Optimal Contour Approximation and Applications

Yu-Tai Ching
Department of Computer and Information Science
National Chiao Tung University
Hsinchu, Taiwan
ytching@cis.nctu.edu.tw
FAX 03-572-1490

Abstract

A contour of n vertices is a polygonal path which is represented using a sequence of vertices $P = < v_0, v_1, \ldots, v_n >, v_0 = v_n$, where $\overline{v_i, v_{(i+1)}}$ is an edges on the path. A contour approximation problem is to substitute the contour P by another simpler contour Q while maintaining the deviation between P and Q. In this article, we present a result that we can find a Q with the least number of edges for a given error bound. **Keywords** Polygon approximation, level of detail, dynamic programming.

1 Introduction

We studied a contour approximation problem in this work. Given a contour P, we look for an approximation of P, denoted Q. Q has less number of edges than P does. Furthermore the deviation between Q and P is under an error allowance ϵ.

One of the motivations of this study is for an application in Geographic Information System. A contour map obtained from a Digital Terrain Map (DTM) generally contains many small line segments on each closed contour. It could take a large number of space to store a contour map. If a small error is allowed, we could have a chance to replace numeral consecutive line segments on the contour by using a line segment. We can use a contour with less number of line segments to approximate the original contour so that we can reduce the memory storage required. Another possible application is to reduce the triangular patches in surface representation of an object. Suppose that we use a set of triangle patches, which is obtained from a contour map, to model a terrain. If an error is allowed, we can reduce the number of triangles by reducing the number of points on the contour. Or in medical application, we can reconstruct geometric model of the region of interest (ROI) from a set of CT or MR volume images. The reconstruction is generally carried out by connecting a pair of contours representing the boundary of the ROI extracted from consecutive slices. Reducing the number of points on the contours can reduce the number of triangular patches.

Previous work having similar motivation was the studies of the geometric compression and the level of details[1, 2, 3, 4, 5, 6]. The formal discussed method to reduce the memory space required for storing triangle patches. And the level of detail studied the techniques to use finer meshes for the places

close to the view point and using coarse meshes for the places far away from the view point. The problem studied in this work is a view point independent level of detail study.

An approximation should meet an error constraint. The error describe how much details should be preserved.

In this article, we model the contour approximation problem as an optimization problem. A contour is a simple closed polygon. Let $P =< v_0, v_1, \ldots, v_n >, v_0 = v_n$ denote the contour. For a given ϵ, a path $< v_i, v_{i+1}, \ldots, v_j >$ on P is ϵ-approximated by the line segment $\overline{v_i, v_j}$ if the distance between every vertex to $\overline{v_i, v_j}$ is less than ϵ. Given a contour P and an error bound ϵ, the contour Q ϵ-approximates P if

1. $Q =< u_0, u_1, \ldots, u_m >$ consists of a subset of the points on P,
2. $u_k = v_i$ and $u_{k+1} = v_j$, every edge $\overline{u_k, u_{k+1}}$ on Q ϵ-approximates $< v_j, \ldots, v_k >$ on P.

Q is an optimal ϵ-approximation of P if the number of points on Q is minimized.

In this article, we show that the optimization problem can be solved using the dynamic programming technique.

In the next section, we describe the dynamic programming technique. Section 3 shows the experimental results.

2 Dynamic Programming

In this section, we present the dynamic programming algorithm to solve the optimization problem. We first consider the case of finding an optimal ϵ-approximation for a piecewise linear path $< v_i, v_{i+1}, \ldots, v_j >$. Let $U_{(i,j)}$ be a path that ϵ-approximates $< v_i, v_{i+1}, \ldots, v_j >$. The number of edges on $U_{(i,j)}$ is the cost for $U_{(i,j)}$. The least cost for $U_{(i,j)}$ is the optimal cost denoted $c_{(i,j)}$.

For the boundary condition that $i = j$, we let $c_{(i,j)} = 0$ since there are no edges. If $j > i$, there are two cases.

Case 1 $U_{(i,j)}$ is the line segment $\overline{v_i, v_j}$.
This case occurs when all the distances between vertices v_k, $i \leq k \leq j$, to $\overline{v_i, v_j}$ are less than ϵ. $\overline{v_i, v_j}$ ϵ-approximates $< v_i, v_{i+1}, \ldots, v_j >$ and thus $c_{(i,j)} = 1$.

Case 2 $U_{(i,j)}$ consists of two or more line segments.
In this case, $U_{(i,j)}$ can be divided into two paths $U_{(i,k)}$ and $U_{(k,j)}$ where v_k is a vertex on the polygonal path $< v_i, \ldots, v_j >$. Note that both $U_{(i,k)}$ and $U_{(k,j)}$ are the ϵ-approximations of the polygonal paths $< v_i, \ldots, v_k >$ and $< v_k, \ldots, v_j >$. The cost for $U_{(i,j)}$ is thus equal to the cost for $U_{(i,k)}$ plus the cost for $U_{(k,j)}$. Furthermore, suppose $U_{(i,j)}$ is the optimal ϵ-approximation of $< v_i, \ldots, v_j >$, both $U_{(i,k)}$ and $U_{(k,j)}$ must be the optimal ϵ-approximations for the paths $< v_i, \ldots, v_k >$ and $< v_k, \ldots, v_j >$. Since if this is not the case, improving either $U_{(i,k)}$ or $U_{(k,j)}$ reduce the cost for $U_{(i,j)}$, a contradiction to that $U_{(i,j)}$ is an optimal ϵ-approximation for $< v_i, \ldots, v_j >$. Since v_k can be any vertex on $< v_i, \ldots, v_j >$, the cost for the optimal ϵ-approximation $c_{(i,j)} = \min_{i<k<j} c_{(i,k)} + c_{(k,j)}$.

Based on the above discussion, the optimal cost can be written in the recurrence

$$\begin{cases} c_{(i,i)} = 0, \\ c_{(i,j)} = 1, \text{if } \overline{v_i, v_j} \ \epsilon-\text{approximates } < v_i, \ldots, v_j > \\ c_{(i,j)} = \min_{i<k<j} c_{(i,k)} + c_{(k,j)}. \end{cases} \quad (1)$$

To implement the dynamic program, we need two n by n matrices, `M` and `K`, where n is the number of vertices on the contour. `M[i][j]` stores the optimal cost for ϵ-approximation for $< v_i, \ldots, v_j >$. `K[i][j]` stores an integer k that the optimal approximation for $< v_i, \ldots, v_j >$ consists of the two ϵ-approximations for $< v_i, \ldots, v_k >$ and $< v_k, \ldots, v_j >$. `K[i][j] = i` if `M[i][j]` is

equal to 1. Since both `M[i][j]` and `K[i][j]` are triangular, these two matrices are stored in the matrix `M`. The cost for optimal ϵ-approximation for $< v_i, \ldots, v_j >$ is stored in `M[i][j]` and the corresponding k is stored in `M[j][i]`. The same as all the dynamic programming, the optimal solution is obtained in a bottom-up manner. In the ith iteration, we calculate the optimal ϵ-approximations for $< v_j, \ldots, v_{(j+i)} >$, $j = 0, \ldots, n-1-i$. using Equation 1. The implementation is summarized in the following pseudo code.

```
for (i=1;i<n;i++) {
     for (j=0;j<n-i;j++){
          Evaluate(j,(j+i),ϵ,MinCost,k);
          M[j][j+i]=MinCost;
          M[j+i][j]=k;
     }
}
```

The function `Evaluate` first determines whether $\overline{v_j, v_{j+i}}$ *epsilon*-approximates $< v_i, \ldots, v_{i+k} >$ or not. If this is the case, the function returns `MinCost` $= 1$ and `k` $=$ `j`. Otherwise, `Evaluate` computes `MinCost` and `k` using Equation 1.

Suppose that the polygonal path P is a closed contour, i.e., $v_0 = v_n$. The optimal ϵ-approximation from $v_0, \ldots, v_n$ might not be the optimal ϵ-approximation for the closed contour since that the optimal approximation might not contain the vertex v_0. In order to find the optimal ϵ-approximation for a closed contour, we can calculate the optimal ϵ-approximations for every $< v_i, \ldots, v_{(i+n) \bmod n} >$, $i = 0, \ldots, n-1$. The approximation has the least cost is the optimal ϵ-approximation.

We now analysis the time complexity for the dynamic programming approach. The time required for the function `Evaluate(j,(j+i),`ϵ`,MinCost,r)` is $O(i)$ since in the worst case, we need to calculate the distances between i points to the line segment $\overline{v_j, v_{(j+i)}}$. `Evaluate` is executed $O(n^2)$ times. The total time required is $O(n^3)$ since k is $O(n)$. To ϵ-approximate a closed contour, we calculate the ϵ-approximations for all the paths starting v_i, $i = 0, \ldots, n-1$. The total time required is $O(n^4)$.

3 Experimental Results

The proposed approach was tested in two cases. The first case was to reduce the number of vertices on a contour. The contour was obtained by converting a set DTM (digital terrain model). A contour had 1314 vertices was studied. The second case studied was the reconstruction of a pulmonary artery from a set of electron beam CT scan of heart. The boundary of the pulmonary artery in each slice of CT scan was obtained by intensity thresholding and a user interface process. The boundaries representation of the pulmonary artery were obtained by connecting the corresponding contours between consecutive slices. The dynamic programming approach was applied to minimize the number of points on each contour so that the number of triangles for the boundary representation are reduced.

Figure 1 shows a contour in 6 different resolutions. The coordinates of the points on the original contour are in the 202 by 384 window. The contour consists of 1314 vertices. The number of vertices on the ϵ-approximations for different ϵ are shown in Table 1. Note that there are only 43 vertices on the 2.0-approximation which is still very similar to the original contour.

Figure 2 shows the pulmonary artery reconstructed from a set of CT scan images. The four images are the rendered images obtained from different resolution of model.

The proposed algorithm calculate the optimal ϵ-approximation for a closed contour.

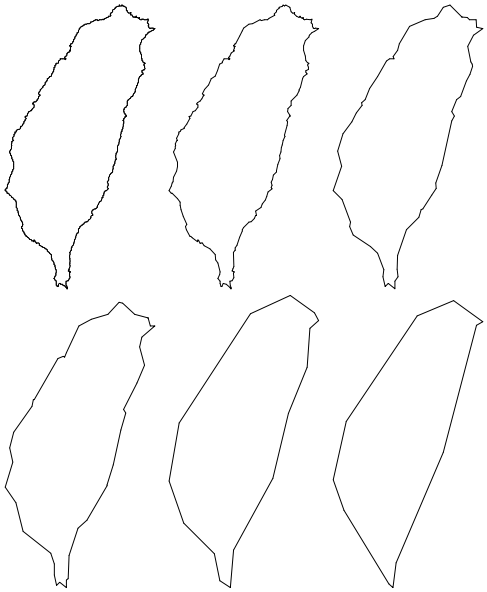

Figure 1: The above contours display the same contour in different resolutions. The above three images from left to right respectively have 1314, 220, and 64 line segments. The other three in the lower row have 43, 15, and 11 edges.

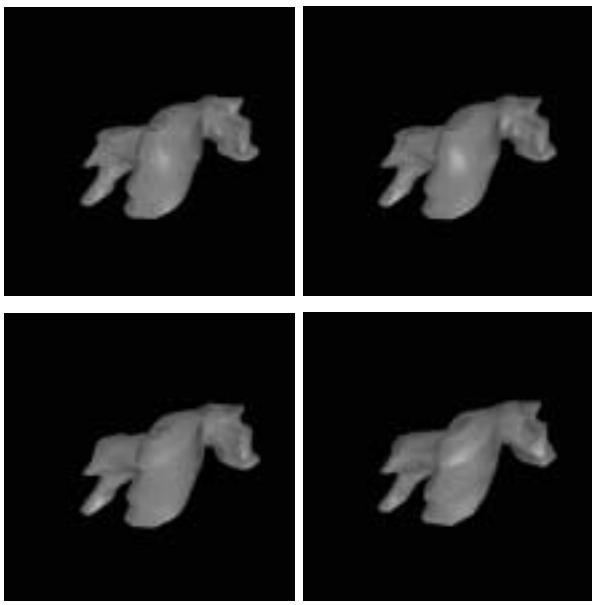

Figure 2: The pulmonary artery reconstructed from a set of CT scan of heart is shown in four different resolutions. The upper left image was rendered from a model containing 8338 triangles. The model for the upper right images consists of 3841 triangles. The images in the lower row were produced using models having 2445 and 1453 triangles respectively.

Table 1: The error bound and the number of vertices in the approximated contour. There are 1314 vertices in the original contour map.

ϵ	0.5	1.5	2.0	5.0	8.0
Number of vertices	220	64	43	15	11

Table 2: The computing time required for processing contours of different length using dynamic programming approach and greedy approach. The computing time is obtained from a PC with AMD k-3 400 CPU. The PC runs Linux operating system. Row A shows the number of vertices. Row B and C present the computing time used for dynamic programming approach and the greedy approach respectively.

A	86	153	1314	3030
B	0.06	0.19	68.39	859.7
C	< 0.02	< 0.03	0.13	0.26

However, the proposed approach need $O(n^2)$ memory space and $O(n^3)$ time to calculate the optimal ϵ-approximation for a contour starting a vertex. If the length of a contour is too long, the proposed algorithm is infeasible in both time and space required consideration. The computing time required for different length contour is shown in Table 2. When the length of a contour is greater than 1000, the computing time required is more than one minute. Table 2 also shows the computing time required for for the greedy approach. The computing time taken by the greedy approach is much less than the dynamic programming approach especially when the length of the contour is long. However, the greedy approach cannot find the optimal ϵ-approximation. The difference between the solution obtained from the greedy approach and the optimal ϵ-approximation increase as the length of the contour increase.

Table 3: The number of vertices, which obtained using different approach, on the ϵ-approximation for different length contours. The number of vertices on the original contours are shown in Row A. Row B and Row C demonstrate the number of vertices on the optimal contour and the contour obtained by using the greedy approach.

A	86	153	1314	3030
B	41	92	550	1509
C	44	93	565	1553

References

[1] Deering, M, "Geometric Compression", *Comput. Graph.* (Proc. SIGGRAPH), 1995, pp. 13-20.

[2] Eck, M., DeRose, T., Duchamp, T., Hoppe, H., Lounsbery, M., and Stuetzie, W., "Multiresolution Analysis of Arbitrary Meshes," *Comput. Graph.* (Proc. SIGGRAPH) 1995, pp. 173-182.

[3] Gueziec, A., "Surface Simplification with Variable Tolerance," Second Annual International Symposium on Medical Robotics and Computer Assisted Surgery, Baltimore, MD., 1995.

[4] Hoppe, H., "Progressive Meshes," *Compu. Graph.*, (Proc. SIGGRAPH) 1996, pp. 99-108.

[5] Hoppe, H., DeRose, T., McDonald, J., and Stuetzle, W., "Mesh Optimization," *Compu. Graph.*, (Proc. SIGGRAPH), 1993, pp. 19-26.

[6] Kalvin, A., and Taylor, R. H., "Superfaces: Polygonal Mesh Simplification with Bounded Error," *IEEE Comput. Graph. Appl. 16, 3*, 1996, pp. 19-26.