INTL JOURNAL OF ELECTRONICS AND TELECOMMUNICATIONS, 2010, VOL. 56, NO. 2, PP. 153-156
Manuscript received May 4, 2010; revised June, 2010. 10.2478/v10177-010-0020-5

# Recognizing Sharp Features of 2-D Shapes

Laxmi Gewali and Joseph P. Scanlan

***Abstract*—We present an efficient algorithm for recognizing and extracting sharp-features from complex polygonal shapes. The algorithm executes in $O(n^2)$ time, where $n$ is the number of vertices in the polygon. Sharp-feature extraction algorithms can be useful as a pre-processing step for measuring shape-similarity between polygonal shapes.**

***Keywords*—Shape recognition, shape similarity, boundary approximation, shape decomposition, shape simplification.**

## I. INTRODUCTION

PROBLEMS dealing with the recognition and simplification of two dimensional shapes have been investigated extensively in robotics, geographic information system, medical imaging, and computational geometry [1], [2], [3], [4], [5], [8], [6], [10], [11]. In computational geometry [1], [2], [3], [4], [5], [8], two dimensional shapes are usually modeled by polygons. An important sub-problem used in shape recognition is the formulation of "shape-similarity" measure. One of the first geometric algorithms for measuring shape similarity is based on the concept of "signature function" [8]. The signature function is essentially a rectilinear step function derived from the local and global properties of the polygonal shape. The signature of an edge $e$ of a polygon is obtained by accumulating the portion of the boundary of the polygon lying to the left of the line passing through $e$. The signature of the whole polygon is obtained by combining the signatures of all the edges of the polygon. If two shapes are similar then the area enclosed between their signatures is very small. For identical shapes the area enclosed between their signatures is zero. The technique of signature analysis has been found to be very effective for comparing orthogonal shapes and in recognizing hand-written characters [8]. Algorithms for measuring shape similarity based on signature functions are not easy to implement. Another approach for measuring shape similarity is based on the notion of "turning function". The turning function of a polygonal shape is also a rectilinear step function. Shape-similarity measures based on the turning function have been used successfully for developing efficient recognition algorithms and these algorithms are not difficult for practical implementation. One drawback of the turning function method is that it does not produce acceptable results when the boundary of the input polygon contains noise edges.

In Section 2 we present a review of the existing boundary simplification algorithms for two dimensional shapes. In Section 3, we propose a new technique for shape recognition. Our technique is based on a partitioning procedure that separates narrow regions from the core regions. For performing such partitioning we introduce the notion of 'sharp-features' for 2-d shapes. We present efficient algorithms for identifying such features. The sharp feature recognition algorithm runs in $O(n^2)$ time, where $n$ is the number of vertices in the polygon. In Section 4, we discuss possible extensions of the proposed technique.

L. Gewali and J. P. Scanlan are with the School of Computer Science University of Nevada, Las Vegas, USA (e-mail: laxmi@cs.unlv.edu, scanlanj@unlv.nevada.edu).

## II. BOUNDARY SIMPLIFICATION

For comparing the similarity between two complex polygonal shapes, it is necessary to approximate their boundaries by simpler ones. This kind of boundary simplification is needed in many other problem areas. In image processing systems, it is necessary to approximate a complex polygonal chain with a large number of vertices with a simpler one with fewer vertices. For example, medical and satellite images contain polygonal contours with exceedingly large number of vertices. Processing such images require prohibitively large amount of time [2]. Polygonal chain approximation algorithms are also useful for geometric compression when image data are transmitted through the World Wide Web.

The most critical issue in polygonal chain approximation is the formulation of rules for selecting vertices that are not contained in the simplified chain. The vertices picked for elimination should be such that the approximated chain retain the prominent features in the original polygonal chain. One of the early works on polygonal chain simplification is the approximation of piecewise linear functions [5]. Piecewise linear functions are basically polygonal chains that are monotone with respect to the x-axis. The approach used in [5] is to convert the chain approximation problem to the problem of computing several weak-visibility polygons. To approximate a monotone chain $Ch$ of $n$ vertices, a monotone polygon $P$ is constructed by laying down two parallel shifted copies of $Ch$ by $\epsilon$ each, where $\epsilon$ is the maximum allowed error bound for the approximation. After obtaining the monotone polygon $P$, a set of weak-visibility polygons are constructed. The "window" edges of the weak visibility polygons are used to determine the approximated chain. To compute the weak visibility polygons efficiently, a clever technique based on the convex hull of shifted chains is introduced in [5]. The execution time of the resulting algorithm is $O(n)$ which is very efficient. The number of vertices in the approximated chain depends on the value of allowed error bound $\epsilon$. The larger the value of $\epsilon$ the fewer are the number of vertices in the approximated chain.

An algorithm based on iterative vertex elimination is reported in [10]. In this approach, vertices for elimination are selected one at a time. If vertex $v_i$ is eliminated then it results $\epsilon_i$ approximation error. The algorithm eliminates vertices in the increasing order of the magnitude of the approximation error. The first vertex selected for elimination is the vertex that corresponds to the minimum approximation error. For three

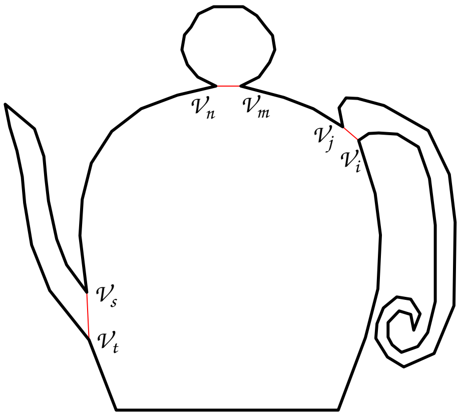


Fig. 1. A Polygonal Shape with Sharp Features.

consecutive vertices $v_{i-1}, v_i$ and $v_{i+1}$, the approximation error for $v_i$ is measured in two ways. In the first way it is given by the area of the triangle $T_i$ with vertices $v_{i-1}, v_i$ and $v_{i+1}$. The other way is to take the height of the triangle $T_i$. To make the algorithm efficient, a priority queue data structure is used to maintain vertices with minimum error bound. When a vertex is eliminated the error bound of other vertices change. It is shown in [10] that if $v_i$ is eliminated then only the error bound of its neighbors $v_{i-1}$ and $v_{i+1}$ are changed. This property is used to make the execution time of the resulting algorithm efficient. In fact, the time complexity of the iterative vertex elimination algorithm is $O(n \log n)$. Another notable beneficial property of the algorithm is that the quality of approximation does not depend on the choice of the starting vertex.

## III. Sharp Feature Recognition

Consider a polygonal shape $P$ whose vertices in counter-clockwise order along the boundary are $v_0, v_1, ..., v_{n-1}$. An example of polygonal shape (the outline of a tea pot) is shown in Figure 1. The *sub-polygon* $P(m, n)$ induced by vertices $v_m$ and $v_n$ is the portion of the polygon lying to the right of the diagonal $(v_m, v_n)$.

Some sub-polygons are *round* or *broad* and others are *sharp* or *skinny*. In Figure 2, the sub-polygon to the right of the diagonal $(v_m, v_n)$ is broad or round and those to the right of the diagonals $(v_s, v_t)$ and $(v_i, v_j)$ are *skinny* or *sharp*. We can formalize the notion of sharpness of a sub-polygon in terms of its structural properties. It can be observed that for a given perimeter, the area enclosed by a round sub-polygon is more than the area enclosed by a sharp sub-polygon. This leads us to the following definition.

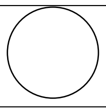
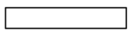
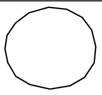
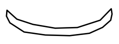
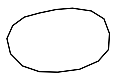
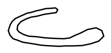
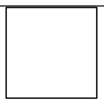
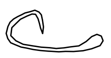
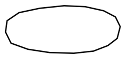
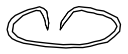
*Definition 1:* The *sharpness* of a sub-polygon $P(i, j)$ denoted $\alpha(i, j)$ is defined in terms of the ratio of perimeter squared over the enclosed area, i. e.

$$\alpha(i,j) = \frac{R(i,j)^2}{A(i,j)}$$

where $R(i, j)$ and $A(i, j)$ represent the perimeter and area of the of the sub-polygon $P(i, j)$, respectively.

It is noted that the value of sharpness is small for circular shapes and for skinny ones it can become very large. It can be easily verified that for circular shapes the sharpness tends to $4\pi$ and for very skinny ones it becomes very high, tending to infinity, for very very long, narrow hair-like shapes. Sharpness values for some interesting shapes are listed in Table 1.

TABLE I
Sample Sharpness Values

| | | | |
|---|---|---|---|
|  | 12.57 |  | 32.66 |
|  | 12.81 |  | 67.17 |
|  | 13.92 |  | 81.23 |
|  | 16.00 |  | 159.41 |
|  | 16.76 |  | 288.57 |

For identifying and extracting sharp-features (i.e. sharp sub-polygons) of a polygonal shape we use additional conditions of sub-polygons listed below. Our preliminary experiments suggest that polygons with $\alpha$ values $> 19$ are perceptually sharp. This leads us to the following condition.

*Condition 1 (Sharpness threshold):* A sub-polygon $P(i, j)$ is sharp if its sharpness value $\alpha(i, j)$ is at least 20.

As we closely examine the boundary of a complex polygon, we find sharp features starts from a reflex vertex. This leads us to the next condition:

*Condition 2 (Originating from reflex vertex):* At least one end-point of the diagonal separating a sharp feature is incident on a reflex vertex.

The area of a sharp feature must not be a large fraction of the area of the whole polygon. Analogous condition applies for the perimeter.

*Condition 3 (Threshold area and threshold perimeter):* The perimeter $R(i, j)$ (respectively area $A(i, j)$) of the sub-polygon $P(i, j)$ is no more than $\delta_1$ (respectively $\delta_2$) times the perimeter (respectively area) of the whole polygon. A typical value of $\delta_2$ could be 0.1.

For most maximal sharp features the diagonal that separates it from the whole polygon is of relatively shorter length.

*Condition 4 (Short Diagonal):* The diagonal $d = (v_i, v_j)$ on which the sub-polygon is subtended should be of short length. We found 0.04 of the total perimeter to work in our tests.

Now we describe the development of an algorithm for identifying sharp features of a polygon. The algorithm uses the visibility graph of the polygon to identify candidate diagonals on which sharp features can be subtended. Note that the visibility graph of a polygon is the graph $VG(V, E)$, where $V$ is the set of vertices of the polygon and $E$ is the set of its internal diagonals. The algorithm examines each diagonal from the visibility graph to determine the sharpness of the subtended sub-polygon $P(i, j)$. Only those sub-polygons are considered for sharpness computation that satisfy the above listed four conditions. The area of the polygon or sub-polygon is computed by using the following expression.

$$A(P) = \frac{1}{2}\sum_{i=1}^{n-1}(x_i + x_{i+1})(x_{i+1} - y_i)$$

The perimeter of a sub-polygon is determined in a straightforward manner by adding the lengths of the edges bounding the sub-polygon. The values of the area and perimeter are used to determine the sharpness value $\alpha$.

*Definition 2:* A diagonal whose sub-polygon satisfies sharpness condition is called a *feasible diagonal.*

*Definition 3:* A feasible diagonal inside a sharp feature is called a *dominated diagonal.*

*Definition 4:* A *prime diagonal* is a feasible diagonal that is not inside any sharp feature.

To recognize and extract all sharp features of a polygon we need to have areas $A(i, j)$'s and perimeters $R(i, j)$'s for all sub-polygons. A brute force approach would be to compute these quantities separately for all diagonals. Time needed to compute area $A(i, j)$ for one pair is $O(n)$. There can be $O(n^2)$ diagonals and consequently the total time for computing all sub-areas in this way is $O(n^3)$.

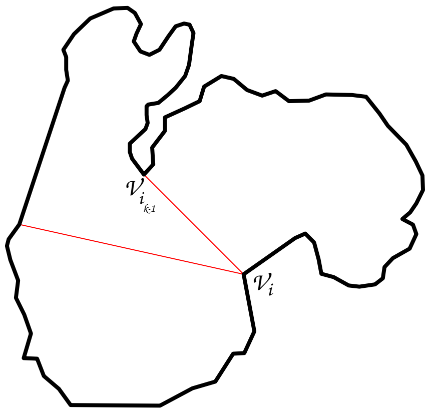


Fig. 2. Relating $A(i, i_{k-1})$ and $A(i, i_k)$.

We can develop a faster algorithm by making use of one sub-area to compute another related sub-area. Our approach is based on angularly sweeping the sub-areas corresponding to diagonals originating from a vertex. Let the list of diagonals in the counterclockwise angular order originating from vertex $v_i$ be $d_{i_1}, d_{i_2}, \ldots, d_{i_m}$. Suppose $A(i, i_{k-1})$ has been computed. Then $A(i, i_k)$ can be expressed as

$$A(i, i_k) = A(i, i_{k-1}) + A(v_i, v_{i_k}, v_{i_k+1}, \ldots, v_{i_{k+1}}) \qquad (1)$$

This is illustrated in Figure 2.

It is thus not necessary to recompute the area $A(i, i_k)$ when computing $A(i, i_{k+1})$. We can process the sub-areas of the diagonals in the angular order $d_i$, $d_{i_2}$, …, $d_{i_k}$, to obtain corresponding sub-areas and also sub-perimeters in $O(n)$ time. Consequently, all sub-areas and all sub-perimeters can be computed in $O(n^2)$ time. A formal listing of the algorithm is given as Algorithm AngularSweep.

Algorithm 1. Angular Sweep

```
1: Compute visibility graph $VG$ of polygon $P$;
2: foreach vertex $v_i$ of $P$ do
3:     Let $d_{i_1}, d_{i_2}, \ldots, d_{i_k}$ be the counterclockwise ordered
       list of diagonals emanating from vertex $v_i$;
4:     $A(i, j_0) = 0$;
5:     $R(i, j_0) = 0$;
6:     for $j = 1$ to $k$ do
7:         $A(i, i_j) =$
           $A(i, i_{j-1}) + Ar(v_i, v_{i_k-1}, v_{i_k-1+1}, \ldots, v_{i_k})$
           $R(i, i_j) =$
           $R(i, i_{j-1}) + Pr(v_i, v_{i_k-1}, v_{i_k-1+1}, \ldots, v_{i_k})$
8:     end for
9: end foreach
```

*Lemma 1:* Angular Sweep algorithm can be executed in $O(n^2)$ time.

*Proof:* The time for computing the visibility graph of a polygon (line 1) can be done in $O(n^2)$ time [9]. It is noted that the angularly ordered diagonals originating from a vertex can be extracted from the visibility graph in $O(n)$ time and hence the time needed for one execution of line 3 is $O(n)$. The for-loop in line 6 executes in $O(n)$ time. Hence the total time for all steps adds-up to $O(n^2)$. ■

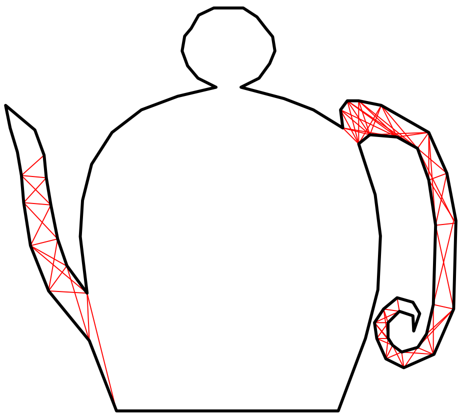


Fig. 3. Illustrating Feasible Diagonals.

The set of feasible diagonals that separate a sharp feature are illustrated in Figure 3. By examining the indices of of the vertices of a feasible diagonal $d_k$ we can determine whether or not $d_k$ is dominated. Let $I_{k_1}$ and $I_{k_2}$ be the indices of the end vertices of a feasible diagonal $d_k$. Let $J_{r_1}$ and $J_{r_2}$ be the indices of the end vertices of another feasible diagonal $d_r$. Then $d_k$ is dominated by $d_r$ if $I_{k_1}$ and $I_{k_2}$ are within the range of $J_{r_1}$ and $J_{r_2}$. When checking range it is necessary to take index addition modulo $n$. In this way all dominated diagonals are marked. The set of unmarked feasible diagonals are the prime diagonals. The sub-polygons subtended by prime diagonals give the sharp features. A formal sketch of the algorithm is listed as Algorithm Sharp Feature Recognition.

Algorithm 2. Sharp Feature Recognition

```
Input: A simple polygon $P$ with vertices
    $v_0, v_1, \ldots, v_{n-1}$
Output: Set of prime diagonals
1:  Compute area $A(P)$ of polygon $P$;
2:  Perform AngularSweep;
    // Determine feasible diagonals
3:  Mark all diagonals unfeasible;
4:  foreach diagonal $d_{ij}$ of $P$ do
5:      if $d_{ij}$ satisfies all conditions then mark $d_{ij}$ feasible
6:  end foreach
    // Determine prime diagonals
7:  Let $d_{yk}$ be the starting prime diagonal;
8:  while all vertices are not processed do
9:      foreach feasible diagonal $d$ within indexRange(y, k)
        do
10:         mark $d$ dominated;
11:     end foreach
12:     $d_{yk}$ = next prime diagonal;
13: end while
```

*Theorem 1:* All sharp features of a polygon can be recognized in $O(n^2)$ time.

*Proof:* Area of a polygon can be computed in $O(n)$ time by using the well known formula and hence line 1 takes $O(n)$ time. By Lemma 1, line 2 can be done in $O(n^2)$ time. There can be $O(n^2)$ diagonals in the worst case and hence the marking task in line 4 can take $O(n^2)$ time. Since all sub-areas and sub-polygons have been pre-computed, each of the four conditions for sharpness satisfaction can be verified in $O(n)$ time. Thus the first for-loop can be done in $O(n^2)$ time. The starting prime diagonal can be obtained in $O(n^2)$ time by scanning the diagonals along the boundary and by using the pre-computed sub-areas and sub-perimeters. Whether or not an index $i$ lies within the onther two indices $y$ and $k$ can be done in constant time and hence the while loop can be done in $O(n^2)$ time. The the total time complexity adds up to $O(n^2)$. ■

## IV. Discussion

We presented a formulation of the notion of sharp-feature in polygonal shapes. We also presented an $O(n^2)$ time algorithm for extracting all sharp-features of a polygon. Sharp-features can have a variety of structured complexities. If sharp-features are required to have monotone property or spiral property then they can be recognized much more efficiently. We conjecture that all monotone and spiral sharp-features can be recognized in one scan of the boundary of a polygon and consequently the time complexity of the recognition algorithm could be linear in the number of vertices $n$ and we are currently working in that direction. The notion of sharp-feature can be used as a pre-processing step for comparing similarity between polygonal shapes. A promising approach would be to first decompose the shapes into sharp and broad features and apply the exiting similarly measuring algorithms on each component separately. This will allow shape-similarity algorithms to tune-up and adjust according to the sharpness of the corresponding components. Many kinds of polygonal decomposition/partitioning tools are available [9]. We believe that the decomposition based on sharp-features would be useful for mesh generation and robotics.

## References

[1] E. M. Arkin, L. P. Chew, D. P. Huttenlocher, K. Kedem, and J. S. B. Mitchell, "An efficiently computable metric for comparing polygonal shapes," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 13, no. 1, pp. 209–215, 1991.

[2] O. Daescu and N. Mi, "Polygonal chain approximation: a query based approach," *Computational Geometry: Theory and Applications*, vol. 30, pp. 41–58, 2005.

[3] E. Gulbert and H. Lin, "B-spline curve smoothing under position constraints for line generalisation," in *Proceedings of ACM International Symposium on Advances in Geographic Information Systems*, 2006, pp. 3–10.

[4] M. Held and J. Eibl, "Biarc approximations of polygons within asymmetric tolerance bands," *Computer-Aided Design*, vol. 37, pp. 357–371, 2005.

[5] H. Imai and M. Iri, "Polygonal approximation of a curve," in *Computational Morphology*, G. T. Toussaint, Ed. Elsevier, 1988.

[6] D. T. Lee, "Similarity measurement using polygon curve representation and fourier descriptors for shape-based vertribral retrieval," in *Proceedings of SPIE*, vol. 5032, 2003, pp. 1283–1291.

[7] J. O'Rourke, http://cs.smith.edu/˜orourke.

[8] ——, "The signature of a plane curve," *Siam Journal on Computing*, vol. 15, no. 1, pp. 34–51, 1986.

[9] ——, *Computational Geometry in C (Second Edition).* Cambridge University Press, 1998.

[10] A. Pikaz and I. Distein, "An algorithm for polygonal approximation based on iterative point elimination," *Pattern Recognition Letters*, vol. 16, pp. 557–563, 1995.

[11] R. M. Rangayyan, D. Guliato, J. Carvalho, and S. Santiago, "Feature extractiom from the turning angle function for the classification of contours of breast tumors," *Journal of Digital Imaging*, 2007.