Matching 2D Fragments of Objects

C. Stringfellow, R. Simpson, H. Bui, and Y. Peng
Department of Computer Science
Midwestern State University
Wichita Falls, TX 76308
{catherine.stringfellow} {richard.simpson}
@mwsu.edu

J. Hood
Department of Mathematics
Midwestern State University
Wichita Falls, TX 76308
jeffrey.hood@mwsu.edu

Abstract

This paper describes an algorithm that pieces together 2D fragments of an object. Matching fragments of an object is useful for solving puzzles or reassembling archaeological fragments. Many factors, such as the number of pieces and the complex shapes of pieces make this a difficult problem. Various approaches to this problem exist. This paper presents an approach on corner detection and curve matching. The work described in this paper starts with 2D objects, but should be easy to extend to 3D problems.

1 INTRODUCTION

The assembly of fragments of an object using computer software has significant applications in the real world. One of many usages is to reconstruct broken pottery and other fragmented, usually priceless, artifacts while excavating ruins of ancient civilizations.

Due to fragmentation, archaeologists encounter a challenging and formidable task reconstructing an entire object. Reconstruction may be handled in two major ways, manually and by using software. A manual approach is time consuming, especially for a large number of fragments and often requires a lot of direct contact with objects. This increases the possibility of damaging those priceless artifacts. In contrast, a software-based approach could efficiently reconstruct a large number of pieces, and the results could then be kept in an electronic document for study.

The assembly of the fragments of 3D objects is complicated. This work focuses the research on 2D objects, rather than 3D. In this way, all the drawbacks in 2D approaches are resolved before leaping to the 3D world to simplify the computational complexity. Further, the 2D problem becomes strikingly similar to the issue of solving popular jigsaw puzzles.

In section 2, previous work on matching fragmented objects is discussed. Section 3 introduces a method to perform 2D curve matching. Section 4 describes the results of processing puzzle pieces with the method introduced in Section 3. Section 5 presents conclusions and future issue to explore.

2 HISTORY REVIEW

Numerous previous works attack curve matching issues. Issues include, but are not limited to the following:

- Curve definition: Should one define the curve by a discrete line of points or Bezier or B-Spline methods? How are critical points defined?

- Critical point detection: How are critical points detected precisely and efficiently?

- Matching approach: Is one fragment matched to others on all sides or are fragments joined and joined fragments then matched? Is there any limitation on the number of fragments that can be matched? Should we use genetic algorithms, curvature calculations, or dynamic programming?

- Efficiency: How efficient is the algorithm, especially compared to other algorithms?

- Reliability: Does the approach precisely produce the desired results? What is the error tolerance?

Approaches may be categorized by the types of objects, such as archeological artifacts, font characters, jigsaw puzzles, pieced maps and documents, etc. Some algorithms use a specific method that place restrictions on fragments, such as filling fragments into a rectangular grid or not including any T-joints. The approaches used by researchers vary based on the type of issues on which they focus. A brief summary of prior work is presented next.

Kampel and Melero [7] describe a document system for 3D archeological fragments based the Hough – inspired approach as described in [17] and the genetic algorithm (GA) based approach. The Hough-inspired approach can lead to incorrect profile lines extracted from the original artifact, due to the decoration on fragments. The GA-based approach is more flexible for irregular

shape objects, but since it requires user interaction, it is only efficient for a small number of fragments.

Leitao and Jorge Stolfi [10] use an incremental dynamic programming sequence-matching algorithm to match fragments outlines. This approach uses a large number of sampling steps to divide the fragment's outline into a series of curves. This approach was tested by re-assembling fragments of ceramic tiles. The results show that most sample pieces matched well. This approach can be implemented to solve for flat objects and other objects with a smooth, but curved surface. One drawback of their application is that the solving speed is slow.

Smet and Corluy [13] suggest an inter-fragment gap and overlap method to optimize the reconstruction of the fragment matching pairs. This approach relocates pieces that have gaps and overlaps due to dislocation of a matching pair. This approach also uses a multi-resolution approach to achieve accurate curve matching. This method also has performance problems.

Horst and Beichl [5] develop an algorithm for curve matching over open and closed space curves and planar curves. In their approach, a function associated with chord and arc length is used to form a set of matching points. A posterior merging step is used to decrease the quantization error. A least square technique is introduced for further improvement of the matching. The advantage of this approach is its O(n) algorithm. The drawback is that it is sensitive to noise level. The posterior merging step also increases execution time.

Some curves can be defined by Bezier and B-spline curve. Itoh and Ohno [6] use Bezier curves to fit Japanese font characters. The contour lines of original objects are divided into segments. These segments are fitted with Bezier curves using a least-square approach. In their work, smaller curves are generated and fitted to the characters, resulting in a curve shape that is closer to the original geometry. This algorithm, however, requires more Bezier curves, if the length of segments is long affecting efficiency.

Lejun Shao and Hao Zhou [12] use cubic Bezier curves to fit character outlines. A length and arc approach is utilized to determine corner points and joint points of each curve. The weighted recursive least square method is applied to improve the tightness of the fitting. Based on the author, the algorithm is reliable. However, some parameters used to determine the appropriate chord length are difficult to measure.

Toe and To [15] match outlines of font characters defined by B-splines. The control points of the B-splines curves are determined and the curves are matched by using dissimilarity between their control points. The approach fails to recognize complicated curvatures that have fast geometrical change along the path. It also fails to recognize curves, if the orientations of the test object curve and sample curve are not in the same path.

Lee et. al., [9] use 2D splines to match curves with deformable shapes. The approach uses strain energy to evaluate dissimilarity for each corresponding pairs of curve segments. This algorithm is applied to model-base shape detection and database retrieval for given data sets.

Kong and Kimia [8] develop a two-step scheme to solve 2D and 3D puzzles. The first step is to find likely candidate pairs of pieces using a partial matching technique. The candidate pairs are placed in a rank-ordered list. Mismatch may exist within the resulting pairs after this step. The next step to replaces ambiguous match pairs due to mismatch with other candidates from the rank-ordered list to find the finer pair. The puzzle is reconstructed from the refined pairs. This approach is applied to a data set retrieved from a 2D puzzle map. The results show that the matching works partially well. However, their reconstructed map does not completely match the original map. This mismatch of the 2D map is probably due to issues in the second matching step.

Goldberg and Malon et al. [2] use a method similar to Wolfson et al. [16] to solve jigsaw puzzles. Border pieces are assembled first and interior pieces afterward. This algorithm is efficient, but not fully automated. Some user interaction is required to identify the straight side of fragments. Also, their algorithm only works on pieces with four corners.

The approach presented by this paper does not require fitting pieces to a grid and requires a very small amount of user interaction to verify automated results and to give the user some control to refine the results during reconstruction.

3 METHOD

The suggested approach includes image processing, corner detection, fitness calculation and final curve matching. Each topic is discussed in details.

3.1 Image processing

To perform any curve matching by a software application, the first step is to extract the digital data of images. This paper uses boundary points to represent outlines of scanned objects. Extraction of the boundary points is done as follows:

1. Scan puzzle pieces in RGB format, convert to grayscale format, and then convert to binary format where each pixel is represented either as white or black.
2. Remove any noise using erosion and dilation techniques. Erosion and dilation are performed the same number of times to maintain the original geometrical property of input objects.

3. In order to create the final outline of the object, the set difference of the piece and its erosion is computed.

3.2 Corner point detection

Corner point detection is an important procedure in curve matching due to the fact that segments of a piece are determined by its corners. These curved segments are what are matched. Corner detection algorithms can be largely categorized into two types – non-parameter based and parameter based. In a parameter based algorithm, user input is needed to determine certain criteria or threshold values for detecting critical points. A non-parameter based algorithm does not require user input. Previous corner detection algorithms, both parameter and non-parameter based, fail to recognize all corners [1,3] or they detect both convex and concave corners simultaneously [11,14]. These algorithms do not suit current work since only convex corners are needed to match segments.

This paper uses a robust corner detection algorithm described in detail in Hood et al. [4]. (See Figure 1.) Hood's algorithm is suitable for detecting corner points on segments with either a regular shape or an irregular shape. The algorithm has the capability to detect convex corners of jigsaw puzzle pieces without presenting the unnecessary concave corners.

1. *Compute the center of mass for the discrete closed curve.*
2. *Calculate the square distance of each point along the curve to the center of mass.*
3. *Compute the average of several slopes away from each point on both sides.*
4. *Compute the curvature defined by the difference between the left slope and the right slope.*
5. *Smooth data to eliminate any minor noisy if exists.*
6. *Measure the mean of curvatures.*
7. *Select candidate corners based on curvature comparison between points.*
8. *Determine corners by looking at the highest points within the candidate regions.*

Figure 1. Corner detection Algorithm [4].

3.3 Candidates selection for curve matching

After convex corners are determined, all curve segments are processed, and sorted according to straight line distance between the start and end points. Processing involves determining important (location independent) characteristic values that describe a curve segment. Figure 2 shows a curve segment divided into four small sub-segments.

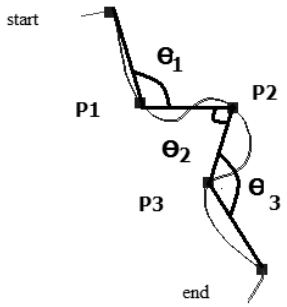

Figure 2. Break down a curve segments in order to calculate its fitness value.

Figure 2 also shows the start points and end points for the four sub-segments. The distance of a segment is computed in two ways: the pixel path length (PD) between *start* and *end* points, and the actual straight line distance between *start* and *end* points, denoted as D. Intermediate points of a segment (midpoint and two quarter points) are chosen to divide the segment into sub-segments. The distance lengths of the sub-segments are denoted d_1, d_2, d_3, d_4, respectively. The four sub-segments form three angles Θ_1, Θ_2, and Θ_3 as shown in Figure 2. All curve segments are processed only once, hence this part of the algorithm is linear in terms of the number of segments. The curve segments are then sorted according to their actual distance lengths, D. Candidate segments close to the length of the segment of the segment of the current piece to be matched are selected. A few fast tests are performed on the candidate segments to see if any may be rejected, based on the characteristic values computed. These tests use point samples, rather than a least squares method.

For example, if the relative difference, R, between the two different distances is greater than some threshold values, the candidate segment is rejected. Another test is applied to determine if the mid-point of segment S_i and mid-point of segment S_j are close to each other after a rotation and a translation are performed. Figure 3 shows the current segment and the candidate segment, while Figure 4 show the segments after the candidate segment is rotated and translated. If they are too far away from each other, then the segment is rejected as a candidate for matching. Similar tests are performed on the other characteristics values.

In order to match curve segments of pieces, a fitness function is introduced to determine how well the remaining candidate curve segments fit the current segment of the current piece. The fitness value, $F(S_i,S_j)$, is calculated between two segments S_i and S_j, where segment $\mathbf{S_i}$ and its

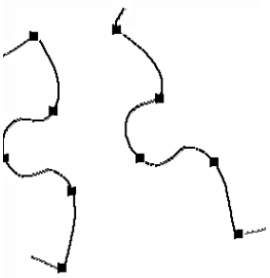

Figure 3. S_i and S_j and their sub-segments.

Figure 4. S_i and S_j and after a rotation and translation.

attributes $(PD_i,D_i,d_{i1},d_{i2},d_{i3},d_{i4},\Theta_{i1},\Theta_{i2},\Theta_{i3})$ and segment $\mathbf{S_j}$ and its attributes $(PD_j,D_j,d_{j1},d_{j2},d_{j3},d_{j4},\Theta_{j1},\Theta_{j2},\Theta_{j3})$ are from two different pieces. $F(S_i,S_j)$ is calculated using relative difference R where is the relative difference between a and b computed as:

$$R(a,b) = |a-b|/\max(a,b). \quad \text{(eq.1)}$$

$$F(S_i,S_j) = [R(D_i, D_j) + R(PD_i, PD_j) + R(d_{i1},d_{j4}) + R(d_{i2},d_{j3}) + R(d_{i3},d_{j2}) + R(d_{i4},d_{j1}) + R(\Theta_{i1}, \Theta_{j3}) + R(\Theta_{i2}, \Theta_{j2}) + R(\Theta_{i3}, \Theta_{j1})]/9. \quad \text{(eq.2)}$$

The smaller the fitness value, the better chance the two segments are actually a match. If the fitness is greater than or equal to 0.15, then the candidate segment is rejected from further consideration. The best candidate segment with the best fitness value is then selected from those remaining, if any.

3.4 Curve matching

The algorithm uses a depth first search and a local approach to match curves. The algorithm starts with one puzzle piece. Next, other pieces with the matches for the piece's curve segments are found and the adjacent pieces are pushed onto a stack. The next piece out of the stack is popped and the best match for its curve segments is found. The process is repeated until the stack is empty. Figure 5 shows this algorithm. This algorithm may result in several merged pieces.

```
Make all pieces unvisited
While any unvisited pieces
  Select unvisited piece and make
  visited
  Push unvisited piece on the stack
  While the stack is not empty
     current_piece=stack.pop()
     for every segment of current_piece
      find the best fit segment
      if there is a match and the
      adjacent piece with the best fit
      is not in the stack already
          rotate and translate adjacent
         piece to new position
          push adjacent piece onto the
          stack
```

Figure 5. Curve matching algorithm

4 RESULTS

The application was developed using C# within the Microsoft Visual Studio.Net 2005 platform. Figure 6shows puzzle pieces scanned in using a HP ScanJet 5200C scanner, several pieces at a time. Grayscale transformation is applied to convert the RGB image to the grayscale image, using the following formula:

New gray color = 0.2125 x R + 0.7154 x G + 0.0721 x B

where R, G, B are the red, green, blue values, respectively, of a pixel and 0 <= R,G,B, new gray color <=255. Then the image is converted to binary. If the new color of a pixel is dark enough, the application sets the color of the pixel to 0 (black), otherwise it sets the color to 255 (white).

Figure 6. RGB image scanned from the backside of 24 puzzle pieces with 4 pieces painted black.

After converting the image into binary format, there are still isolated black pixels and unwanted white pixels (inside the piece). Erosion and dilation are used to clean up the noise. The boundary of each puzzle piece is generated by comparing the original puzzle piece with the result of erosion applied on that piece. The boundary is the set of pixels that are black in the original piece, but are white in the result piece (Figure 7.)

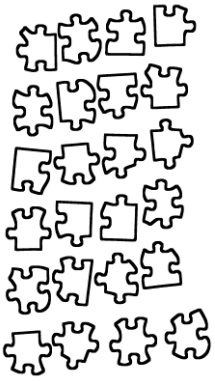

Figure 7. Puzzle pieces after boundary extraction.

Figure 8 shows the result of finding corner points using Hood's corner detection algorithm [4]. Notice some corners are missing.

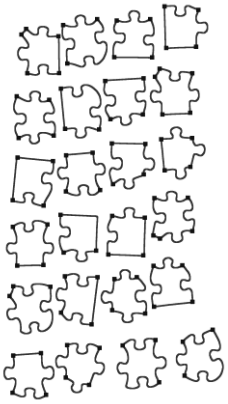

Figure 8. Corner points detection.

Once the corners are detected, the curve matching process can start by applying the curve matching algorithm as shown in Figure 5. Figures 9, 10, and 11 show the matching results by using this algorithm.

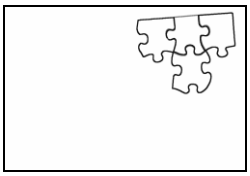

Figure 9. First step of curve matching.

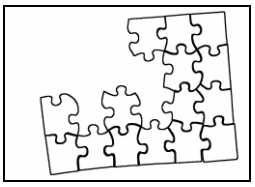

Figure 10. Intermediate step of curve matching

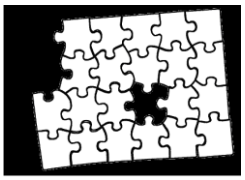

Figure 11. Final result of curve matching shown on a black background.

Applied to a 24 piece puzzle, the implemented application is able to put 21 pieces together. This application does not match all pieces: there are three missing pieces due to the existence of T-joints in them. Two of the three pieces did match up to each other, though.The algorithm was also run on a ten-piece puzzle without T-joints. One of the pieces only had three sides. In addition, this puzzle does not follow a grid. The algorithm successfully matched up all pieces. Figures 12 and 13 show the puzzle pieces after image processing and corner detection, and after matching.

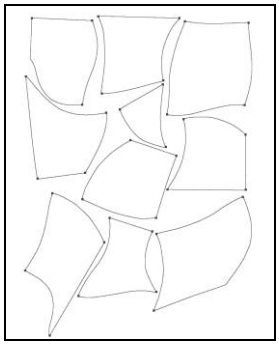

Figure 12. Ten piece puzzle after image processing and corner detection.

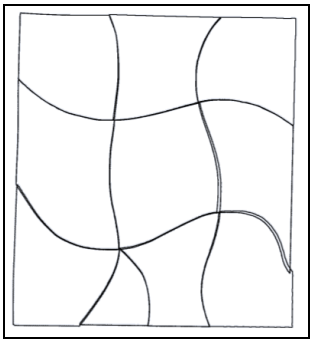

Figure 14. Ten piece puzzle after matching.

The only interaction required was during image processing to clean up noise. Boundary extraction and corner detection are the most time consuming part of the process. The matching algorithm actually works quite fast, due to rejecting candidates quickly leaving only a few segments left to compare.

5 CONCLUSION AND FUTURE WORK

This paper proposes a method and its implementation for automatic reconstruction of 2D jigsaw puzzles. Even though the algorithm does not handle T-joints, the suggested approach is considered a successful one, as it can match a piece to a different adjacent piece. This approach reconstructed most pieces in the given set of objects and it does reduce the interaction significantly.

One issue involves fitness parameters, which currently have to be adjusted based on puzzle piece characteristics. Some preliminary study suggests that basing the parameters on puzzle piece size may eliminate the problem.

Future research will include improving the accuracy of the corner detection algorithm, back-tracking to check that segments of neighboring pieces match, and limited user interaction to give the user some control. Under consideration is a method to combine connected pieces into one single piece and re-evaluate the corner points of the amalgamated piece to get curve segments for further matching. Applying this approach could resolve the T-joint issue.

Also for the future is extending this research to 3D, and actually reconstructing broken artifacts.

6 REFERENCES

[1] Abe K., Morii, R., and Kadonaga T. "Comparison of Methods for Detecting Corner Points from Digital Curves – A Preliminary Report," Proc. of the Second Intl. Conf. on Document Analysis and Recognition, pp. 854-857, 1993.

[2] Goldberg, D., Malon, C. and Bern, M., "A Global Approach to Automatic Solution of Jigsaw Puzzles," Proc. of the Eighteenth Annual Symp. on Computational Geometry, pp. 82-87, 2002.

[3] Guru, D., Dinesh, R. and Nagabhushan, P. "Boundary Based Corner Detection and Localization Using New 'Cornerity' Index: A Robust Approach," 1st Canadian Conf. on Computer and Robot Vision, pp . 417-423, 2004.

[4] Hood, J., Peng, Y. and Sims, D. "Locating Convex Corner Points on Discrete Closed Curves," Proc. of the Intl. Conf. on Image Processing, Computer Vision, & Pattern Recognition, pp. 309 – 313, 2007.

[5] Horst, J. and Beichl, I. "Efficient Piecewise Linear Approximation of Space Curves Using Chord and ARC Length," Proc. of the Society of Manufacturing Engineers Applied Machine Vision '96 Conf., 1996.

[6] Itoh, K. and Ohno, Y. "A Curve Fitting Algorithm for Character Font," Electronic Publishing, vol. **6** (3), pp. 195-205, 1993.

[7] Kampel, M. and Melero, J. "Virtual Vessel Reconstruction from a Fragment's Profile," 4th Intl. Symp. on Virtual Reality, Archaeology and Intelligent Cultural Heritage, 2003, pp. 79-88.

[8] Kong, W. and Kimia, B. "On Solving 2D and 3D Puzzles Using Curve Matching," IEEE Computer Society Conf. on Computer Vision and Pattern Recognition, pp. 583-590, 2001.

[9] Lee, S., Abbott, A., Clark, N. and Araman, P. "Spline curve matching with sparse knot sets: applications to deformable shape detection and recognition," Proc. of the 29th Annual Conf. of the IEEE Industrial Electronics Society, pp. 1808-1814, 2003.

[10] Leitão, H. and Stolfi, J. "A Multiscale Method for the Reassembly of Two-Dimensonal Fragmented Objects," IEEE Trans. on Pattern Analysis and Machine Intelligentce, vol. **24**(9), pp. 1239-1251, 2002.

[11] Sarfraz M., Rasheed, A. and Muzaffar Z. "A Novel Linear Time Corner Detection Algorithm," Conf. on Computer Graphics, Imaging and Visualisation, pp. 191-196, 2005.

[12] Shao, L. and Zhou, H. "Curve Fitting with Bézier Cubics," Graphical Model and Image Processing, vol. **58**(3), pp. 223-232, 1996.

[13] Smet, P. and Corluy, E. "High-precision Recomposition of Fragmented 2-D Objects," 14th Program for Research on Integrated Systems and Circuits workshop on Circuits, Systems and Signal Processing, pp. 432-436, 2003.

[14] Teh, C. and Chin, R. "On the Detection of Dominant Points on Digital Curves," IEEE Trans. on Pattern Analysis and Machine Intelligence, vol. **11**(8), pp. 859-872, 1989.

[15] Toe, T. and To, T. "Curve Matching by Using B-spline Curves," Intl. Conf. in Central Europe on Computer Graphics, Visualization and Computer Vision, pp. 173-176, 2004.

[16] Wolfson, H., Schonberg, E., Kalvin, A. and Lamdan Y. "Solving Jigsaw Puzzles by Computer," Annals of Operations Research, vol. **12**, pp. 51-64, 1998.

[17] Yacoub, S. and Menard, C. "Robust Axis Determination for Rotational Symmetric Objects out of Range Data," 21th Workshop of the Oeagm, pp. 197-202, 1997.