Partial Matching of Planar Polygons Under Translation and Rotation

Eric C. McCreath*

Abstract

Curve matching is an important computational task for domains such as: reconstruction of archaeological fragments, forensics investigation, measuring melodic similarity, and model-based object recognition. There are a variety of measures and algorithmic approaches used to address the curve matching problem including: shape signature strings with substring matching, geometric hashing, and Hausdorff distance approaches. In this paper we propose an approach that uses a turning function representation of the shape and also uses a L_2 measure for comparing matches. The novel algorithm presented finds the best match along a fixed length portion of two polygon's perimeters where the polygons may be arbitrarily translated and rotated. The algorithm's time complexity is $O(mn(n+m))$ where n and m are the numbers of vertices in the perimeters being matched. The utility of the algorithm is demonstrated in the reconstruction of a small jigsaw puzzle.

1 Introduction

Reconstruction of a broken object is an important yet time consuming task for a number of disciplines including forensics and archeology. Digitally automating or semi-automating this process is beneficial. Jigsaw puzzles, which are a simplistic form of this reconstruction problem, have been investigated by a number of researchers over the last 50 years. Freeman and Garder[6] produced what is generally considered the first of these investigations. They matched portions of a piece by comparing features extracted from the shape of those portions. There has since been a variety of other approaches in solving this problem including: curve matching combinatorial optimization[11], use of critical points[10, 8, 7], shape and image matching[12], and even an attempt to reconstruct the puzzle via a robot [4]. In many respects the jigsaw puzzle problem is a much simpler problem than the more general reconstruction of a broken fragment due to the well defined constraints on the shape of jigsaw puzzles, though it is not a simple problem to solve.

This paper proposes a novel algorithm which takes two polygons and matches a fixed length portion of the polygons perimeter. The polygons may be arbitrarily translated and rotated, however they are not scaled. This algorithm finds the match which minimizes the L_2 distance of the turning functions of the two portions of the polygons. The novelty of the approach taken in this paper is how a fixed length portion of two unscaled polygons can be matched. Such matching is useful for reconstruction when fragments are matched against the complement of other fragments.

Arkin et al. [3] proposed using the L_2 distance between turning functions of polygons to compare two shapes. Their algorithm works in $O(mn \log mn)$ time where n and m are the numbers of vertices in the polygons. The Arkin et al. approach is different to the contribution made in this paper as they find matches between two entire polygons which have both been rescaled to have a perimeter length of 1.

Cohen and Guibas [5] developed an algorithm that matches a polyline by translation, rotation, and scaling to a part of another polyline. Their algorithm works in $O(m^2n^2)$ time where m and n are the numbers of edges within the polylines. The approach taken in this paper is different to the Cohen and Guibas approach, as Cohen and Guibas finds the shifting and stretching parameters that minimize a combination of L_2 distance of the turning functions and match length. Whereas, in this paper, the approach presented finds the two shifting parameters which determine where the matching portions of the polygons will start.

Aloupis et al. [1, 2] developed an approach that finds the minimum area between two given orthogonal melodies with periods of 2π. Their approach runs in $O(n^2 \log n)$ time and can be used for matching short patterns in a database of music. This is the same problem of matching polygons when a turning function representation is used. The problem Aloupis et al. address is different to the problem this paper addresses as Aloupis et al. use an L_1 distance and they focus on comparing either two cyclic melodies (parallels with Arkin et al.) or a melody which matches a portion of another melody. Whereas, this paper uses the L_2 distance and focuses on fixed length portions as these portions could occur anywhere along the x-axis of the two turning functions. If the approach presented in this paper was applied to the music domain, then the approach could be used to find common melodies of fixed length which occur anywhere within two items of music.

*Department of Computer Science, The Australian National University, ACT 0200 Australia ericm@cs.anu.edu.au

2 Problem Setup

Let A and B denote two planar polygons with n and m vertices respectively. The vertices of A are points in the xy-plane denoted $\{\dot{a}_0, \dot{a}_1, \ldots, \dot{a}_{n-1}\}$. Vertex $\dot{a}_i$ is connected to vertex $\dot{a}_{i+1}$ by an edge. Also vertex $\dot{a}_{n-1}$ is connected to vertex $\dot{a}_0$ by an edge. To simplify the wrapping of the polygon we also define $\dot{a}_{i+n} = \dot{a}_i$, $\forall i \geq 0$. Let $\bar{a}_i$ denote the vector from $\dot{a}_i$ to $\dot{a}_{i+1}$, this vector provides the direction and length of edge i and may be calculated by $\bar{a}_i = \dot{a}_{i+1} - \dot{a}_i$. So $|\bar{a}_i|$ is the length of edge i. Let a_i denote the distance from $\dot{a}_0$ to $\dot{a}_i$ following the perimeter of the polygon. More formally $a_i = \sum_{j=0}^{i-1} |\bar{a}_j|$. Note that $a_0 = 0$ and a_n is the length of the perimeter. Let the turning function $t_A(d)$ be the accumulative turning angle at distance d around the perimeter of A from $\dot{a}_0$. $\dot{b}_i$, $\bar{b}_i$, b_i, and $t_B(d)$ are defined for polygon B in a similar way to that of polygon A.

Figure 1 shows an example of a simple polygon and its turning function representation. We wish to determine how well portions of one polygon will fit together with that of another. The turning function provides an efficient way of determining if polylines closely follow each other[3]. This efficiency is due to the translation invariant nature of the representation. Also, finding the best rotation of one polygon onto the other can be analytically determined without explicitly searching this dimension.

An L_2 distance is used over a fixed length l of the perimeter to determine the error in matching a particular configuration. Given this fixed perimeter length we must find the minimum error over a 3 dimensional space, where the dimensions are: the starting location s_A of the matching on the perimeter of polygon A; the starting location s_B on polygon B; and the angle of rotation θ. The start location s_A (and s_B) is the distance around the perimeter from a_0 (and b_0). Thus the error we wish to minimize over s_A, s_B, and θ is:

$$\text{error}(s_A, s_B, \theta) = \int_0^l (t_A(s_A + x) - t_B(s_B + x) + \theta)^2 dx$$

3 Searching

To search for the values that minimize the error, we first show how to calculate the θ that minimizes the error for a given s_A and s_B. We denote this minimum angle with the function $\theta^*(s_A, s_B)$. This calculation is done in the same way as [3]. We set the partial derivative of $\text{error}(s_A, s_B, \theta)$ to zero finding the only critical point at:

$$\theta = \frac{-\int_0^l t_A(s_A + x) - t_B(s_B + x)dx}{l}$$

A second derivative test reveals that this is the minimum. Thus we set:

$$\theta^*(s_A, s_B) = \frac{-\int_0^l t_A(s_A + x) - t_B(s_B + x)dx}{l}$$

This enables us to reduce the degrees of freedom for the search down to 2 as:

$$\min\{\text{error}(s_A, s_B, \theta)\} = \min\{\text{error}(s_A, s_B, \theta^*(s_A, s_B))\}$$

Let:

$$\begin{aligned}
\text{error}^*(s_A, s_B) &= \text{error}(s_A, s_B, \theta^*(s_A, s_B)) \\
&= \int_0^l (t_A(s_A + x) - t_B(s_B + x) \\
&\quad - \tfrac{\int_0^l t_A(s_A+x) - t_B(s_B+x)dx}{l})^2 dx \\
&= \int_0^l (t_A(s_A + x) - t_B(s_B + x))^2 dx - \\
&\quad \tfrac{(\int_0^l t_A(s_A+x) - t_B(s_B+x)dx)^2}{l} \\
&= II(s_A, s_B) - \tfrac{1}{l} I(s_A, s_B)^2
\end{aligned}$$

where

$$I(s_A, s_B) = \int_0^l t_A(s_A + x) - t_B(s_B + x)dx$$

and

$$II(s_A, s_B) = \int_0^l (t_A(s_A + x) - t_B(s_B + x))^2 dx$$

Since both s_A and s_B are continuous values it is impossible to explicitly search all possibilities. However, this search space may be partitioned by lines into a number of regions. The lines are either when vertices of the two polygons line up or when the start or end of the matching region lines up with a vertice on a polygon. The minimum of the error function over each region can be found at the crossing points on the border of the region. Hence, the minimum over the entire search space can be found by considering all the points at which these lines intersect.

As the functions t_A and t_B are both piecewise constant, both $t_A(s_A + x) - t_B(s_B + x)$ and $(t_A(s_A + x) - t_B(s_B + x))^2$ will also be piecewise constant functions in the variable x. Hence to calculate $I(s_A, s_B)$ and $II(s_A, a_B)$ one can simply sum the length of the contribution of each of the constant sections multiplied by the value for that section. We let $X_{ij}(s_A, s_B)$ be the length of the contribution made by the polygon sides $\bar{a}_i$ and $\bar{b}_j$. This can be calculated via:

$$\begin{aligned}
X_{ij}(s_A, s_B) = \quad & |(\max\{0, a_i - s_A, b_j - s_B\}, \\
& \min\{l, a_{i+1} - s_A, b_{j+1} - s_B\})|
\end{aligned}$$

where $|(x, y)| = \max\{y - x, 0\}$. We also let $\theta_{ij} = t_A(a_i) - t_B(b_j)$. So we now have:

$$I(s_A, s_B) = \sum_{ij} X_{ij}(s_A, s_B)\theta_{ij}$$

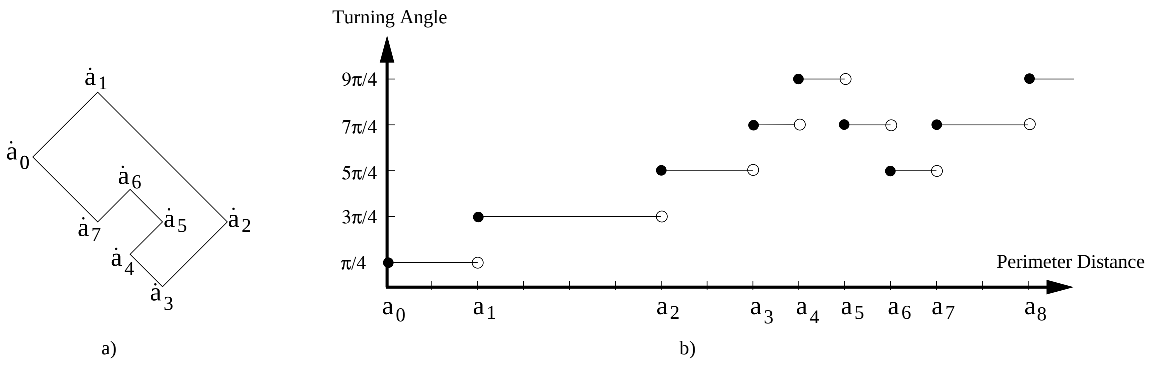

Figure 1: a) A simple polygon. b) The turning function for the polygon shown in a).

and

$$II(s_A, s_B) = \sum_{ij} X_{ij}(s_A, s_B)(\theta_{ij}^2)$$

The X_{ij} can be rewritten as a linear function in s_A and s_B for 10 different regions of the s_A, s_B plane. This can be done by considering: the three different possible maximums with the three possible minimums; along with the no overlapping possibility.

$$X_{ij}(s_A, s_B) = \begin{cases} l & \text{if } 0 \geq a_i - s_A \wedge \\ & 0 \geq b_j - s_B \wedge \\ & l \leq a_{i+1} - s_A \wedge \\ & l \leq b_{j+1} - s_B \\ a_{i+1} - s_A & \text{if } 0 \geq a_i - s_A \wedge \\ & 0 \geq b_j - s_B \wedge \\ & a_{i+1} - s_A \leq b_{j+1} - s_B \wedge \\ & a_{i+1} - s_A \leq l \wedge \\ & a_{i+1} - s_A \geq 0 \\ \vdots & \vdots \\ 0 & \text{otherwise} \end{cases}$$

The s_A,s_B plane can be divide up into regions via the following lines: $0 = a_i - s_A$, $0 = b_j - s_B$, $a_i - s_A = b_j - s_B$, $l = a_i - s_A$, and $l = b_j - s_B$. Within each of these regions X_{ij} will be linear with respect to s_A and s_B. Moreover, within each of these regions both $I(s_A, s_B)$ and $II(s_A, s_B)$ will be linear with respect to s_A and s_B. For each region r, let c_1, c_2, c_3, c_4, c_5, and c_6, be the constants such that:

$$I_r(s_A, s_B) = c_1 s_A + c_2 s_B + c_3$$

and

$$II_r(s_A, s_B) = c_4 s_A + c_5 s_B + c_6$$

then

$$\begin{aligned} \text{error}_r^*(s_A, s_B) &= II_r(s_A, s_B) - \tfrac{1}{l} I_r(s_A, s_B)^2 \\ &= -\tfrac{c_1^2}{l} s_A^2 - \tfrac{c_2^2}{l} s_B^2 - \tfrac{2c_1c_2}{l} s_A s_B + \\ & (c_4 - \tfrac{2c_1c_3}{l}) s_A + (c_5 - \tfrac{2c_2c_3}{l}) s_B + \\ & c_6 - \tfrac{c_3^2}{l} \end{aligned}$$

It is simple to confirm that the minimum of this function will be at one of the vertices of the region. Therefore, to find the minimum of this function over the entire s_A, s_B plane, one may simply find the minimum over all the points at which the lines cross. Fortunately, we do not need to recalculate I and II for every point, as we can move from one crossing point to a neighboring crossing point and evaluate the new error from information from the previous point in a constant amount of time.

4 The Algorithm

The algorithm works by calculating the error on each of the crossing points between the lines over the entire plane. There are at most[1] nm sloping lines $a_i - s_A = b_j - s_B$. These are all parallel with each other and have a gradient of -1. Also, there are at most $2n$ horizontal lines $0 = a_i - s_A$ or $l = a_i - s_A$. Finally, there are at most $2m$ vertical lines $0 = b_j - s_B$ or $l = b_j - s_B$. The sloping lines will intersect with both the vertical and horizontal lines. To calculate the minimum over all these intersecting points we consider each of the sloping lines in turn. We begin at any point on a line and calculate $I(s_A, s_B)$ and $II(s_A, s_B)$. This may be accomplished in $n+m$ steps by moving across the turning functions of A and B summing contributions to the integrals. Once this is calculated it is possible to slide along this sloping line to the next point where a vertical or horizontal line intersects with it. We denote this new point (s'_A, s'_B). $I(s'_A, s'_B)$ and $II(s'_A, s'_B)$ can be calculated using $I(s_A, s_B)$ and $II(s_A, s_B)$ and subtracting the contributions that no longer overlap and adding the new overlapping contributions. This may be achieved in constant time. Therefore, finding the minimum over all points which intersect with the sloping lines is $O(mn(m+n))$. The other points that must be considered occur when horizontal and vertical lines intersect. There are at most $4nm$ of these. The error for

[1]There could be fewer lines if a number of combinations of i and j produce the same line.

each of these can be calculated separately in at most $n + m$ steps. Hence, the complexity of calculating the points of intersection for both the horizontal and vertical lines is $O(mn(n+m))$. Moreover, the time complexity of calculating the configuration that minimizes error is $O(mn(n + m))$. Note that the space required for this algorithm is only $O(n + m)$.

5 Discussion

The matching algorithm was implemented and used within a puzzle solving program to demonstrate the utility of the matching algorithm. The data sets consisted of a simple 20 piece puzzle. The puzzle solving program used a greedy approach. The minimum error is found between each fragment and the complement of another fragment using the algorithm presented in this paper. The fragments with the minimum error are removed from the set of fragments, then the fragments are joined forming a new fragment which is then incorporated back into the set of fragments. This process is repeated until all the fragments are joined into a single fragment. Clearly, this greedy approach is not guaranteed to produce either an optimal or correct solution. However, in the puzzle tested the greedy approach produced a correct result. Note that, the fixed matching length was manually tune for this particular puzzle.

A larger class of shapes can be more compactly and accurately represented by including circular arcs as edges. In such cases the turning function is piecewise linear. Arkin et al. [3] considered this for matching shapes. In a similar way the algorithm presented in this paper could be extended to include circular arcs. Such a representation would clearly perform well for shapes like the puzzle fragments.

In terms of improving the performance of the algorithm it would be possible to use an approach similar to that of Latecki et al. [9] where polygons undergo a curve evolution to approximate a polygon with fewer edges. This approximation would be within some known error of the turning function. This could be used to prune large sections of the search, as bounds could be found for particular regions of the search space. The optimal configuration could then be found on this restricted search space. Hence, the overall algorithm would produce the optimal result more quickly. In general it is unlikely that such a modification would improve the worst case complexity of the algorithm, however, it could improve the expected running time of the algorithm.

References

[1] G. Aloupis, T. Fevens, S. Langerman, T. Matsui, A. Mesa, Y. Nuñez, D. Rappaport, and G. Toussaint. Algorithms for computing geometric measures of melodic similarity. *Comput. Music J.*, 30(3):67–76, 2006.

[2] G. Aloupis, T. Fevens, S. Langerman, T. Matsui, A. Mesa, D. Rappaport, and G. Toussaint. Computing the similarity of two melodies. In *Proceedings of the 15th Canadian Conference on Computational Geometry (CCCG'03)*, pages 81–84, 2003.

[3] E. M. Arkin, P. Chew, D. P. Huttenlocher, K. Kedem, and J. S. Mitchell. An efficiently computable metric for comparing polygonal shapes. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 13(3):209–216, March 1991.

[4] G. Burdea and H. Wolfson. Solving jigsaw puzzles by a robot. *IEEE Transactions on Robotics and Automation*, 5(6):752–764, Dec 1989.

[5] S. D. Cohen and L. J. Guibas. Partial matching of planar polylines under similarity transformations. In *SODA: ACM-SIAM Symposium on Discrete Algorithms (A Conference on Theoretical and Experimental Analysis of Discrete Algorithms)*, 1997.

[6] H. Freeman and L. Garder. Apictorial jigsaw puzzles: The computer solution of a problem in pattern recognition. *IEEE Transactions on Electronic Computers*, EC-13(2):118–127, April 1964.

[7] D. Goldberg, C. Malon, and M. Bern. A global approach to automatic solution of jigsaw puzzles. In *SCG '02: Proceedings of the eighteenth annual symposium on Computational geometry*, pages 82–87, New York, NY, USA, 2002. ACM Press.

[8] D. Kosiba, P. Devaux, S. Balasubramanian, T. Gandhi, and K. Kasturi. An automatic jigsaw puzzle solver. *Pattern Recognition, 1994. Vol. 1 - Conference A: Computer Vision & Image Processing., Proceedings of the 12th IAPR International Conference on*, 1:616–618 vol.1, 9-13 Oct 1994.

[9] L. J. Latecki and R. Lakamper. Shape similarity measures based on correspondence of visual parts. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(10), 2000.

[10] R. Webster, P. LaFollette, and R. Stafford. Isthmus critical points for solving jigsaw puzzles in computer vision. *IEEE Transactions on Systems, Man and Cybernetics*, 21(5):1271–1278, Sep/Oct 1991.

[11] H. Wolfson, E. Schonberg, A. Kalvin, and Y. Lamdan. Solving jigsaw puzzles by computer. *Ann. Oper. Res.*, 12(1-4):51–64, 1988.

[12] F.-H. Yao and G.-F. Shao. A shape and image merging technique to solve jigsaw puzzles. *Pattern Recognition Letters*, 24:1819–1835, 2003.