Rotation independent hierarchical representation for Open and Closed Curves and its Applications

Siddharth Shivapuja[1], Vineetha Bettaiah[2], Thejaswi Raya[3] and Heggere Ranganath[4]

[1]Honeywell Scanning and Mobility
Blackwood, NJ 08012, USA
sshivapuja@gmail.com
[2]The University of Alabama in Huntsville
Huntsville, AL 35899, USA
vb0003@cs.uah.edu
[3]The University of Alabama in Huntsville
Huntsville, AL 35899, USA
thr0001@cs.uah.edu
[4]The University of Alabama in Huntsville
Huntsville, AL 35899, USA
ranganat@cs.uah.edu

***Abstract*— The algorithm used for the segmentation of an image, and scheme used for the representation of the segmentation result are mostly selected based on the final image analysis or interpretation objective. The boundary based image segmentation and representation system developed by Nabors segments and stores the result as a graph-tree hierarchical structure that is capable of supporting diverse applications. This paper shows that Nabors' hierarchical representation of curves is not invariant to rotation, and proposes an enhanced representation which retains its structure and remains invariant under rotation. The curve matching algorithm which matches two curves based on their hierarchical representation makes it easy to determine if a curve is a section of a larger curve. The potential of the representation is illustrated by developing image registration and image stitching methods based on the new representation.**

***Keywords*— Image Segmentation, Curve Matching, Rotation Independent Hierarchical Representation, Image Registration, Image stitching, Boundary Representation**

I. Introduction

The selection of an algorithm for the segmentation of an image, representation scheme used to store the segmentation result, and algorithms used for the extraction of features is almost always done with the prior knowledge of the final image interpretation objective. Often, it is desirable to segment, represent and store the segmented image for later use without any kind of prior knowledge on how the segmentation results will be used by multiple users at different times. As an example, consider Hough Transform based representation. If the goal is to find all straight lines present in an image then the edge image may be stored as a Hough array in the slope-intercept parameter space. If the goal is to find circles in the same image, then the edge image must be mapped to a 3-dimensional parameter space (coordinates of the center, and radius). In other words, final analysis objective determines the representation.

Sensors on NASA satellites generate many terra bytes of image data of Earth and space each year which is archived in NASA data centers. Space and Earth scientists, with diverse analysis and interpretation objectives, access and process these images. In this situation, it is highly desirable to segment and store the segmentation result in a compact manner using an appropriate representation that is suitable for many diverse applications. In 2000, Nabors has developed one such representation scheme [1]. He argued that an image, a two dimensional projection of a three dimensional scene, consists of lines, curves and regions. An image region is defined as a set of spatially connected pixels with similar spatial property. It may be represented by its boundary and spatial property. If the input image is transformed into a binary edge image then the resulting edge image consists of lines, open curves and closed curves. These features are captured and stored in a hierarchical representation which utilizes weighted graph as well as tree data structures defined on the same set of nodes [1]. An object is stored as a related set of open and closed curves and an image is stored as a spatially related set of objects. Each curve is represented by a node in the weighted graph, and this node serves as the root node for the tree data structure which describes the curve in terms of smaller curve segments and each curve segment is represented by a node in *level-1*. Further, each curve segment is partitioned into straight line segments, and each line is represented by a node in *level-2*. The weighted edges of the graph encode the broad spatial relationship that exists among various curves of the object.

Nabors defined four types of primitive curve segments based on the slope along the curve as shown in Table I. If a curve segment is traced from top to bottom, at each pixel, the option is to move to one of its two possible neighbours. Therefore, a curve segment can be encoded by a binary string (1-bit chain code) in which 0 represents horizontal or vertical neighbour depending on the segment type, and 1 represents the diagonal neighbour. Based on the properties of straight lines, it is easy to partition the binary string of a curve segment into sub-strings where each sub-string represents a straight line [1].

The curve extraction and representation system developed by Nabors receives the thinned edge image as input and outputs Nabors' hierarchical representation of the input image. The system consists of four curve segment extraction networks (N_1, N_2, N_3 and N_4), and a line detector as shown in Fig. 1. The curve extraction network N_i scans the image in the raster scan order finds all *Type-i* curve segments in the input image. It consists of an array of processing nodes, one processing node for each image pixel. Each processing node receives input from its eight neighbours, and is capable of doing the following:

DOI: 10.5176_2010-2283_1.2.38

TABLE I
TYPES OF CURVE SEGMENTS

Curve Segment	Slope Range
Type-1	Less than -1
Type-2	[-1, 0]
Type-3	[0, 1]
Type-4	Greater than 1

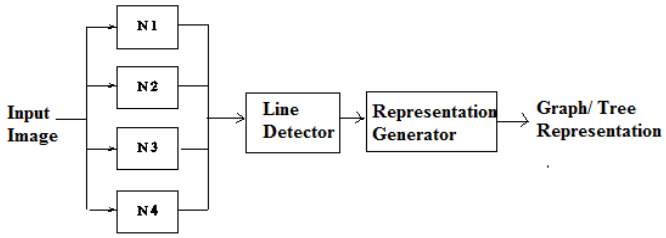


Fig. 1 Curve extraction and representation system

1) Initiate the tracing of a new curve segment of *Type*-i, if new curve initiation conditions are met. In this case, this pixel becomes the starting point of the curve segment.
2) Continue growing the curve segment that is being traced by appending a bit to the 1-bit chain code received from its neighbour. A 1 is appended if the data packet of the curve segment is received from its diagonal neighbour. Otherwise, 0 is appended to the binary chain code. The data packet is sent to the next node.
3) If curve segment termination condition is satisfied, the curve segment is terminated. The data packet consisting of the coordinates of the starting point and 1-bit chain code is sent to the line detector.
4) After terminating a curve segment, N_i sends signals to the other three networks to begin tracing the next curve segment of the curve starting from the termination point.
5) The line detector which consists of two state machines partitions each curve segment into straight lines.
6) The representation unit generates the hierarchical representation.

II. AN ALALYSIS OF NABORS' REPRESENTATION

Nabors' boundary based hierarchical representation scheme has many merits. The structure of the weighted graph is invariant to rotation, translation and scale. Each curve, irrespective of its type, is represented by a tree. The tree leaf nodes represent an open curve as a poly-line, and closed curve as a polygon. Therefore, one may view this representation as a unified representation scheme which handles all types of curves uniformly. It has been demonstrated that computationally efficient algorithms can be developed for the determination of several geometric attributes (concave up, concave down, number of inflection points, convex hull, minimum bounding rectangle, number of concavities, etc.) of a curve directly from the representation [2,3]. Its potential in several practical problems such as object recognition, scene matching, content based image retrieval, and the generation of the union of images with overlapping fields of view has been demonstrated [4].

However, careful analysis of the representation reveals that there are a few critical issues which must be resolved. The number of nodes in the first level of the tree data structure that represents a curve is not invariant to rotation. The line detector is not always able to partition a curve segment and its rotated version into the same number of line segments. Minor irregularities in boundary due to noise and artefacts of the thinning algorithm used for thinning edge images contribute to this problem. Therefore, the task of matching two instances of an object or a curve, where one instance is a rotated version of the other, is not straight forward. Also, the task of finding if a curve in one representation is a part of larger curve in another representation is not an easy problem when there is rotation (In other words, smaller curve is a part of its rotated version).

Nabors correctly identified that the structures of trees in the hierarchical representation are not invariant to rotation. Accordingly, he developed a two step approach for matching two representations based on the two theorems he stated and proved in his dissertation [1]. These theorems are given below.

Theorem 1: If a curve is detected by a single curve extraction network, then one or two networks are required for the detection of the rotated version of the curve.

Theorem 2: If a curve is detected by k curve extraction networks, then, k-1, k, or k+1 curve extraction networks are needed to detect a rotated version of the curve.

Step 1, ensures that the graph structures of the two representation match based on broad features such as *slope differential sequence*, number of curve segments, and edge weights associated with the nodes in the two representations. If successful, the difference in rotation and scale are estimated based on the pairs of matching nodes of the two representations determined in Step 1. Then curves of one representation are rotated and scaled before the final matching in Step 2. Finally, pairs of curves represented by matching nodes of the two graphs are matched by comparing the coordinates of the starting points of their line segments.

The above approach to handle rotation during curve matching has major disadvantages. *Theorem 2* states that the number of curve segments into which a curve gets partitioned changes by at most one when the curve is rotated. This is not always true. *Theorem 2* may not hold if the curve is not smooth or have inflection points (points at which curvature changes sign). For example, if the curve in Fig. 2(a) is rotated by 30 degrees in the counter clockwise direction, the number of curve segments changes from 7 to 9 as shown in Fig. 2(b).

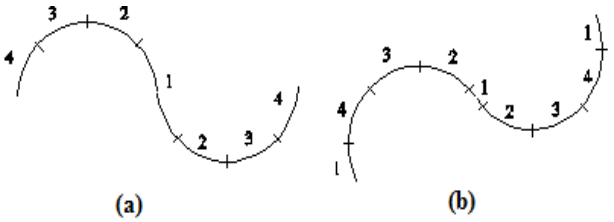


Fig. 2 Segments of a curve and its rotated version.

Also, if a curve consisting of segments of type 4, 3 and 1 (not a smooth curve as segment type 2 is missing) with no inflection points is rotated, the resulting curve may consist of segment types 4, 3, 2, 1, and 4. Once again the number of curve segments may change by two. Therefore, it is not accurate to use the number of curve segments in the manner to Nabors used to determine initial match between two curves. As a result, the rotation angle and the scale factor estimated in Step 1 may not be accurate. Finally, the ability to determine if a curve is a part of another larger curve is important to many applications. It is not possible to develop an efficient algorithm to accomplish the task based on Nabors' hierarchical representation.

III. ROTATION INDEPENDENT HIERARCHICAL REPRESENTATION

In this section, an enhanced hierarchical representation which is completely independent of rotation is presented. The new

representation retains the graph portion of the Nabors' representation without any modification. However, the trees which represent the curves of the object are obtained using a different approach. Instead of partitioning an open curve into segments based on the slope along the curve, it is partitioned into curve segments by breaking the curve at inflection points. An inflection is the point at which the second derivative changes sign. This partitioning method is not sensitive to rotation, as the relative position of the inflection point on the curve does not change with rotation. The number of curve segments (number of nodes in *Level-1*) into which an open curve gets partitioned is one more than the number of inflection points. The line detector, because of the rounding errors, may break a curve segment and its rotated version into different numbers of line segments. To keep the number of child nodes in *Level-2* and beyond consistent, a different method is proposed to partition the curve segments into line segments. Each curve segment is recursively partitioned at its mid point, until each partition can be approximated by a straight line with the desired degree of accuracy. This partitioning is not affected by rotation. The structure of the sub-tree which represents a curve segment is a binary tree. However, the number of levels in the tree is not fixed.

The above enhanced hierarchical representation can easily be obtained from Nabors' representation or the 3-bit chain code. It has been shown that general shape features of curves such as concave-up segment, concave-down segment, local maximum, local minimum, and infection points can be determined by examining the *network sequence* and *slope differential sequence* which are stored at the root node of each curve[4]. For the purpose of illustration, consider the curve in Fig. 2(a). Its *network sequence*, *network differential sequence* (first difference of *network sequence*) and *slope differential sequence* (obtained by adding contiguous numbers of the same sign in *network differential* sequence) are [4, 3, 2, 1, 2, 3, 4], [-1, -1, -1, +1, +1, +1] and [–3 +3], respectively. The number of inflection points on a smooth curve is equal to the number of sign changes in the *slope differential sequence*. The change in sign from –3 to +3 indicates the presence of one inflection point. Each negative number in the *slope differential sequence* represents a concave down (left) section, and each positive number represents a concave up (right) segment of the curve. Concave down and concave up segments of a smooth curve meet at an infection point.

In order to determine the locations of the inflection points, the network sub-sequences corresponding to each value in the *slope differential sequence* are determined. Adjacent network sub-sequences will have a common curve segment. Each curve segment that is common to a pair of adjacent network sub-sequences has an inflection point on it. For example, the network sub-sequences corresponding to -3 and +3 in the *slope differential sequence* of the curve in Fig. 2(a) are [4 3 2 1] and [1 2 3 4], respectively. Therefore, the curve has one inflection point which is located on the *Type1* curve segment that is common to the two network sub-sequences. It is easy to see that there is one inflection point on each curve segment which connects to the same type of curve segments at both ends. Note that the*Type1* curve segment in Fig. 2 (a) connects to *Type2* curve segments at both ends. The actual location of the inflection point may be obtained by examining the 1-bit chain code of the curve segment. The number of zeros (ones) between successive ones (zeros) will be non- decreasing on one side of the inflection point and non-increasing on the other side as the curve is traversed from one end to the other end, or vice versa.

In the enhanced representation, the curve in Fig. 2(a) and its rotated version in Fig. 2(b) are both partitioned consistently into two segments at the inflection point, and each curve segment is represented by a *Level-1* node of the tree data structure. At each *Level-1* node, the end points and the 3-bit chain code of the curve segment, which can be obtained from Nabors' hierarchical representation, are stored. The next step is to recursively split each curve segment at the mid point, until all sections of the curve represented by the leaf nodes can be approximated as straight lines with the desired degree of accuracy. One of the following methods can be used to check if a curve may be approximated by a straight line or not.

1) Calculate the ratio of the length of the curve and the length of the line joining the end points of the curve. For a straight line, this ratio is equal to 1. All curves for which this ratio is less than a threshold may be considered as a straight line. The value of the threshold depends on the accuracy required.
2) If all points on the curve are within a specified distance from the line joining the curve's end points, then the curve is taken as a straight line. In order to keep computation reasonable, this measure is computed for a few points fairly uniformly spaced along the curve.

Each curve segment in Fig. 2 partitions into 8 straight line segments if it is recursively partitioned until all pixels on each of the resulting partition are within a distance less than 12% of the length of the line joining the endpoints of the partition.

The Closed curves which may or may not have concavities pose a different challenge. Since a closed curve does not have definite start and end points, Nabors selects the top most point as the starting as well as ending point. Obviously, the starting point selected in this manner is not invariant to rotation. If the starting point changes, the partitioning also changes. Therefore, the new approach partitions a closed convex curve into segments by breaking the curve at points where the minimum bounding rectangle touches the boundary of the curve. If the closed curve has concavities, then it is partitioned at the inflection points, similar to the way an open curve is partitioned. If desired, a closed curve with concavities may be partitioned at infection points as well as the points at which the minimum bounding rectangle touches the curve. All these partitions are invariant to rotation.

IV. CURVE MATCHING ALGORITHM

This section presents a new and elegant algorithm to match and determine the similarity between two curve segments, *Curve-Seg1* and *Curve-Seg2*, by matching their binary-tree representations. First, a few properties or observations about the binary tree are given below:

1) The section of the curve represented by a leaf node is linear or almost linear.
2) The root node represents the entire curve. Each Level-1 node represents ½ of the curve. In general, each Level-h node represents one of the 2^h sections of the curve.
3) The j^{th} node from left in level h, represents the j^{th} section of the curve from the left.
4) The flatness of the section of the curve represented by node k can be inferred by the depth of the sub tree for which node k is the root.

Therefore, the binary tree representation of a curve allows one to describe the shape of the curve in a non-quantitative manner like human beings do. For example, consider the binary tree representation of a curve in which the leaf nodes, in the order of inorder traversal, are [4 5 12 13 48 98 99 50 51 30 31]. An ordered list of the levels in which these leaf nodes appear in the tree is [2 2 3 3 5 6 6 5 5 4 4]. From the list of levels, the following statements can be made.

1) The left quarter of the curve is linear.
2) The second left quarter of the curve is linear.
3) The left half of the curve is likely flat.

4) The curve starts bending roughly at the center and then turns sharply.
5) The curve is relatively flat at the right end (1/8 of the curve).

Now, a method for the determination of the dissimilarity between two curve segments based on the location of the leaf nodes in their respective binary tree representation is presented. Let A = [2 2 3 3 5 6 6 5 5 4 4] and B = [2 3 4 5 6 6 5 5 4 3 2] be the lists of the levels in which leaf nodes of *Curve-A* and *Curve-B* appear in their binary tree representations. Since A[0] = B[0] = 2, it is clear that the first quarter of both curves are linear and have roughly the same shape. Therefore, the error for this section is taken as zero. The next quarter of the first curve is linear (A[1] = [2]). However, the corresponding quarter of the second curve is not linear. It is represented by a binary sub-tree of depth 4 with 5 leaf nodes (B[1..5] = [3 4 5 6 6]). The error or dissimilarity between these curve sections is taken as 4, the height of the sub-tree in B that corresponds to the section of the curve represented by the leaf node corresponding to A[1] = 2. Next A[2] = 3 the next one eighth of the first curve, and B[6..8] = [5 5 4] represents the corresponding eighth of the second curve. The error between these two sections is 2. The shape of the next eighth in the first and second curves match (A[3] = A[9] =3), and contribute no error. Finally, the last fourth of the second curve (B[10] = 2) do not match in shape to the last fourth of the first curve (A[4..10] = [5 6 6 5 5 4 4]), and contribute 4 units to the error. The overall measure of dissimilarity between the two curves is obtained by adding individual errors, and in this case, is 10. The algorithm which implements the above method is given below. The asymptotic order of computation is O(m+n), where n and m are the number of leaf nodes in the two binary trees being matched.

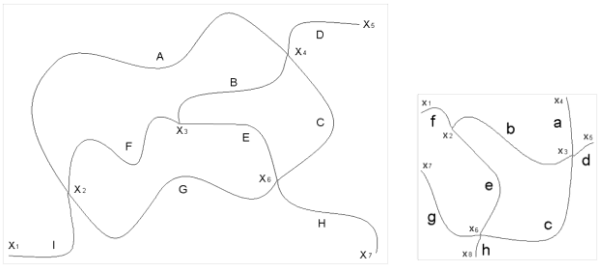
```
BinMatch(A[0..n-1], B[0..m-1])
{// A contains levels of leaf nodes of Curve-Seg1 in
   the order of inorder traversal
 // B contains levels of leaf nodes of Curve-Seg2 in
   the order of inorder traversal
        error ← 0; i ← 0; j ← 0;
        while((i < n) and (j < m))
        {
            if (A[i] == B[j])
            {
                i← i + 1; j←j + 1;
            }
            else if (A[i] < B[j])
            {
                p ← 1 / 2^A[i]; q ← 0;
                d ← B[j];
                while ( q != p)
                {
                    q ← q + 1 / 2^B[j];
                    if (d < B[j] )
                        d ← B[j];
                    j←j + 1;
                }
        error ← error + (d – A[i])
        i← i + 1;
    }
    else
    {
        p ← 1 / 2^B[j]; q ←0;
        d ← A[i];
        while ( q != p)
        {
            q ← q + 1 / 2^A[i];
            if (d < A[i] )
                d ← A[i];
            i← i + 1;
        }
        error ← error + (d – B[j])
        j← j + 1;
    }
  }
  return error;
}
```

V. APPLICATIONS

The enhanced hierarchical representation and the curve matching algorithm presented in this paper support many practical applications including object recognition, scene matching, contents based image retrieval, image registration and image stitching. In this section, image registration and image stitching methods based on the enhanced hierarchical representation are described. Assuming that the field of view of a given reference image is completely contained within the field of view of a larger image (known as search area), the task of the registering the reference image within the search area is the process of finding the subimage of the search area whose field of view is same as that of the reference image. In general, the two images differ in rotation and scale. Image stitching is defined as the process of determining the image that represents the union of the fields of view of several images. It is assumed that the fields of view of images to be stitched have sufficient overlap to allow the construction of the image corresponding to their union.

A. Image registration

The search area shown in Fig. 3(a) is a synthetic edge image of size 600x800. It consists of nine curves, and therefore, its graph representation has nine nodes. The reference image (300x400) to be registered, Ref-1, is shown in Fig. 3(b). Ref-1 is required to include at least two points at which three of more curves meet. Ref-1 contains eight curves labeled *a* through *h* and its field of view is completely contained within the field of view of the search area. However, the reference image differs from the search image in rotation by 45 degrees. A four step registration method based on the enhanced hierarchical representation is given below.



(a) (b)

Fig. 3 Illustration of Image registration. (a) Search Area. (b) Reference image.

TABLE II
CONNECTIVITY OF CURVES IN THE SEARCH AREA

End Point	Connected Curves	Number of Curves
X_1	I	1
X_2	A, F, G, I	4
X_3	B, E, F	3
X_4	A, B, C, D	4
X_5	D	1
X_6	C, E, G, H	4
X_7	H	1

Step 1: The co-ordinates of curve end points in the search area at which three or more curves meet are determined by examining the end points of all curves in the hierarchical representation, one curve at a time, and then by grouping the curves which share a common and point. The seven curve end points (X_1 through X_7) and the curves meeting at each end point are shown in Table II.

Step 2: The co-ordinates of curve end points in Ref-1 at which three or more curves meet are also determined and the results are summarized in Table III.

TABLE III
CONNECTIVITY OF CURVES IN THE REFERENCE

End Point	Connected Curves	Number of Curves
x_1	f	1
x_2	b, e, f	3
x_3	a, b, c, d	4
x_4	a	1
x_5	d	1
x_6	c, e, g, h	4
x_7	g	1

Step 3: Based on the information obtained in Step 1 and Step 2, all mappings of the end points of Ref-1 to those of the search area are determined. The six possible mappings are shown in Table IV.

TABLE IV
MAPPINGS OF ENDPOINTS OF REF-1 AND SEARCH AREA

Mapping	Pairs of intersection points
Mapping1	(x_3,X_2), (x_6,X_4), (x_2,X_3)
Mapping2	(x_3,X_4), (x_6,X_2), (x_2,X_3)
Mapping3	(x_3,X_6), (x_6,X_2), (x_2,X_3)
Mapping4	(x_3,X_2), (x_6,X_6), (x_2,X_3)
Mapping5	(x_3,X_4), (x_6,X_6), (x_2,X_3)
Mapping6	(x_3,X_6), (x_6,X_4), (x_2,X_3)

Step 4: The validity of mapping (x_i, X_j) in Table IV is determined by matching each curve meeting at X_j in search area with each curve meeting at x_i in Ref-1. For example, matching results for curves associated with the mapping (x_3, X_2) is given in Table V. The numbers in the "angle" column are the angle between the chords of the two curve segments being matched. The number of curve segments of each curve is also shown. For example, G consists of 3 curve segments which is shown as G(3). Note that curves b and c of Ref-1 do not match with any of the curves in the search area meeting at X_2. Therefore it is concluded that X_2 does not map to x_3 and options Mapping 1 and Mapping 4 are eliminated from further consideration. Similarly mappings 2, 3 and 6 also get eliminated. Results in Table VI indicate that x_3 of Ref-1 matches X_4 of search area. This is because the curve pairs (a, A), (b, B), (c, C) and (d, D) cluster tightly in the two-dimensional error-angle space. Similarly, x_6 matches with X_6, and $x_2$matches with X_3. Therefore, Mapping 5 becomes the only valid mapping, and it identifies the subimage of the search area that matches Ref-1.

B. Image synthesis

The image synthesis method based on the enhanced hierarchical representation is illustrated by the union of 2 edge images, IMG1 and IMG2, shown in Fig. 4 and Fig. 5, respectively. The two images have the same spatial resolution and differ in rotation by 45 degrees. There is considerable overlap in the two fields of view. The procedure for the construction of the union is given in the step below.

TABLE V
CURVE MATCHING RESULTS FOR MAPPING (x3, X2)

	A(3)		I(1)		G(3)		F(3)	
	Error	Angle	Error	Angle	Error	Angle	Error	Angle
a(1)	0	24	2	177	1	-143	1	-26
b(1)	8	-	8	-	8	-	8	-
c(1)	8	-	8	-	8	-	8	-
d(1)	0	91	0	-133	0	-85	0	52

TABLE V1
CURVE MATCHING RESULTS FOR MAPPING (x3, X4)

	A(3)		B(2)		C(1)		D(1)	
	Error	Angle	Error	Angle	Error	Angle	Error	Angle
a(1)	0	45	2	126	1	-147	2	-54
b(2)	8	-	2	45	8	-	8	-
c(1)	8	-	8	-	0	45	8	-
d(1)	0	104	1	-146	0	-82	0	47

Step 1: From the hierarchical representation of IMG1 and IMG2, the co-ordinates of X_2, X_4, and X_5, the points at which three or more curves meet, are determined.

Step 2: From the representation of IMG2, co-ordinates of x_2, x_4, and x_6, the point at which three or more curves meet, are determined.

Step 3: All pairs of points (x_i and X_j) for which the group of curves at x_i match the group of curves meeting at X_j are determined using the curve matching algorithm. For the current example, it is found that x_4 matches X_4, x_6 matches X_5 and IMG2 must be rotated by 45 degrees before merging with IMG1.

Step 4: The hierarchical representation of IMG2 is modified to account for the rotation of IMG2 by 45 degrees. A detailed algorithm for modifying the representation is given in [3].

Step 5: End points of all lines are changed to account for the translation required to make the matching curve meeting points of IMG1 and IMG2 coincide.

Step 6: In this step, the two representations are merged to obtain the representation of the union based on the following rules.

Rule 1 – Pairs of curves, one from IMG1 and another from IMG2, that match must be represented by the longer curve in the union.

Rule 2 – If a curve is outside the region in which IMG1 and IMG2 overlap then its corresponding node along with its edges is included in the graph of the union.

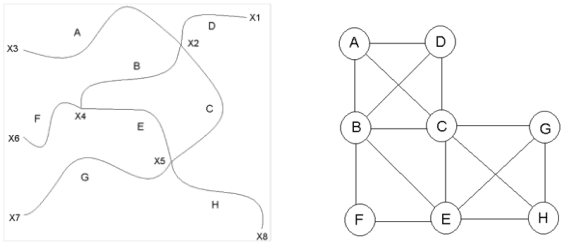
Rule 3 – If the free end point of a curve of one image is in the region of overlap, then it must be examined for merging with its other part (if present) in the second image.

As x_4 matches with X_4, and x_6 matches X_5, the curve connecting x_4 and X_6 can be taken from either of the representation. Curves B and f match and is longer than F. Therefore, the node that represents B in the hierarchical representation of IMG1 is used in the representation of the union (Rule 1). Similarly, the application of Rule 1 includes the nodes corresponding to curves C, H, c and D into the unions representation. Application of Rule 2 includes nodes corresponding to D and b. As the free end point of A and a are in the overlapping region and their overlapping sections are close, they are merged and represented by a single node. The union of IMG1 and IMG2 is same as the one in Fig. 6, and its graph is shown in Fig. 7.

VI CONCLUSION

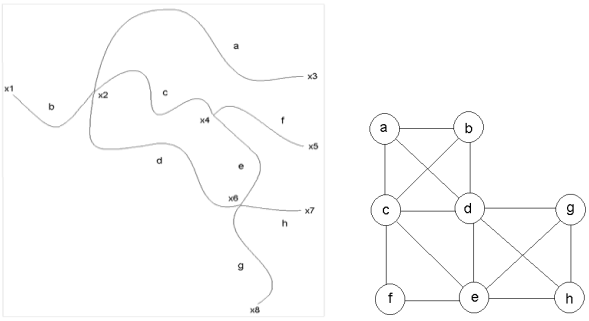
A rotation independent representation of edge or boundary image is described in this paper. The curve matching algorithm described elegantly matches two curves based on rough shape of corresponding sections. The representation with the help of curve matching algorithm has made the task of determining if a curve is a section of a larger curve an easy task even under rotation. The new rotation

independent representation may be obtained directly from Nabors hierarchical representation or from 3-bit chain code. The suitability



(a) (b)

Fig. 4 (a) IMG1 (b) Graph of IMG1



(a) (b)

Fig. 5 (a) IMG2 (b) Graph of IMG2

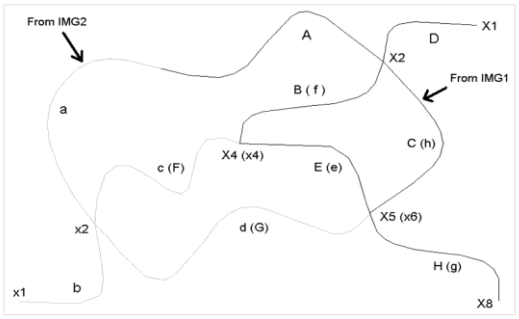


Fig. 6 Union of IMG1 and IMG2

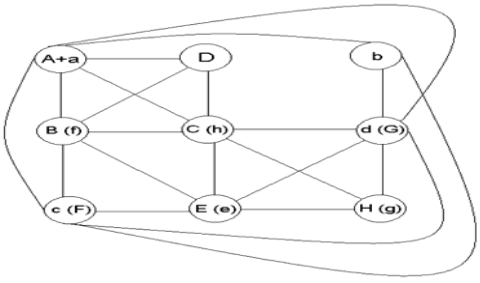


Fig. 7 Graph of the union of IMG1 and IMG2

of the representation for many practical applications in image interpretation area is illustrated by developing computationally efficient methods stitching based on the enhanced hierarchical representation for image registration and image.

REFERENCES

[1] D. H. Nabors, "*A boundary based image segmentation and representation method for binary edge images*," Ph.D. Dissertation, The university of Alabama in Huntsville, 2000.

[2] S. K. Kim and H. S. Ranganath, "Solving discontinuity and distortion problems in boundary based image segmentation and representation," *Proceedings of ACM Southeast Conference*, pp.431-436, 2007.

[3] S. K. Kim and H. S. Ranganath, "Efficient algorithms to extract geometric features of edge images," *The 2010 International Conference on Image Processing, Computer Vision, & Pattern Recognition*, Las Vegas, USA, July 12-15, 2010.

[4] S. K. Kim," *Hierarchical Representation of Edge Images for Geometric Feature Based Image Interpretation*, Ph.D. Dissertation, The University of Alabama in Huntsville, 2007.

Siddharth Shivapuja received his Bachelor of Engineering degree in Telecommunication from Vishveshwaraiah Technological University, India, in 2003. He then worked for Infosys Technologies for a year as a Software Engineer. Later he moved to the United States and got his Master's degree in Computer Science from the University of Alabama in Huntsville, Huntsville, Alabama, in 2007. He is currently working at Honeywell Scanning and Mobility as a Software Engineer where he is using Image Processing techniques to decode barcodes.

Vineetha Bettaiah received her Bachelor of Engineering degree in Computer Science from M.S Ramaiah Institute of Technology, Bangalore, India, in 2008. She is currently pursuing her Master of Science degree in Computer Science at the University of Alabama in Huntsville, Huntsville, Alabama. She is employed by the computer science department as a Graduate Teaching and Research Assistant. Her areas of research interest include Multimedia Systems, Artificial Neural Networks, Image Processing and Spatio-temporal Databases.

Thejaswi H Raya received his Bachelor of Engineering degree in Computer Science from B.M.S College of Engineering, Bangalore. Immediately after completion he joined SpikeSource, Inc based in Redwood City, CA, where he worked as a Software Engineer developing Opensource applications for enterprises. Currently he is pursuing his Master of Science degree in Computer Science at the University of Alabama in Huntsville, Huntsville, Alabama. As a teaching assistant, he teaches entry-level programming classes. Areas of his research interests include Multimedia Systems, Pattern Recognition and Artificial Neural Networks.

Heggere Ranganath received his Ph.D. degree in Electrical Engineering from Auburn University in 1980. Since 2002 he is serving as the chair of the Computer Science Department at the University of Alabama in Huntsville, Huntsville, Alabama. During his 30-year career as a professor, Dr. Ranganath has served as a technical advisor to many private technology companies and Government agencies, received over $4,000,000 in research funding, and published over 100 technical papers. His areas of research include Image Processing, Pattern Recognition, and Artificial Neural Networks.