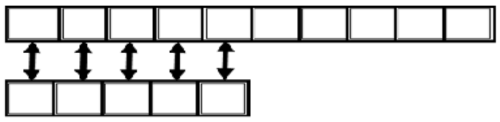

Fig3. Initial position of two shapes

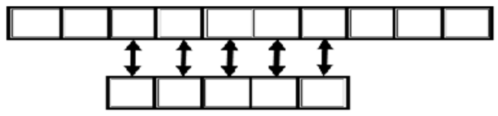

Fig4. WhenShape2 is shifted by certain position

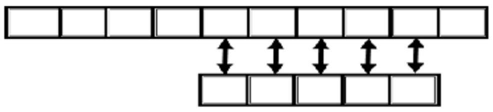

Fig5. When Almost Comparison has been done

ALGORITHM: FRAGMENT MATCHING

```
1.  Initialize i=1,j=1
2.  while true
3.      for k=0 to tol  do
4.          if shape1( i +k) == shape2(j)  then
5.              xstart(xstart_next)  ← xshape1(i+k)
6.              ystart(ystart_next)  ← yshape1(i+k)
7.              x2start(x2start_next) ← xshape2(j)
8.              y2start(y2start_next) ← yshape2(j)
9.              i=i+k, i++, j++ ,
10.             goto 18.
11.         elseif shape1(i) == shape2(j+k)  then
12.              xstart(xstart_next) ← xshape1(i)
13.              ystart(ystart_next) ← yshape1(i)
14.              x2start(x2start_next) ← xshape2(j+k)
```

```
15.             y2start(y2start_next) ←yshape2(j+k);
16.             j=j+k, i++, j++
17.             goto 18
18.         end if
19.     end for loop
20.
21.      if  k>tol  then
22.         i++ , j++,
23.         goto  3
24.     end if
25.     while true
26.         if shape1(i) == shape2(j)  then
27.            i++, j++ ,
28.         else
29.            xend(xend_next)  ← xshape1(i)
30.            yend(yend_next)  ← yshape1(i)
31.            x2end(x2end_next)  ← xshape2(j)
32.            y2end(y2end_next)  ←yshape2(j)
33.            goto 35
34.         end if
35.     end while loop
36.
37.     if  i == length(shape1)   or   j == length(shape2)
    then
38.         goto 40
39.      end if
40. end while loop
```

The output of the fragment matching algorithm is the starting and ending points of the longest common sub sequences between two fragments.

2.4 JOINING OF FRAGMENTS

The two fragments are joined according to the common sub sequences found between them. The starting and ending points are averaged to find the appropriate translation in x and y co-ordinates. Translate the points of the second fragment and find the single reconstructed image.

3. PROPOSED APPROACH FOR NON-ALIGNED FRAGMENTS

The approach for the joining of two fragments, which are not aligned to each other and having some relative angle in between them, is following –

i. Find the two such lines/curves one of first fragment and other of second one based on similarity.
ii. Find the angle between lines/curves.
iii. Rotate the second fragment with respect to the mid-point of the it with the angle between these fragments
iv. Both fragments are aligned now, apply the approach of aligned fragments in order to join these two.

3.1. SIMILAR LINES / CURVES

We have to first find the local curvature of the fragment, which can be treated as one to one mapping between a regular curve and its curvature function. The curvature function is the derivative of the tangent angle to the curve, parameterized as a function of its arc length. The local curvature of the boundary in a pixel pis calculated as the magnitude of the derivative of the local tangent in p.

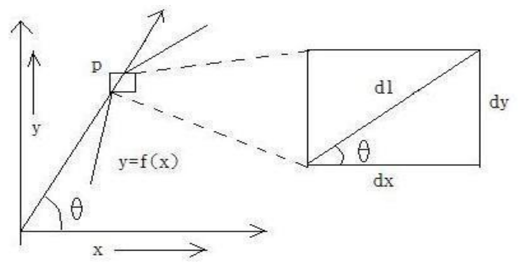

Fig6. Local curvature

Curvature can be used to detect features where the image surface bends sharply.

The similar line can be taken in a fragment in such a way that it is the first most number of pixels having the same slope Same method will be for other fragment also.

The slopes of the lines can be calculated by the formula-

$$m = \frac{y2 - y1}{x2 - x1}$$

3.2. ROTATION ANGLE

We get the relative angle between these two fragments by calculating the angle between these two similar lines-

Let the slopes of the lines are m1 and m2 , θ is the angle between them.

$$\tan\theta = \left|\frac{m_1 - m_2}{1 + m_1 m_2}\right|.$$

3.3.ROTATION

i. If m1= m2, both lines are parallel to each other, But we are taking here the non _aligned case, so we have to rotate the second fragment with angle 180 deg.
ii. If m1*m2 = -1, both line perpendicular to each other, require rotation of one line with respect to other at 90 deg.
iii. Otherwise rotate the second fragment with angle θ with respect to the mid point of of it.

The rotation matrix is –

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}$$

Where (x', y') are the co-ordinates of the point after rotation, and the formulae for x' and y' can be seen to be

$$x' = x\cos\theta - y\sin\theta$$
$$y' = x\sin\theta + y\cos\theta.$$

The vectors $\begin{bmatrix} x \\ y \end{bmatrix}$ and $\begin{bmatrix} x' \\ y' \end{bmatrix}$ have the same magnitude and are separated by an angle θ as expected

After rotation both fragments are aligned . Now,we use the method for aligned fragments to get the reconstructed image.

4. CONCLUSION

In this paper joining of 2D fragments of an image is present. The method presented uses information about boundaries. The outlines of the fragments are represented by chain code. This paper mainly focused to improve method for reassembling several fragments and to experimentally test the method in real-case applications.

5. FUTURE SCOPE

Another direction of future work is to capable the method in order to joining of n fragments. The reassembled images have been obtained by designing an algorithm and from the collection of fragments, take two fragments with the best match (the longest common portion of borders), add this new fragment obtained by joining the two fragments and remove fragments that recently have been matched from the collection, and continue until we get a single image.

REFERENCES

[1] Guide to the Recovery, Re composition, and Restoration of Shattered Wall Paintings: Experience Gained at the Basilica di San Francesco in Assisi, ICR, Rome, Italy, 2001.

[2] G. Budea and H. Wolfson, "Solving jigsaw puzzles by a robot," *IEEE Transactions on Robotics and Automation*, vol.5, no. 6, pp. 752–764, December 1989.

[3] A Method For Reassembling Fragments In Image Reconstruction**,** Francesco Amigoni, Stefano Gazzani, Simone Podico, Artificial Intelligence and Robotics Laboratory; Dipartimento di Elettronica e Informazione Politecnico di Milano; Piazza Leonardo da Vinci 32, 20133 Milano, Italy.

[4] A. Pope, "Model-based object recognition: A survey of recent research," Tech. Rep. TR-94-04, University of California at Berkeley, Berkeley, CA, USA, January 1994.

[5] MIT Computer Graphics Group, "Intro to computer graphics," 2003.

[6] H. Wolfson, "On curve matching," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 12, no. 5, pp.483–489, May 1990.

[7] H. C. de Gama Leitao and J. Stol, "A multiscale method for the joining of two-dimensional fragmented objects", IEEE Transactions on Pattern Analysis and Machine Intelligence,vol. 24, no. 9, pp. 1239-1251, September 2002.

[8] K. Hori, M. Imai, and T. Ogasawara, "Joint detection for potsherds of broken earthenware", in Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), June 1999, vol. 2, pp. 440-445.