Polyline Simplification Using Quadric Error Metric with Bounded Error

BY

Sheung-Hung Poon

A Thesis Presented to

The Hong Kong University of Science and Technology

In Partial Fulfillment

of the Requirements for

the Degree of Master of Philosophy

in Computer Science

Hong Kong, January 1999

Authorization Page

Polyline Simplification Using Quadric Error Metric with Bounded Error

BY

Sheung-Hung Poon

APPROVED:

SUPERVISOR, SUPERVISOR

PROF. ROLAND T. CHIN, HEAD OF DEPARTMENT

Department of Computer Science

20 January 1999

Polyline Simplification Using Quadric Error Metric with Bounded Error

BY

Sheung-Hung Poon

A Thesis Presented to

The Hong Kong University of Science and Technology

In Partial Fulfillment

of the Requirements for

the Degree of Master of Philosophy

in Computer Science

Hong Kong, January 1999

Abstract

We study the problem of polygonal line simplification. The objective is to seek a polygonal line of smaller size that approximates the original one well. We present an algorithm that is based on edge contraction. An edge contraction merges two adjacent vertices into a new vertex and this new vertex will be made the new endpoint of the uncontracted edges incident to the two vertices merged. Thus, repeated applications naturally yield a simplification algorithm. We implemented three algorithms QG, QGG, and QLG based on this approach. QGG and QG support each edge contraction in $O(\log m)$ time, where m is the size of the current polygonal line, whereas QLG supports each edge contraction in $O(1)$ time. The selection of the edge to be contracted is based on the quadric error produced, which was introduced by Garland and Heckbert in simplifying 3D polygonal surfaces. We use and improve their algorithm in 2D. If

the quadric error of each edge contracted is at most ϵ^2, we can guarantee that the directed Hausdorff distance from the simplified line to the original one is within ϵ. We can supply an error tolerance ϵ to QG, QGG and QLG and let edges be contracted repeatedly until the current minimum quadric error exceeds ϵ^2. On the other hand QGG and QG can also allow the user to directly and interactively reduce the number of edges in the simplified line when deemed necessary. We have conducted some experiments to measure the approximation error of the simplified lines produced by our algorithms.

Chapter 1

Introduction

The automation of mapmaking, initiated during 1950s, has become increasingly popular in the field of cartography. A major effort in automating the mapmaking process has been the development of methods for generalizing digital data. There are four significant components of automatic generalization which include simplification, smoothing, displacement, and enhancement.

This thesis describes an edge contraction approach to do simplification. The remaining three components are briefly described in the following. Detailed descriptions can be found in [4]. Smoothing routines relocate or shift coordinate pairs in an attempt to smooth out small perturbations and capture only the significant trends of the polygonal lines. Displacement routines are concerned about shifting between polygonal lines to prevent coalescence or overlap at a reduced scale. Finally, enhancement routines allow detail to be added to the already simplified data set. Recently some geographers have become interested in applying the theory of fractals for the purpose of enhancement.

We now turn to have an overview on the component of simplification. Simplification algorithms are concerned about identifying and eliminating superfluous coordinates. They are performed by applying a variety of mathematical/statistical criteria, such as angular deviation or distance between points/segments. We will summarize most of the existing simplification algorithms in Chapter 2.

1.1 Thesis objectives and motivation

In this thesis, we use edge contraction based on quadric error to do the simplification. The approach was originally designed for 3-dimensional surface simplification by Garland and Heckbert [1]. We found that their results preserved many important features of the original surface even though the percentage of faces reduced was up to 80 or 90. We suspect that there is a way to quantify the approximation error. Since a 3-dimensional surface can be very complex, we turn to 2-dimension, and succeed in improving the edge contraction approach and prove an approximation error bound.

1.2 Thesis organization

Chapter 2 reviews on the previous work on polygonal line simplification. In Chapter 3, we present some preliminaries and define some basic concepts for our algorithms and proofs.

In Chapter 4, we describe an basic edge contraction algorithm, which in fact is the 2-dimensional version of Garland and Heckbert's edge contraction surface simplification algorithm [1]. Chapter 5 points out the deficiencies of this basic algorithm, and proposes the remedies. We describe three algorithms based on the improved edge contraction procedure. In Chapter 6, we prove the bound on the directed Hausdorff distance from the simplified line to the original one.

In Chapter 7, we present the experimental results which compare the basic algorithms, the three new algorithms and the Douglas and Peucker algorithm (DP), which is a the high-quality algorithm used popularly in cartography. Unlike two of the new algorithms QG and QGG, DP does not allow the user to directly and interactively increase the simplification. Thus, our experiments are by no means to show that our algorithms can replace DP. In fact, DP outperforms our algorithms for most of the cases in terms of approximation quality. Our objective is to verify whether the output quality of our algorithms are close to that of DP.

Finally, we conclude the thesis in Chapter 8, and suggest some directions for future work in Chapter 9.

In Appendix **??**, we supplement the thesis by presenting the experimental graphs of more than 10 data sets for error analysis of the outputs of the algorithms.

Chapter 2

Previous Work on Polyline Simplification

The polygonal line simplification problem is a well-studied problem in various disciplines including geographical information systems (GIS) [4, 6, 7, 8, 9, 10], digital image analysis [11, 12, 13], and computational geometry [14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25]. We first summarize the algorithms used in cartography. And then we will have a survey on the theoretical algorithms in computational geometry.

2.1 Algorithms in cartography

There are various kinds of classifications for these many different methods. we will stick to the classification proposed by McMaster [4], who claimed that some other classifications seemed to be inadequate. Although the original classification has five categories, I will only use four categories by combining two of them. The difference is that the third category described below was further divided into two categories, called constrained and unconstrained ones.

The algorithms are divided into four categories: 1) Independent point algorithms, 2) Local processing algorithms, 3) Extended local processing algorithms, and 4) Global algorithms.

2.1.1 Independent point algorithms

These routines are very simple in nature and do not, in any way, account for any meaningful relationship with the neighboring points. The best example is the nth point routine where every nth point (eg. 3rd, 6th, 9th, ...) is retained. Although this routine is simple and efficient, it is useless in any application for high-quality approximation. However, it have some utility when we are eliminating points on extremely dense polyline.

2.1.2 Local processing algorithms

Local processing routines utilize the characteristics of the immediate neighboring points to determine whether to retain the points. For instance, an algorithm eliminates points if they are closer together than a prescribed distance. Other examples are Tobler's algorithm (1965) [4] and Jenk's algorithms [4].

2.1.3 Extended local processing algorithms

Unlike local routines, these algorithms search beyond the immediate neighboring points and evaluate sections/parts of the polygonal line. In this category, some routines are less constrained in their search, and others may be more constrained.

One example of extended processing routine with constraint is Lang's routine (1969) [4]. It uses a Euclidean distance measure for vertex removal. It operates on six consecutive vertices $v_0, v_1, \dots, v_5$ at a time. First we connects v_0 to v_5. Let ℓ be the supporting line of v_0v_5. If the distance of any $v_i (1 \leq i \leq 4)$ from ℓ exceeds the input error tolerance ϵ, ℓ is repositioned from points v_0 to v_4 and the distances from this line to the remaining intermediate points are calculated. As soon as all perpendicular distances between the two end points are less than ϵ, the end point v_4 is retained, v_4 becomes the new v_0, and the next segment of the polygonal line is processed.

Other examples are Reumann-Witkam's routine (1974), Opheim's routines (1981, 1982) and Johannsen's routine (1974) [4].

2.1.4 Global algorithms

This category, unlike others, considers the polyline in its entirety while processing. Another difference is that the algorithms in the previous categories are sequential. The global simplification algorithm commonly used in cartography was developed by Douglas and Peucker (1973) [6]. It is called the *Douglas and Peucker Algorithm* (DP). This method is summarized by the authors as follows:

> This method begins by defining the first point on the line as an anchor and the last point as a floating point. These two points define a straight segment. The intervening points along the curved line are examined to find the one with the greatest perpendicular distance between it and the straight line defined by the anchor and the floater. If this distance is less than the maximum tolerable distance, the straight segment is deemed suitable to represent the whole line. in the case where the condition is not met, the point lying furthest away becomes the new floating point [6].

For more details on this algorithm, please refer to [4, 6]. This algorithm has been reported to be superior in choosing critical points [6, 26, 27]. The asymptotic time complexity has been reduce to $O(n \log n)$ by Hershberger and Snoeyink [9] from its original $O(n^2)$. The routine uses the data structure of dynamic path hull to keep track of the farthest point from a line. However, the overhead increases so much that the improved algorithm runs slightly slower than the original version for the statistical properties of cartographic data [28]. Recently, Hershberger and Snoeyink [25] further reduced the running time to $O(n \log^* n)$ again. However, the improvement itself is more of theoretical interest.

2.2 Algorithms in computational geometry

The first category below, in fact, does not address polyline simplification directly. We put it here because the algorithms in the categories following it will use some results of them.

2.2.1 Homotopic path and loop results

Hershberger and Snoeyink defined the concepts of homotopic path and loop in [14]. Two paths among obstacles in the plane are *homotopic* if they are the same endpoints and they can be deformed continuously to each other without leaving the plane. Similarly, two loops among obstacles in the plane are *homotopic* if they can be deformed continuously to each other without leaving the plane.

Given a path α inside a triangulated polygon possibly with holes, Hershberger and Snoeyink [14]

presented linear time algorithm to compute a shortest path homotopic to α and a minimum-link path (a path with minimum number of edges) homotopic to α. Note that the minimum-link path returned may be self-intersecting. Similarly, given a loop α inside a triangulated polygon possibly with holes, a shortest loop homotopic to α can be computed in linear time. However, it is not known how to compute a homotopic loop with minimum edges. Hershberger and Snoeyink [14] showed how to compute in linear time a homotopic loop with at most one more than the minimum number of edges. Again, this homotopic loop may be self-intersecting.
In [16], Kahan and Snoeyink showed that representing the minimum-link path may require $O(n^2 \log n)$ bits while the input polygon requires only $O(n \log n)$ bits. Thus, all the linear-time algorithms for minimum-link paths heavily depend on the theoretical assumption that storing and carrying out a single arithmetic operation involving real numbers can be done in constant time.

2.2.2 Approximating x-monotone polyline using uniform error metric

According to different requirements and error metric, different algorithms arouse. Generally, researchers are interested in two versions of the simplification problem:

- min-# problem: given an input error tolerance ϵ, minimize the number of edges, and
- min-ϵ problem: given a bound on the number of edges in the simplified line, minimize the error ϵ.

For a x-monotone polyline using uniform error metric, Hakimi and Schmeichel [15] solved the min-# problem in $O(n)$ time. The trick is to fatten the monotone line to a polygon, and then apply a linear time algorithm similar to the minimum-link homotopic path algorithm. On the other hand, the min-ϵ problem can be solved in $O(n^2 \log n)$ time. The trick is to identify all "critical values" of ϵ (i.e., for any no. of edges, the minimum error must be equal to one of these critical values). There are $O(n^2)$ of these values. After sorting these values in increasing order, the min-ϵ problem can be solved by binary searching among these values and solving the corresponding min-# problems. Later, Wang et al. [17] used a clever plane-sweep approach to improve the running time to $O(n^2)$. The running time has subsequently been improved further to to $O(n \log n)$ by Goodrich [18] using parametric searching.

2.2.3 Approximating a general polyline

For general polyline, Imai and Iri [23] proposed a version of the problem in which vertices of simplified line must be vertices of the original line, and the simplified line must visit these vertices in the same order as they appear in the original line. Thus, the simplification can be viewed as replacing a subchain between two vertices by a line segment between them. Imai and Iri [23] proposed several error metrics. One of them has subsequently been studied by others:

Error of joining v_iv_k equals the maximum distance of v_j from v_iv_k for $i \leq j \leq k$.

Note that this error measure will finally guarantee a bound on the Hausdorff distance between the simplified line and the original line.

Melkman and O'Rourke [24] solved the min-# problem in the above setting in $O(n^2 \log n)$ time. Later, Chan and Chin [20] improved the running time to $O(n^2)$ time which also implied an $O(n^2 \log n)$ time algorithm for the min-ϵ problem. Chan and Chin [20] also studied the min-# and min-ϵ problems for simplifying a closed polygonal line. For a closed polyline, the min-# problem can be solved in $S(n)$ time where $S(n)$ is the running time for solving the all-pairs shortest path problem in an unweighted directed graph while the min-ϵ problem can be solved in $S(n) \log n$ time.

Guibas et al. [22] proposed other versions of the problem in which vertices of the simplified line need not be vertices of the original line. They only dealt with the min-# problem. One approach is to fatten the polygonal line to a "splinegon" first by convolving it with a disk of radius ϵ. Now the input line is a path within this splinegon. Then the splinegon is triangulated. Afterwards, they apply the homotopic minimum-link path algorithm in [14] to compute the simplified line. The fattening and triangulation are expensive operations in practice.

Guibas et al. [22] also proposed other versions of the min-# problem which place further requirements on the ordering and locations of the vertices of the simplified line. The simplified line is required to stab the fattened vertices (disks of radius ϵ centered at the vertices), as well as the fattened edges (convex hull of the adjacent fattened vertices) in order. New vertices of the simplified line can either 1) lie anywhere (which implies that the directed Hausdorff distance is not bounded), 2) lie inside fattened edges, or 3) lie inside fattened vertices. If new vertices can lie anywhere or must be inside fattened edges, the minimum-link path can be found in $O(n^2 \log n)$ time. This also uses a graph approach like in [23]. If new vertices must lie inside fattened vertices, a path with at most twice as many edges as the minimum-link path can be computed in linear time using a simple greedy algorithm. If the fattened vertices are disjoint, and new vertices can lie anywhere or inside fattened edges, a minimum-link path can be computed in linear time using a greedy approach.

2.2.4 Negative results

Guibas et al. [22] proved two negative results in some settings. Given a subdivision S inside a region R, computing a subdivision S' with minimum-link so that S and S' can be continuously deformed into each other within R (homotopic) is NP-hard. Given a simple polygon curve α inside a region R, computing a simple polygon curve α' with minimum-link so that α and α' can be continuously deformed into each other within R (homotopic) is also NP-hard.
Thus the general problem of simplifying a map in GIS research is likely to be difficult. Also, optimizing the size of the simplified line while enforcing simplicity is probably a difficult problem too.

Chapter 3

Preliminaries and Definitions

A *polygonal line* or simply *polyline* L is a contiguous sequence of edges in the plane. We denote L by a sequence of vertices $v_0 \ldots v_m$ where $v_{i-1}v_i$, $1 \leq i \leq m$, is an *edge*. A vertex v_i is an *interior vertex* of L if it is adjacent to two vertices in L. An edge v_iv_{i+1} is an *interior edge* of L if both v_i and v_{i+1} are interior vertices. A *subchain* of L is a polygonal line represented by a contiguous subsequence of vertices.

L is *monotone* if the intersection between L and any vertical line is a single contiguous interval, which can degenerate to a single point. L is *convex* if L is monotone and for any two points x and y on L, the line segment xy lies above the subchain between x and y. Similarly, L is *concave* if L is monotone and for any two points x and y on L, the line segment xy lies below the subchain between x and y. Traverse L from v_0 to v_m and orient the supporting lines l_i

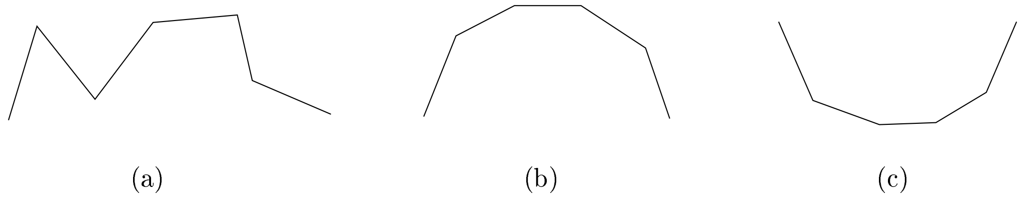

Figure 3.1: (a) A monotone polyline; (b) a concave polyline; and (c) a convex polyline.

for $1 \leq i \leq m$ consistently. The *turning angle* at an interior vertex v_i is negative (positive) if v_iv_{i+1} lies on the left (right) of the oriented l_i. The magnitude of the turning angle at v_i is the smallest rotation angle for the oriented l_i about v_i to align with the oriented l_{i+1}. The turning angle is zero if v_iv_{i+1} lies on l_i. L is called a *spiral* if no two turning angles are of different signs. L is a *inflection* polygonal line if it is not a spiral. In this case, L must contain an edge

v_iv_{i+1} such that v_i and v_{i+1} are interior vertices and the signs of the turning angles at v_i and v_{i+1} are different. We call v_iv_{i+1} an *inflection edge*. We call L *nice* if the magnitude of each

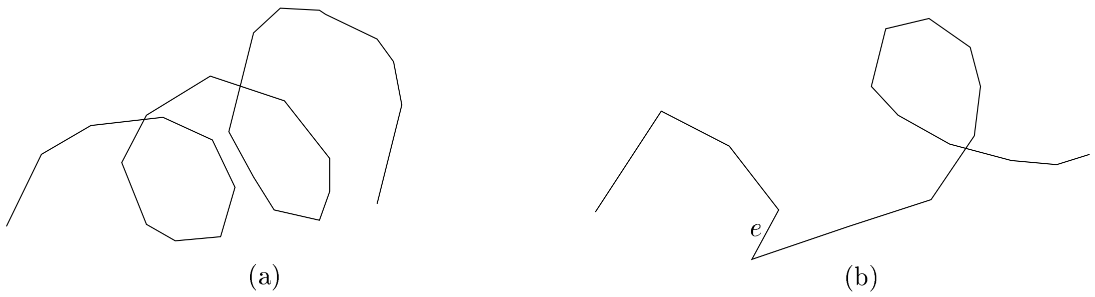

Figure 3.2: (a) A spiral; and (b) A non-spiral where e is a inflection edge.

turning angle is at most $\pi/2$.

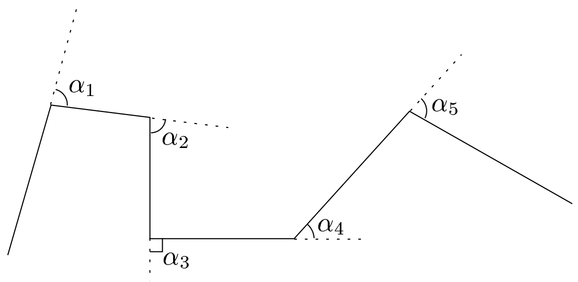

Figure 3.3: A nice polyline where all turning angles $\alpha_1, \alpha_2, \ldots, \alpha_5$ are $\leq \pi/2$.

Each edge e in L induces a *strip* which is the region between the two perpendicular lines through the endpoints of e. Each interior vertex v_i of L induces a *wedge* defined as follows. Shoot two rays from v_i normal to $v_{i-1}v_i$ and v_iv_{i+1} towards the larger angle at v_i. The cone between these two rays is the wedge induced by v_i. See Figure 3.4 for an example.

The *directed Hausdorff distance* from a point v to an object $\mathcal{O}$ is the minimum Euclidean distance from v to $\mathcal{O}$. We use $H(v, \mathcal{O})$ to denote the square of this distance. The directed Hausdorff distance from $\mathcal{O}'$ to $\mathcal{O}$ is the maximum directed Hausdorff distance from any point v in $\mathcal{O}'$ to $\mathcal{O}$.

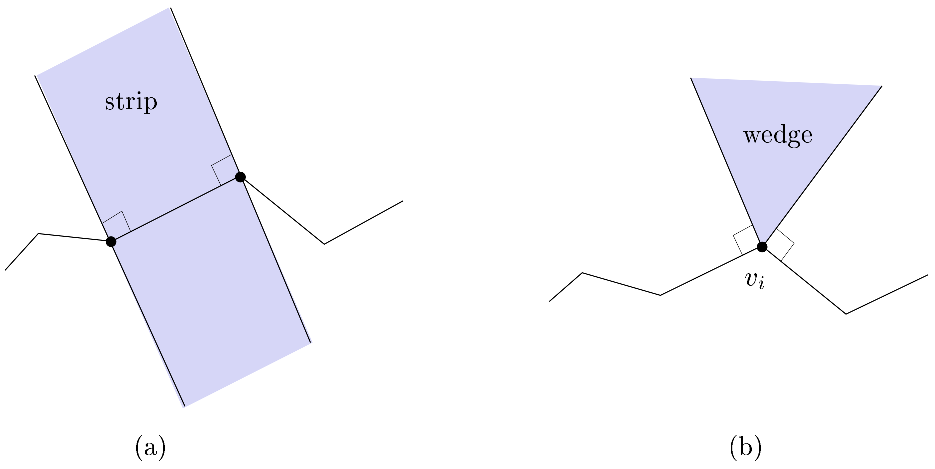

Figure 3.4: (a) A strip induced by edge e; and (b) a wedge induced by interior vertex v_i.

Chapter 4

An Edge Contraction Algorithm

4.1 Quadric error

An *edge contraction* merges the two endpoints of an edge uu' into a new vertex v, which is called a *destination vertex*. Sometimes the edge contraction is denoted by $(u, u') \rightarrow v$.The figure below is an example of edge contraction. The uncontracted edges incident to v_i and v_{i+1} will become incident to v. This leaves us the issue of fixing the location of v which is related to

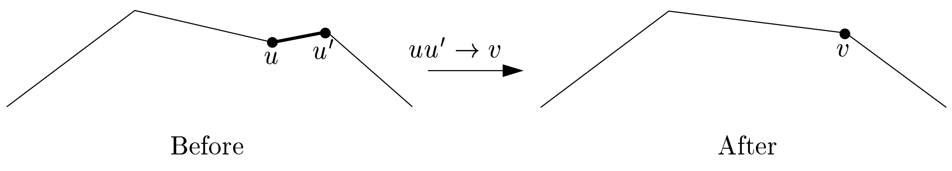

Figure 4.1: Contract edge uu' to destination vertex v.

the objective that the simplified line should approximate the original one well. That is, an error measurement should be developed. In the related work of simplifying 3D polygonal surfaces [1], Garland and Heckbert defined a notion of *quadric error*. We translate that to 2D. In this case, the quadric error of contracting v_iv_{i+1} to v is $H(v, l_i) + 2H(v, l_{i+1}) + H(v, l_{i+2})$. Thus, v is to be fixed such that $H(v, l_i) + 2H(v, l_{i+1}) + H(v, l_{i+2})$ is minimized.

We represent v using *homogeneous coordinates* $(x, y, 1)$. Let l_i be the line $a_ix + b_iy + c_i = 0$, where $a_i^2 + b_i^2 = 1$. We define a *line matrix* K_{l_i} for l_i to be

$$\begin{pmatrix} a_i^2 & a_ib_i & a_ic_i \\ a_ib_i & b_i^2 & b_ic_i \\ a_ic_i & b_ic_i & c_i^2 \end{pmatrix}.$$

Then $H(v, l_i) = (a_i x + b_i y + c_i)^2$ which can be written as $(x, y, 1)K_{l_i}(x, y, 1)^t$. Thus, we want to fix v such that $\Pi = (x, y, 1)(K_{l_i} + 2K_{l_{i+1}} + K_{l_{i+2}})(x, y, 1)^t$ is minimized. We define a *vertex matrix* $M_{v_i} = K_{l_i} + K_{l_{i+1}}$ for each vertex v_i. Thus, Π can be rewritten as $(x, y, 1)(M_{v_i} + M_{v_{i+1}})(x, y, 1)^t$. Furthermore, if we define $M_{v_i} + M_{v_{i+1}}$ to be the vertex matrix of the new vertex v, then the quadric error Π can be simplified to be $(x, y, 1)M_v(x, y, 1)^t$.

Suppose that M_v equals

$$\begin{pmatrix} m_{11} & m_{12} & m_{13} \\ m_{12} & m_{22} & m_{23} \\ m_{13} & m_{23} & m_{33} \end{pmatrix}.$$

Then Π becomes $m_{11}x^2 + 2m_{12}xy + 2m_{13}x + m_{22}y^2 + 2m_{23}y + m_{33}$. At the minimum, $\partial\Pi/\partial x = 0$ and $\partial\Pi/\partial y = 0$. Thus, we have the following system of linear equations

$$\begin{pmatrix} m_{11} & m_{12} & m_{13} \\ m_{12} & m_{22} & m_{23} \\ 0 & 0 & 1 \end{pmatrix}\begin{pmatrix} x \\ y \\ 1 \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix}.$$

whose solution yields the optimal location for v. Note that the last matrix is identical to M_v except that the last row is replaced by $(0, 0, 1)$. To see that we indeed obtain the minimum (instead of the maximum or other possibilities), let's check the determinant of second order derivatives

$$\begin{vmatrix} \partial^2\Pi/\partial x^2 & \partial^2\Pi/\partial y\partial x \\ \partial^2\Pi/\partial x\partial y & \partial^2\Pi/\partial y^2 \end{vmatrix},$$

which equals

$$\begin{vmatrix} 2m_{11} & 2m_{12} \\ 2m_{12} & 2m_{22} \end{vmatrix}.$$

The above determinant equals $4(m_{11}m_{22} - m_{12}^2)$. Observe that by using a multiset $I = \{i, i+1, i+1, i+2\}$ of indices, $(m_{11}m_{22} - m_{12}^2)$ can be written as

$$\begin{aligned}
& (\textstyle\sum_{i\in I} a_i^2)(\sum_{i\in I} b_i^2) - (\sum_{i\in I} a_i b_i)^2) \\
= & \textstyle\sum_{i\neq j\in I}(a_i^2 b_j^2 - a_i b_i a_j b_j) \\
= & \textstyle\sum_{i<j\in I}(a_i^2 b_j^2 - a_i b_i a_j b_j + a_j^2 b_i^2 - a_j b_j a_i b_i) \\
= & \textstyle\sum_{i<j\in I}(a_i^2 b_j^2 - 2a_i b_i a_j b_j + a_j^2 b_i^2) \\
= & \textstyle\sum_{i<j\in I}(a_i b_j - a_j b_i)^2 \\
= & \textstyle\sum_{i<j\in I} a_i^2 b_j^2 [1 - (a_j/b_j)(b_i/a_i)]^2.
\end{aligned}$$

Thus, the determinant is always positive unless the supporting lines l_i, for all $i \in I$, are parallel. We assume that this is not the case. (In the next chapter, we will improve upon this algorithm and a side effect is that this assumption is always enforced.) Since we can assume some rotation of the figure so that both a_i and b_i are nonzero for all i, we have $(\sum_i a_i^2) > 0$ and $(\sum_i b_i^2) > 0$. These two conditions together with the positive determinant implies that the point (x, y) found indeed minimizes the quadric error [33]. Incidentally, $(m_{11}m_{22} - m_{12}^2)$ is also the determinant of the system of equations representing $\partial\Pi/\partial x = 0$ and $\partial\Pi/\partial y = 0$. Therefore, we conclude that this system is solvable with a unique solution.

4.2 Repeated edge contraction

The above completes the description of contracting one edge. How do we proceed afterwards? In general, let S_u be a multiset of lines associated with each vertex u in the current simplified line. Initially, for each vertex v_i of L, S_{v_i} contains l_i and l_{i+1}. In general, if we contract an edge uu' to the new vertex v in the current simplified line, let $S_v = S_u \cup S_{u'}$ without removing duplicates, and fix v so as to minimize the quadric error $\sum_{l \in S_v} H(v, l)$. The corresponding edges of the lines in S_v form a subchain $C_v = v_{i-1}v_i \ldots v_k$ of L for some i and k. Note that $l_{i+1}, \ldots, l_{k-1}$ appear twice in S_v and l_i and l_k appear once in S_v. Hence, the quadric error $\sum_{l \in S_v} H(v, l)$ equals $H(v, l_i) + 2\sum_{j=i+1}^{k-1} H(v, l_j) + H(v, l_k)$.

In the implementation, it is not necessary to explicitly maintain the multisets S_v because this information is captured by the vertex matrix M_v. Contracting an edge uu' is carried out by three simple steps. First, add M_u and $M_{u'}$ to form M_v(which corresponds to $S_v = S_u \cup S_{u'}$). Second, replace the last row of M_v by (0,0,1) to form a matrix M_v^*. Third, determine the x and y coordinates of v by the equation $(x, y, 1)^t = (M_v^*)^{-1}(0, 0, 1)^t$. Note that $(x, y, 1)M_v(x, y, 1)^t$ equals the quadric error $\sum_{l \in S_v} H(v, l)$.

We can use the same argument in last section to verify that $(M_v^*)^{-1}$ exists and the v determined indeed minimizes the quadric error, provided that not all the lines in S_v are parallel.

4.3 A basic algorithm

Repeated application of the edge contraction above yields a basic algorithm, which is the 2-dimensional version of Garland and Heckbert's algorithm proposed in [1]. We denote this algorithm by Q.

We use linked lists to represent lists of the vertices and edges of L, and use a heap to store the quadric error of each possible edge contraction for current simplified polygonal line. We sketch the algorithm Q below.

Algorithm *Q*

Input. A polyline L, and an error tolerance ϵ.

Output. A simplified polyline $L = v_0 v_1 \ldots v_m$.

```
1.  Compute $M_{v_i} = K_{l_i} + K_{l_i+1}$ for all interior vertices $v_i$ of $L$.
2.  For each interior edge $v_i v_{i+1}$, set $M_v = M_{v_i} + M_{v_{i+1}}$, and solve $M_v^*(x,y,1)^t = (0,0,1)^t$ for
    $v = (x,y)$, and compute the quadric error $(x,y,1)M_v(x,y,1)^t$.
3.  Insert all the edges into a min-heap with their quadric errors as the key.
4.  while the minimum quadric error in the heap is not larger than $\epsilon^2$,
5.          Remove from the heap the edge $uu'$ with minimum quadric error.
6.          Contract $uu'$ to $v$.
7.          Update the adjacency information of $v$.
8.          Update the quadric errors(in the heap) of the two edges incident to $v$.
```

Line 4 of the above algorithm can be replaced by

```
4.  while size of the current simplified polyline > the input target size,
```

if the input parameter is the target size instead of the error tolerance ϵ.

Clearly, each edge contraction takes $O(\log m)$ time where m is the size of the current simplified polyline.

Chapter 5

Our Improved Algorithms

5.1 Deficiencies

The basic algorithm stated in the last chapter suffers from the problem that the quadric error used does not guarantee a bound on the distance of a new vertex v from the original polygonal line. We describe two problematic scenarios below.

Figure 5.1 shows a spiral $L = v_0 \dots v_n v_{n+1} \dots v_{2n+1}$ in which the edges $v_{i-1}v_i$ for $1 \le i \le n$, and $v_{i-1}v_i$ for $n+2 \le i \le 2n+1$ lie on two lines l_1 and l_2 respectively. Let l_3 be the line through $v_n v_{n+1}$. Let l be the line through v_{n+1} perpendicular to l_1. Suppose that we first contract $v_n v_{n+1}$ and then repeatedly contract edges without destroying v_0 or v_{2n+1}. Then we will eventually obtain a simplified line $v_0 v v_{2n+1}$. We claim that the vertex v can lie very far to the right while the maximum quadric error of any edge contraction performed is very small.

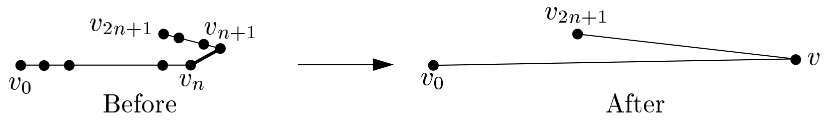

Figure 5.1: Contract $v_0 \dots v_{2n+1}$ down to $v_0 v v_{2n+1}$.

By Lemma 6.1 in the next chapter, v must lie in the triangle bounded by l_1, l_2, and l_3. We arrange the three lines such that this triangle is isosceles. See Figure 5.2. Let d_i be the distance of v from l_i. Then the quadric error of producing v is $(2n-1)d_1^2 + (2n-1)d_2^2 + 2d_3^2$. At the optimal location for v, d_1 and d_2 should be balanced. Let l_3' be the line that is parallel to, at distance d_3 from, and below l_3. For any fixed d_3, the optimal location of v is the intersection

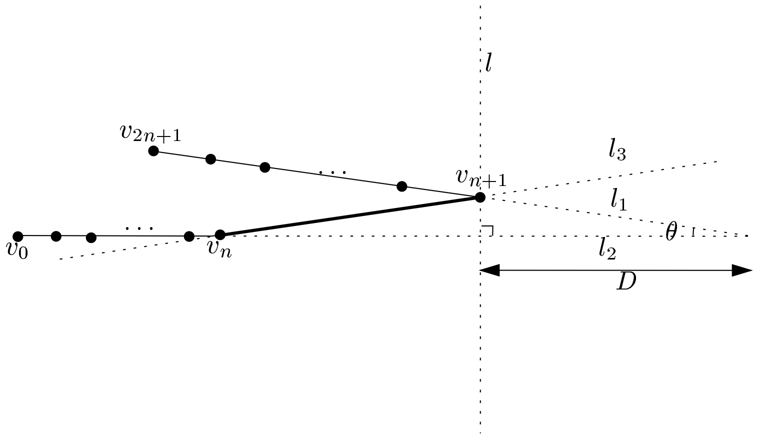

Figure 5.2:

between l'_3 and the bisector of angle θ.

If we increase n at this fixed d_3, then the quadric error will increase. To reduce the quadric error again, we have to decrease d_1 and d_2 which can only be achieved by increasing d_3. Thus, by increasing n, we can push v arbitrarily close to the right vertex of the isosceles triangle. Let D be the half of the length of bottom of the triangle. So we can make the directed Hausdorff distance from v to L arbitrarily close to D. Let θ be the right angle of the isosceles triangle. Then $d_3 \leq 2D \sin\theta$. The quadric error of the right vertex of the triangle is certainly an upper bound on the minimum quadric error of producing v since v lies inside the triangle by Lemma 6.1 in the next chapter. The former quadric error is $8D^2 \sin^2\theta$. Hence, by decreasing θ, we can make the quadric error of producing v small compared with D. The quadric error of other edge contractions prior to the one producing v can only be smaller. This completes the example.

Another problematic scenario is that when we contract a inflection edge that is almost parallel to the two adjacent edges, the new vertex produced can also be far from the original line. Figure 5.3 shows a polygonal line $v_0 \ldots v_n v_{n+1} \ldots v_{2n+1}$ in which $v_n v_{n+1}$ is a inflection edge. Note that the three lines containing the edges of the original polygonal line are arranged in the same fashion as in Figure 5.2. Thus, when we repeatedly contract down to a polygonal line $v_0 v v_{2n+1}$, the vertex v will also be very far from the original polygonal line while the maximum quadric error of edge contractions performed is small.

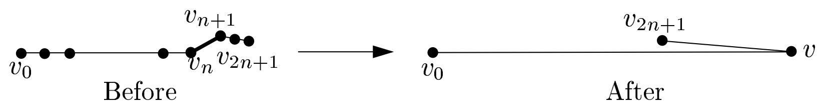

Figure 5.3: Contract $v_0 \ldots v_{2n+1}$ down to $v_0 v v_{2n+1}$.

5.2 Revised quadric error and its computation

We suggest methods for handling the two bad scenarios in the previous section. In the next chapter, we will prove that they are sufficient to translate the quadric error to a bound on the directed Hausdorff distance from the simplified line to the original line.

The basic idea is that we need not restrict ourselves to supporting lines of the edges of L for quadric error computation. Thus, our approach is to add extra lines. For each interior vertex v_i of L such that the magnitude of the turning angle at v_i is more than $\pi/2$, we add a line l that locally touches L at v_i and makes the same acute angle with $v_{i-1}v_i$ and v_iv_{i+1}. See Figure 5.4. Doing so will constraint the corresponding contracted vertex to be within some distance around v_i, which we will prove it rigorously in Chapter 6. Correspondingly, $S_{v_i} = \{l_i, l_{i+1}, l\}$. The vertex matrix M_{v_i} of v_i is set to be $K_{l_i} + K_{l_{i+1}} + K_l$ (instead of $K_{l_i} + K_{l_{i+1}}$), where K_{l_i}, $K_{l_{i+1}}$, and K_l are the line matrices of l_i, l_{i+1}, and l respectively. As in the previous chapter, whenever we contract two edges uu' in the current simplified line to produce a new vertex v, the vertex matrix of v M_v is set to be $M_u + M_{u'}$ and S_v equals $S_u \cup S_{u'}$ without removing duplicates. The quadric error of contracting uu' is again $(x, y, 1)M_v(x, y, 1)^t$, where $v = (x, y)$.

Suppose that the corresponding edges of the line in S_v form the subchain $C = v_{i-1} \dots v_k$. Then the quadric error can be rewritten as of the last edge contraction producing v is $H(v, l_i) + 2\sum_{j=i+1}^{k-1} H(v, l_j) + H(v, l_k) + \sum_{l \in E} H(v, l)$, where E is the set of extra lines added at sharp turns at interior vertices of C. As a shorthand, we denote this sum by $Q(v, C)$.

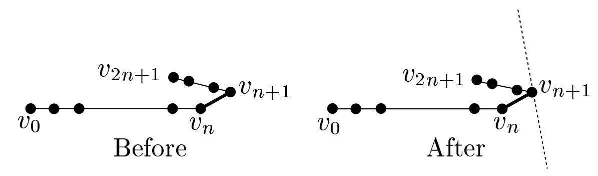

Figure 5.4: Add an extra line to the sharp corner at v_{n+1}.

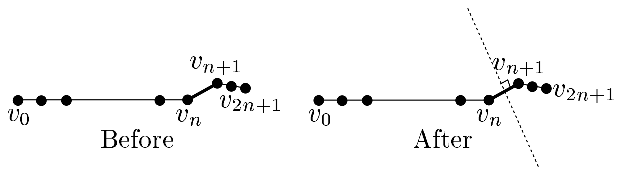

Figure 5.5: Add a complementary line for the inflection edge v_nv_{n+1}.

We also add extra lines for inflection edges or edges that are adjacent to two collinear edges. For each such edge e, we add a line l that passes through the midpoint of e and perpendicular to e. We call l the *complementary line* of e. See Figure 5.5. The effect of doing this will again constraint the corresponding contracted vertex to be within some distance around the midpoint of e, which we will prove it rigorously in Chapter 6. Unlike the extra lines added at sharp turns, complementary lines are much less natural. Suppose that we are producing a new vertex v that represents a subchain C_v of the original polygonal line. We will see in the next chapter that for our bound on the directed Hausdorff distance to hold, complementary lines of edges in C_v are only needed when C_v does not have any sharp turn at interior vertices. Also, in this case, we only need one complementary line for the proof to work. Therefore, during edge contraction, we should eliminate complementary lines to avoid increasing the quadric error of future edge contractions unnecessarily (recall that the quadric error will bound the squared directed Hausdorff distance from the simplified line to the original one).

To cater for complementary line, we keep an auxiliary vertex matrix M'_v in addition to M_v for each vertex v of the current simplified line. Initially, M'_{v_i} is the zero matrix for each vertex of the original polygonal line. In general, let C_v be the subchain of the original polygonal line that v represents. If an extra line has been added to a sharp turn at an interior vertex of C_v, then M'_v is set to be the zero matrix; otherwise, M'_v is the line matrix of the complementary line of some inflection edge in C_v. How do we select this inflection edge? This is done during edge contraction.

Suppose that we contract an edge uu' to a new vertex v. Let K be the line matrix of the complementary line of uu' if uu' is a inflection edge, and let K be the zero matrix otherwise. If an extra line has been added to a sharp turn at an interior vertex of $C_u \cup C_{u'}$, then we set M'_v to be the zero matrix. Otherwise, if at most one of K, M'_u and $M'_{u'}$ is not the zero matrix, then we set M'_v to be it. Finally, if at least two of K, M'_u and $M'_{u'}$ are not the zero matrix, then we set M'_v to be one of them. That is, we keep the complementary line of at most one inflection edge in $C_u \cup C_{u'}$. After obtaining M'_v, the quadric error for contracting uu' to produce v is $(x, y, 1)(M_v + M'_v)(x, y, 1)^t$. If we replace the last row of $M_v + M'_v$ by $(0, 0, 1)$ to form a matrix M^*_v, then solving $M^*_v(x, y, 1)^t = (0, 0, 1)^t$ yields the location of v. Clearly, one obtains different edge contraction procedures by using a different selection criteria when at least two of K, M'_u and $M'_{u'}$ are not the zero matrix. We describe several algorithms resulted in the next section.

In all, if v represents $C_v = v_{i-1} \dots v_k$, then the quadric error of the edge contraction producing v has been revised to be the sum of two terms $Q(v, C_v) + \Delta$, where

- $Q(v, C_v) = H(v, l_i) + 2\sum_{j=i+1}^{k-1} H(v, l_j) + H(v, l_k) + \sum_{l \in E} H(v, l)$, where E is the set of extra lines added at sharp turns at interior vertices of C_v, and
- $\Delta = 0$ if no complementary line is kept with C_v, or $\Delta = H(v, l)$ if l is the complementary line kept with C_v.

$Q(v, C_v) + \Delta$ is the revised quadric error used by our simplification algorithm and it is computed as $(x, y, 1)(M_v + M'_v)(x, y, 1)^t$ where $v = (x, y)$.

5.3 Three improved algorithms

First of all, our proof of the bound on the directed Hausdorff distance in the next chapter holds independent of the edge contraction procedure chosen. In the experimentation, we implemented two edge contraction procedures. In one procedure, we greedily select the nonzero matrix among K, M'_u and $M'_{u'}$ depending on which yields the smallest quadric error for contracting uu'. In the other procedure, we prefer M'_u first, then $M'_{u'}$, and then K, assuming that u appears before u' in the representation of the current simplified line.
Finally, one still has freedom in selecting the next edge to be contracted. We experimented with two methods. The first method is greedy in nature: choose the next edge that yields the smallest quadric error. Since there are two edge contraction procedures as described in the previous paragraphs, and each may produce a different quadric error for contracting the same edge, the greedy method actually yields two simplification algorithms. In both of these two algorithms, all quadric errors for contracting an edge of the current simplified polygonal line are stored in a minimum heap. After each edge contraction, we have to update(in the heap) the quadric errors of two edges incident to the vertex produced by the contraction. Outlines of these two algorithms are similar to the basic algorithm presented in last chapter. So the time needed for each edge contraction for both of these two versions is $O(\log m)$ where m is the size of the current simplified polyline. We use QGG to denote the algorithm that picks the next edge greedily and sets M'_v to obtain the smallest quadric error. We use QG to denote the algorithm that picks the next edge greedily and sets M'_v according to the order of preference M'_u, $M'_{u'}$, and then K.
In the second edge selection method, we always choose the leftmost edge (according to the representation of the current simplified line) subject to the constraint the resulting quadric

error is below some prescribed threshold. The quadric error computation is based on setting M_v' to obtain the smallest quadric error. A heap is no longer needed and we just need to keep track of the leftmost edge with quadric error without exceeding the threshold. each edge contraction now takes constant time. We denote this algorithm by QLG.

Chapter 6

Error Bound Analysis

Let L_s be a simplified polyline obtained from a polyline L by a sequence of edge contractions of any of our improved algorithms. In this chapter, we will show that the directed Hausdorff distance from L_s to L is bounded by the square root of the maximum quadric error of the edge contractions performed. We prove our result in two steps. First, we prove that for each vertex on L_s, it is bounded by that distance from L. Then we prove the same for each edge of L_s.

6.1 Bound for new vertices

We first prove two technical lemmas.

Lemma 6.1 *Consider the arrangement of $m \geq 2$ lines, $l_1, \ldots, l_m$, such that not all of them are parallel. Let S_2 be the union of all bounded cells, S_1 the union of all bounded edges, and S_0 the set of all vertices. If v is a point that minimizes $\sum_{i=1}^{m} c_i H(v, l_i)$ for any positive integers c_i, then $v \in S_0 \cup S_1 \cup S_2$.*

Proof: Assume to the contrary that $v \notin S_0 \cup S_1 \cup S_2$. Let v lie in the interior of an unbounded cell U or on the two semi-infinite edges of U. See Figure 6.1.

Let the chain of boundary edges of U be $e_1, e_2, \ldots e_t$, where e_1 and e_t are semi-infinite. Let x be the intersection point of the supporting lines of e_1 and e_t. If e_1 and e_t are parallel, then their supporting lines intersect at two points at infinity and let x be the one that lie outside U at infinity.

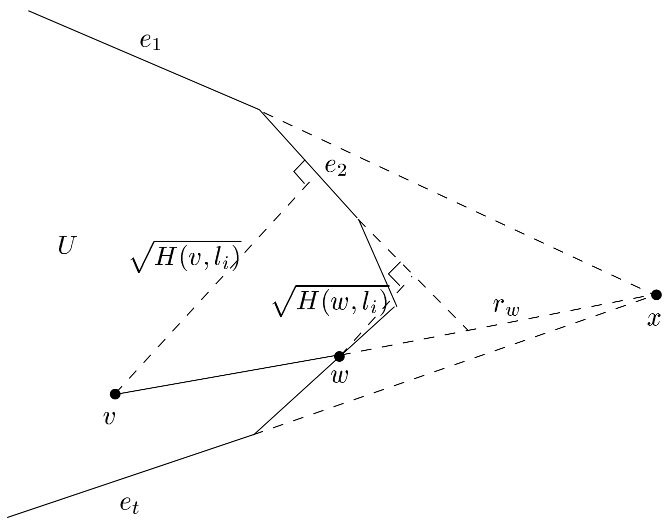

Figure 6.1:

The ray r_v emanating from v through x intersects the boundary of U, say at a point w. Let r_w be the ray emanating from w through x. For each line l_i that does not contain v, since l_i avoids the interior of U, l_i either intersects r_w or is parallel to r_w. Thus, $H(v,l_i) \geq H(w,l_i)$. Moreover, by assumption, not all the lines are parallel and so there exists l_k such that l_k intersects r_w and so $H(v,l_k) > H(w,l_k)$. This contradicts the given condition that v minimizes $\sum_{i=1}^{m} c_i H(v,l_i)$. $\square$

Lemma 6.2 *Let x be the intersection of two lines l_1 and l_2. Let $-_1$ and $-_2$ be the perpendiculars to l_1 and l_2 at x respectively. If the smaller angle between $-_1$ and $-_2$ is acute, then $|vx|^2 \leq H(v,l_1) + H(v,l_2)$ for any point v inside the angle.*

Proof: Let vf_i be the perpendicular from v to l_i. See Figure 6.2. Extend the line segment vf_2 to intersect l_1 at f_2'. Since $\angle vxf_1 = \angle xvf_2 + \angle vf_2'x$, $\angle vxf_1 \geq \angle xvf_2$. Since $d_1 = |vx|\sin(\angle vxf_1)$ and $|xf_2| = |vx|\sin(\angle xvf_2)$, $d_1 \geq |xf_2|$. Hence, $d_1^2 + d_2^2 \geq |xf_2|^2 + d_2^2 = |vx|^2$. $\square$

Lemma 6.3 *If l_1, l_2, and l are three concurrent lines such that l stabs the smaller (break ties arbitrarily) double wedge formed by l_1 and l_2, then $H(v,l) \leq H(v,l_1) + H(v,l_2)$ for any arbitrary point v.*

Proof: Let $d_i = \sqrt{H(v,l_i)}$ and $d = \sqrt{H(v,l)}$. Let x be the common intersection of l_1, l_2, and l. Let vf_i be the perpendicular from v to l_i and let vf be the perpendicular from v to

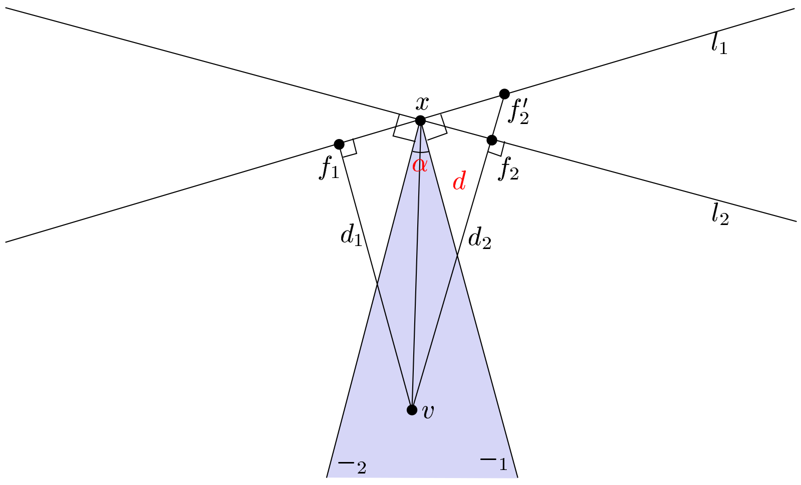

Figure 6.2:

l. Let $-_1$ and $-_2$ be the perpendiculars to l_1 and l_2 at x respectively. Suppose that v lies inside the acute angle between $-_1$ and $-_2$. It is obvious that $|vx| \geq d$. By Lemma 6.2, we have $d_1^2 + d_2^2 \geq |vx|^2 \geq d^2$. Suppose that v lies outside the acute angle between $-_1$ and $-_2$. Without loss of generality, let v be in the region as shown in Figure 6.3. It is obvious that

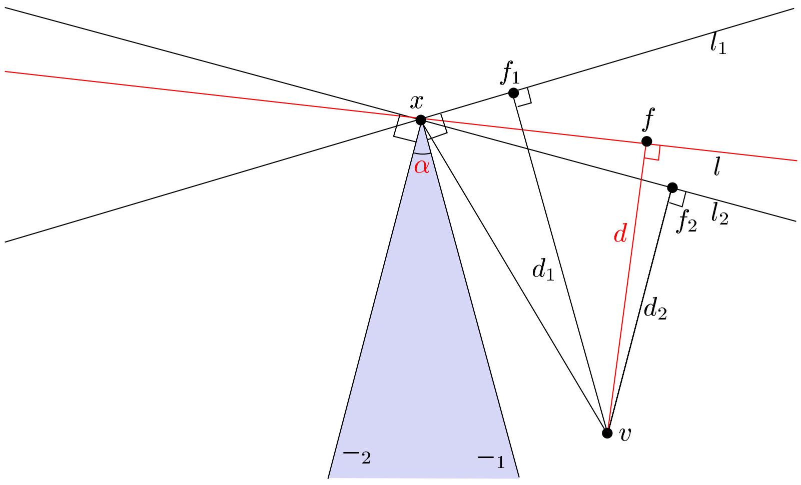

Figure 6.3:

$\angle vxf_1 \geq \angle vxf$. Since $d_1 = |vx| \sin(\angle vxf_1)$ and $d = |vx| \sin(\angle vxf)$, we have $d_1 \geq d$. □

For the following lemmas, we let L be the subchain of the original polyline associated with a vertex v of the current simplified polyline. Now if L is nice and concave/convex, we can prove that the distance of v from L is within the square root of the quadric error for contracting L

downto v.

Lemma 6.4 *Let $L = v_0 \dots v_m, m \geq 2$, be a nice concave/convex polyline. If v is a point that minimizes $Q(v, L)$, then $H(v, L) \leq Q(v, L)$.*

Proof: We consider the case where L is convex as the other case is symmetric. See Figure 6.1. Let l_i be the supporting line of the edge $v_{i-1}v_i$. Consider the arrangement of the lines l_i's. Let

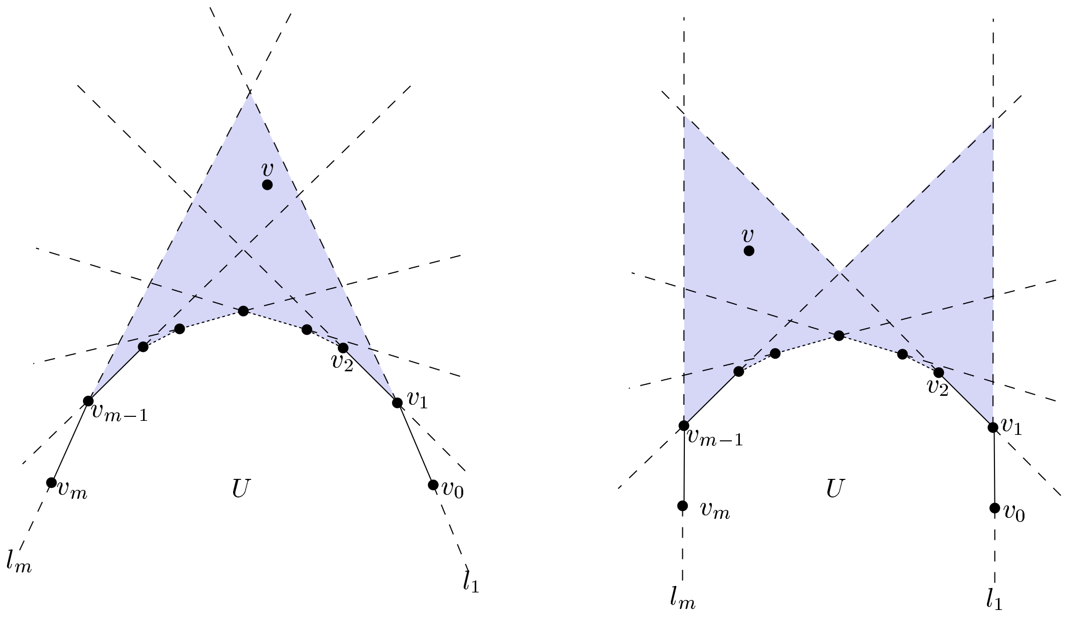

Figure 6.4:

U be the cell such that L is a subset of the boundary of U. Since L is convex, U is unbounded. We have two situations depending on whether l_1 and l_m are parallel or not. See Figure 6.1. By Lemma 6.1, v lies inside the shaded region as the other cells are all unbounded. So v must lie inside some strip or wedge induced by an edge or interior vertex of L. If v lies inside a strip, then clearly $H(v, L) \leq H(v, l_i)$ for some i which is less than $Q(v, L)$. If v lies inside a wedge, then since L is nice, then angle of the wedge is acute and so by Lemma 6.2, $H(v, L) \leq Q(v, L)$. ⊔⊓

Now we prove that the same result holds when L is a nice spiral.

Lemma 6.5 *Let $L = v_0 \dots v_m, m \geq 2$, be a nice spiral. If v is a vertex that minimizes $Q(v, L)$, then $H(v, L) \leq Q(v, L)$.*

Proof: Rotate the figure so that v_0 and v_m are below v and they lie on opposite sides of a vertical line ℓ through v. Without loss of generality, we assume that v_0 and v_m lie on the left

and right of ℓ respectively. We call a vertex v_i on L a local maximum if v_i is not an endpoint of L and v_i is not below its two neighboring vertices. Local minimum is defined symmetrically.
Case 1: there is a concave subchain C of L that contains a local maximum and intersects ℓ above/below v.

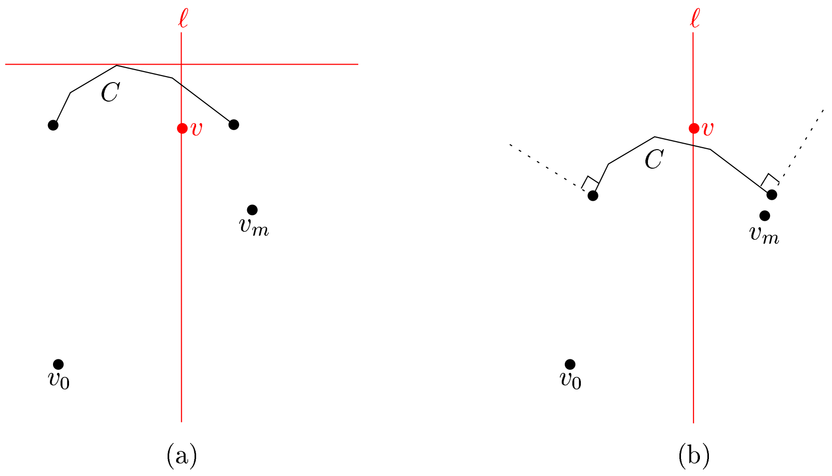

Figure 6.5: Case 1.

Suppose that C intersects ℓ above v. See Figure 6.5(a). Let ℓ' be the horizontal line tangential to the local maximum in C. The perpendicular from v to ℓ' must then intersect C. By Lemma 6.3, the length of this perpendicular is at most $\sqrt{Q(v,L)}$ and so is the distance of v from C. Suppose that C intersects ℓ below v. See Figure 6.5 (b). Then v lies inside some strip or wedge induced by edges or interior vertices of C and so $H(v,L) \leq Q(v,L)$.
Case 2: there is a maximal concave subchain C of L such that C contains a local maximum, and v_m (v_0) is not the left(right) endpoint of C if C lies on the right (left) of ℓ. See Figure 6.6. We can assume that case 1 does not apply and so C lies on one side of ℓ, say the right side. See Figure 6.6(a). Let v_k be the left endpoint of C. Since $v_k \neq v_m$ by assumption, v_k is incident to another edge e not in C. Since C is maximal and L is a spiral, e must lie on the right of v_k below the edge in C incident to v_k. If v lies inside the strip induced by e or some edge in C, or v lies inside the wedge induced by v_k or some interior vertex of C, then we are done. If this is not the case, then observe that both endpoints of e must lie above v and the strip induced by e intersects l above v. Let C' be the maximal convex subchain that contains e. Then either C' contains a local minimum or C' contains v_m which lies below v. See Figure 6.7. In either case, v must lie inside some strip or wedge induced by edges or interior vertices in C'. Hence, $H(v,L) \leq Q(v,L)$.

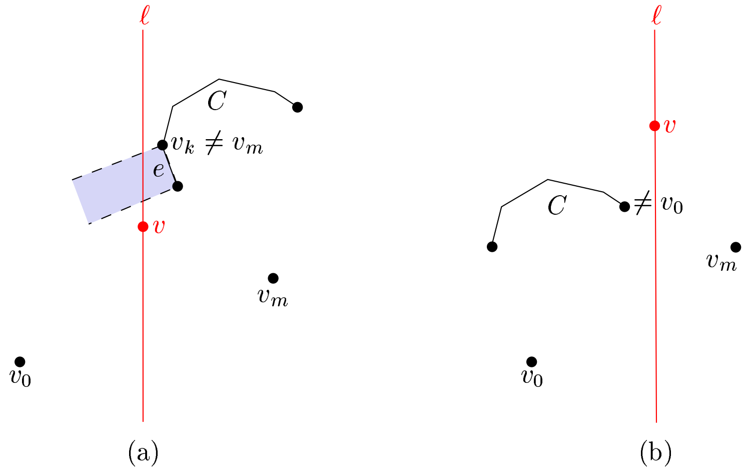

Figure 6.6: Case 2 (a) & (b).

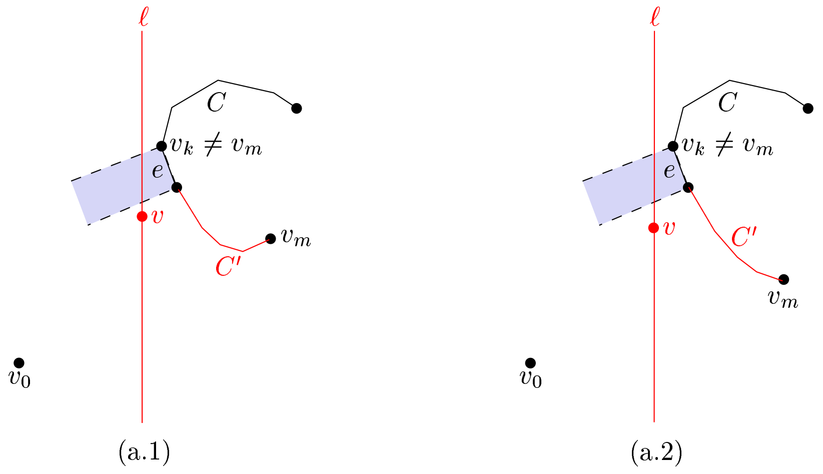

Figure 6.7: Case 2 (a.1) & (a.2).

Case 3: there is a local minimum in L. Let C' be the maximal convex subchain that contains this local minimum. Suppose that C' intersects ℓ. If C' intersects ℓ below v (See Figure 6.8(a)), then by Lemma 6.3, the distance of v from the horizontal tangential line touching the local minimum in C' is bounded by $\sqrt{Q(v, L)}$. So is the distance of v from C' then. If C' intersects ℓ above v (See Figure 6.6(b)), then v must lie inside some strip or wedge induced by edges or interior vertices in C' and so $H(v, L) \leq Q(v, L)$. Suppose that C' does not intersect ℓ. Without loss of generality, assume that C' lies on the left of ℓ. Let C be the maximal concave subchain shares an endpoint with C' and separates C' from v_m. If the common endpoint of C' and C is the right endpoint of C, then it is clearly not v_0 as v_0 has degree one in L. Otherwise, since C separates C' and v_m, the right endpoint of C cannot be v_0. If C contains a local maximum

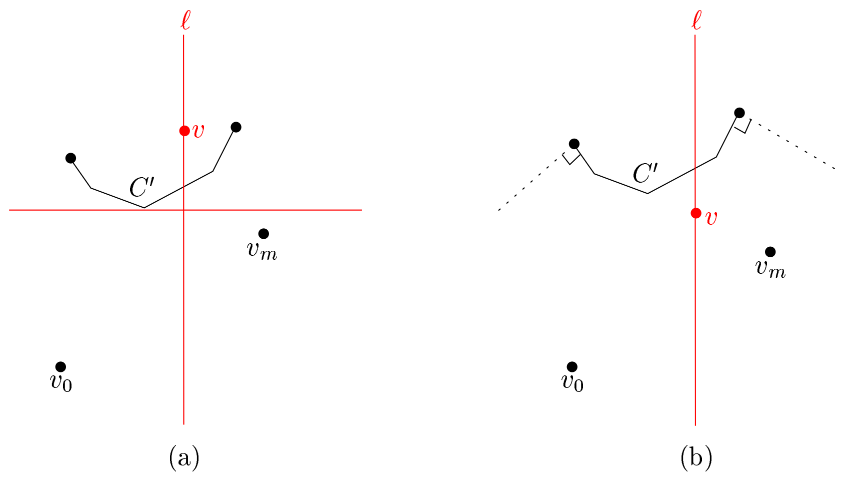

Figure 6.8: Case 3 (a) & (b).

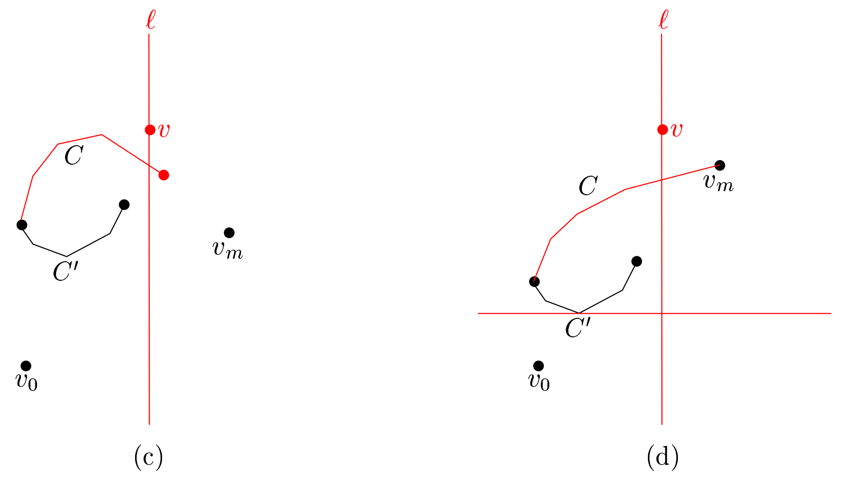

Figure 6.9: Case 3 (c) & (d).

(See Figure 6.9(c)), then C satisfies cases 1 or 2 and we are done. If C does not contain a local maximum (See Figure 6.9(d)), then since L is a nice spiral, C must be incident to the left endpoint of C' and v_m, increasing from left to right, and cross ℓ below v and above the local minimum in C'. Hence, the vertical distance from v to C is less than the distance from v to the horizontal tangential line touching the local minimum in C'. The latter is bounded by $\sqrt{Q(v,L)}$ by lemma 6.3 and so $H(v,L) \leq Q(v,L)$.

Case 4: The remaining possibility is that L does not contain any local minimum. If L does not contain any local maximum too, then L is actually a convex/concave chain with respect to horizontal direction (See Figure 6.10(a)) and Lemma 6.4 applies. Suppose that L contains a local maximum. Let C be the maximal concave subchain containing this local maximum. Since

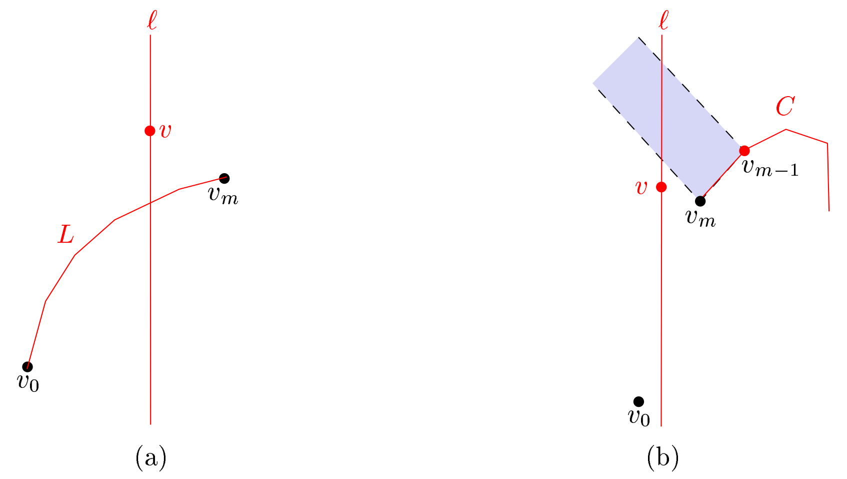

Figure 6.10: Case 4 (a) & (b).

cases 1 and 2 do not apply, without loss of generality, we can assume that C lies on the right of ℓ and v_m is the left endpoint of C. See Figure 6.10(b). So $v_{m-1}v_m$ is an edge in C. We can also assume that v lies below the intersection between ℓ and the strip induced by $v_{m-1}v_m$, otherwise v must lie inside some strip or wedge induced by edges or interior vertices in C and

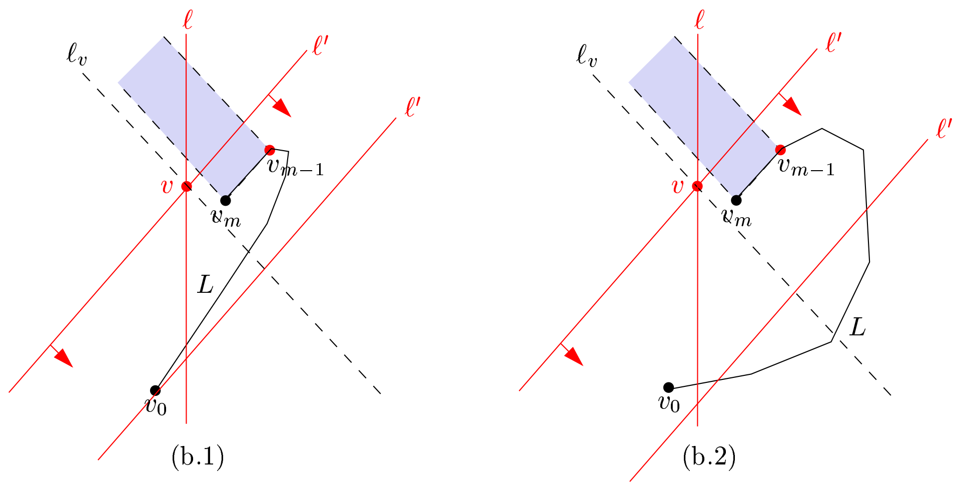

Figure 6.11: Case 4 (b.1) & (b.2).

we are done. Let ℓ_v be the line through v orthogonal to $v_{m-1}v_m$. Note that v_0 and v_m lies on the left and right of ℓ_v respectively. Sweep a line ℓ' parallel to $v_{m-1}v_m$ through v downward. If $\ell' \cap L$ is always connected throughout the sweeping (See Figure 6.11(b.1)), then L must be a convex chain with respect to the leftward orientation of ℓ'. Thus, Lemma 6.4 applies. If $\ell' \cap L$ ever becomes disconnected during the sweeping, (See Figure 6.11(b.2)), then L contains a local minimum with respect to the upward orientation of ℓ_v. Thus, we can rotate the whole figure so that ℓ_v becomes vertical and apply case 3 above. $\square$

Now we can prove a more general statement. That is for a general polyline L, we can prove that the distance of v from L is within the square root of the quadric error for contracting L downto v after performing a sequence of edge contractions.

Lemma 6.6 *Let $L = v_0v_1 \dots v_m, m \geq 2$, be a polygonal line. If v is a vertex that minimizes $Q(v, L) + \Delta$, where $\Delta = 0$ if no complementary line is associated with L, or $\Delta = H(v, l)$ if l is the complementary line associated with L. Then $H(v, L) \leq Q(v, L) + \Delta$.*

Proof: Case 1: If L is not nice, then we have an extra line l added at a sharp turn at vertex v_i and $\Delta = 0$. Let l_1 and l_2 be the supporting lines of the two incident edges of v_i. We denote l by l_3 for uniformity. Then the six angles formed by l_1, l_2, and l_3 are at most $\pi/2$. Denote them by θ_i, $i = 1, 2, \dots, 6$. Let $d_i = \sqrt{H(v, l_i)}$. Assume that v lies inside angle θ_2. See Figure 6.12(a). Let f_1 and f_2 be the feet of the perpendiculars from v to l_1 and l_2 respectively. Without loss

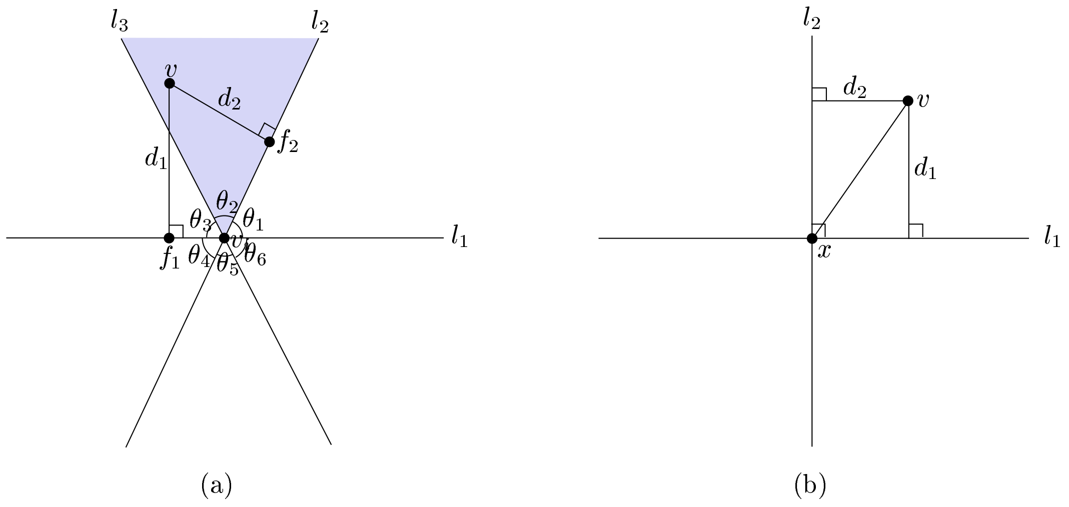

Figure 6.12: (a) A star for Case 2; and (b) a cross for Case 3.

of generality, suppose $f_1 = v_i$ or f_1 is on the left of v_i. Since $\theta_2 \leq \pi/2$, f_2 must be above or on line l_1. By the same reasoning as in the proof of Lemma 6.2, $|vv_i|^2 \leq d_1^2 + d_2^2 \leq Q(v, L)$.

Case 2: If L is nice but not a spiral, then $\Delta = H(v, l)$ where l is the associated complementary line of some edge v_iv_{i+1}. Let l_1 be the supporting line of v_iv_{i+1}. And we denote l by l_2 for uniformity. Note that $l_1 \cap l_2$ is the midpoint of v_iv_{i+1}, say x. Let $d_i = \sqrt{H(v, l_i)}$. Then by Pythagoras' Theorem, $|vx|^2 = d_1^2 + d_2^2$. Since $d_1^2 \leq Q(v, L)$ and $d_2^2 = H(v, l)$, $|vx|^2 \leq$

$Q(v, L) + H(v, L) = Q(v, L) + \Delta$.

Case 3: If L is a nice spiral, then by Lemma 6.5, $H(v, L) \leq Q(v, L)$. □

Up to here, we have proved a bound on the distance of v from the original polygonal line L. We summarize as the following theorem.

Theorem 6.1 *Let L be a polygonal line. Let L_s be the simplified polygonal line after performing a sequence of edge contractions. Then for any vertex v on L_s, $H(v, L)$ is at most the maximum quadric error of the edge contractions performed.*

Proof: By Lemma 6.6, $H(v, L_v) \leq Q(v, L_v) + \Delta$, which is quadric error of the edge contraction producing v. □

6.2 Bound for new edges

We now prove a bound on the directed Hausdorff distance from each edge of L_s to the original polygonal line L.

We first prove two technical lemmas.

Lemma 6.7 *Let $L = v_0 v_1 \dots v_m$, $m \geq 1$, be a polygonal line. Let l be a line and let v'_0, v'_m, and L_l be the orthogonal projections of v_0, v_m and L onto l respectively. Suppose that L avoids l except possibly at v_0 and v_m. Then for all point x in L_l, $H(x, L) \leq \max(Q(y, L), |v_0 v'_0|^2, |v_m v'_m|^2)$ for any point y on l. Note that $v'_0 v'_m \subseteq L_l$.*

Proof: Since L avoids l, L is on one side of l. Let v_i be a vertex of L with maximum distance from line l. Through v_i, draw a line l' parallel to l. Let h be the distance between parallel lines l and l'.

Case 1: v_i is neither v_0 nor v_m. See Figure 6.13.

If the turn at v_i is not sharp, then l' stabs the smaller double wedge formed by the supporting lines of edges incident to v_i. So by Lemma 6.3, $h^2 \leq Q(y, L)$. If the turn at v_i is harp, then an extra line l'' is added at v_i. So l' will stab the smaller double wedge formed by l'' and the supporting line of an edge incident to v_i. Hence, we again conclude that $h^2 \leq Q(y, L)$ by Lemma 6.3.

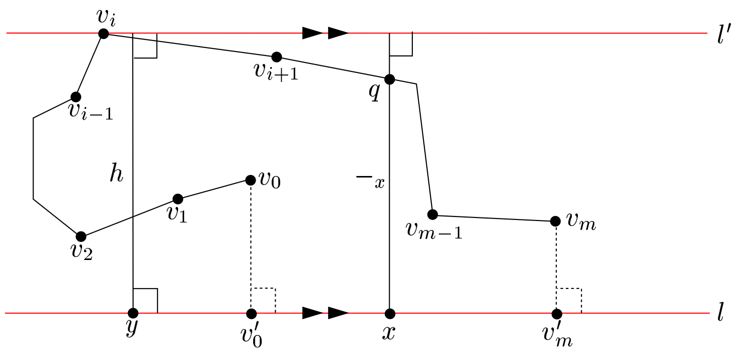

Figure 6.13: Case 1: v_i is neither v_0 nor v_m.

Draw the perpendicular $-_x$ from x to l'. Since $x \in L_l$, $-_x$ must intersect L at a point, say q, before reaching line l'. Thus, we have $|qx|^2 \le h^2 \le Q(y, L)$.

Case 2: v_i is v_0 or v_m. See Figure 6.14. Without loss of generality, assume that $v_i = v_m$. In

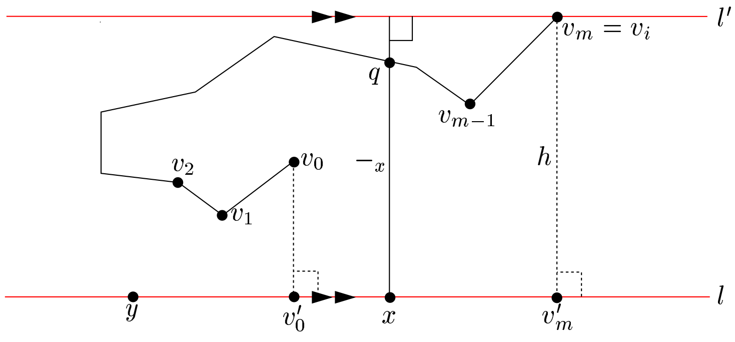

Figure 6.14: Case 2: v_i is v_0 or v_m.

this case, $h = |v_m v'_m|$ and $|qx|^2 \le h^2 = |v_m v'_m|^2$.

Therefore $H(x, L) \le \max(Q(y, L), |v_0 v'_0|^2, |v_m v'_m|^2)$. ◻

Note that in Lemma 6.7, $Q(y, L)$ is undefined when L has only one edge. For this case, $H(x, L) \le \max(|v_0 v'_0|^2, |v_m v'_m|^2)$.

We can generalize the previous lemma to the case when l does not avoid L.

Lemma 6.8 *Let $L = v_0 v_1 \dots v_m, m \ge 1$, be a polygonal line. Let l be any line and let v'_0, v'_m and L_l be the orthogonal projections of v_0, v_m and L on l respectively. Then for all point x on L_l, $H(x, L) \le \max(Q(y, L), |v_0 v'_0|^2, |v_m v'_m|^2)$ for any point y on l.*

Proof: Draw a line $-_x$ perpendicular to l through x. Since $x \in L_l$, $-_x$ must intersect L at some point, say q. See Figure 6.15. Let C be a maximal subchain of L such that C contains q

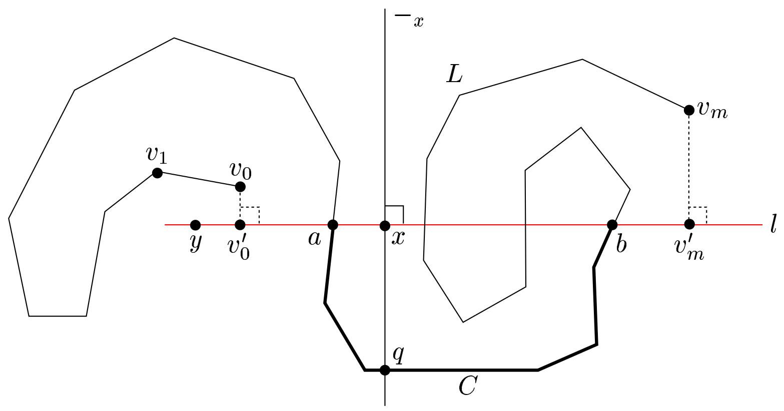

Figure 6.15: Case 2: v_i is v_0 or v_m.

and avoids l except possibly at its two endpoints. Let a and b be the endpoints of C and let a' and b' be their orthogonal projections onto l respectively. Since C contains q, x lies in the orthogonal projection of C onto l. By Lemma 6.7, we have $H(x, C) \leq \max(Q(y, C), |aa'|^2, |bb'|^2)$. Since a are possibly be v_0, v_m, or a', $|aa'|^2 = |v_0v_0'|^2$, $|v_mv_m'|^2$, or 0. Similarly we have same possible values for $|bb'|^2$. Also $H(x, L) \leq H(x, C)$ and $Q(y, C) \leq Q(y, L)$ by definition. Hence we conclude that $H(x, L) \leq \max(Q(y, L), |v_0v_0'|^2, |v_mv_m'|^2)$. $\square$

Finally, we can combine all the results above, and can prove a bound on the directed Hausdorff distance from each edge of the simplified polyline L_s to the original polyline L.

Theorem 6.2 *Let L be a polygonal line. Let L_s be the simplified polygonal line by performing a sequence of edge contractions. Then for each edge e of L_s, $H(e, L)$ is at most the maximum quadric error of the edge contractions performed.*

Proof: Let ϵ^2 be the maximum quadric error of the edge contractions performed. Let uv be an edge on the simplified polygonal line L_s. Let L_u and L_v be the subchains of L associated with u and v respectively. By Theorem 6.1, we have $H(u, L_u) \leq \epsilon^2$, and $H(v, L_v) \leq \epsilon^2$. Let p and q be two points on L_u and L_v respectively such that $|pu|^2 = H(u, L_u)$ and $|qv|^2 = H(v, L_v)$. Then we immediately have $|pu| \leq \epsilon$ and $|qv| \leq \epsilon$. Let L' be the subchain of L between p and q. Let C_u and C_v be circles centered at u and v of radius ϵ respectively. See Figure 6.16. So p and q must lie inside C_u and C_v respectively. Let l be the line through u and v. Let p_l and q_l be the orthogonal projections of p and q onto l respectively. So p_l and q_l lie inside C_u and C_v

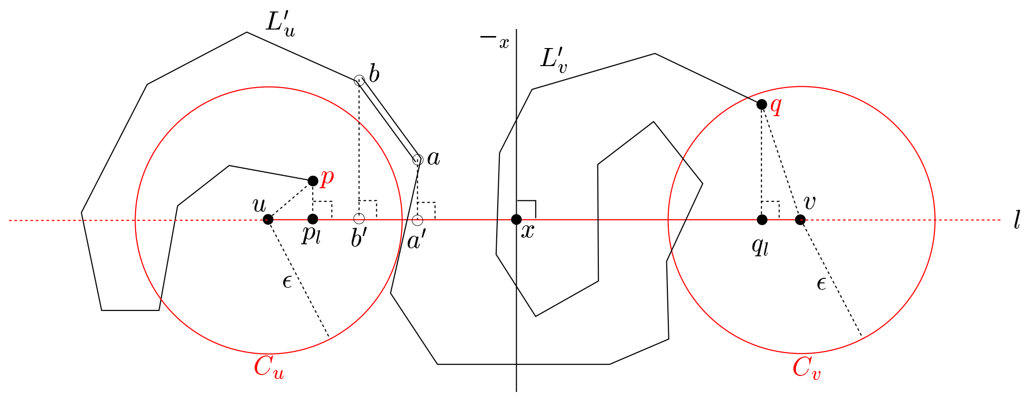

Figure 6.16:

respectively. This implies that $|px| \leq |pu| \leq \epsilon$ for all point x on $p_l u$, and $|qx| \leq |qv| \leq \epsilon$ for all point x on $q_l u$.

Now consider any point x on $p_l q_l$. Note that $L' \subseteq L_u \cup L_v$, and L_u and L_v overlap on one edge as uv is an edge on L_s. Let this overlapping edge be ab such that a is an end-point of L_u and b is an endpoint of L_v. Let a' and b' be the orthogonal projections of a and b onto l respectively. Let L'_u be the subchain between p and a, and L'_v be the sub-chain between q and b. Note that $L' = L'_u \cup L'_v$. Let $(L'_u)_l$ and $(L'_v)_l$ be the projections of L'_u and L'_v onto l respectively. Consider any x on $p_l q_l$. Draw a line $-_x$ perpendicular to l through x. Then $-_x$ must intersect L'_u or L'_v. This means x is on $(L'_u)_l$ or $(L'_v)_l$. If x is on $(L'_u)_l$, by Lemma 6.8, $H(x, L'_u) \leq \max(Q(u, L'_u), |pp_l|^2, |aa'|^2)$. Symmetrically, if x is on $(L'_v)_l$, $H(x, L'_v) \leq \max(Q(v, L'_v), |qq_l|^2, |bb'|^2)$. Combining the two results, as $H(x, L) \leq H(x, L'_u)$ and $H(x, L) \leq H(x, L'_v)$, we have $H(x, L) \leq \max(Q(u, L'_u), |pp_l|^2, |aa'|^2, Q(v, L'_v), |qq_l|^2, |bb'|^2)$.

Suppose that a is not farther than b from l, that is, $|aa'|^2 \leq |bb'|^2$. Considering the farthest vertex ($\neq a$), say v_i, on L'_u from l. If $v_i = p$, then $|bb'|^2 \leq |pp_l|^2$. Otherwise, if $v_i \neq p$, then v_i is an interior vertex on L'_u. By applying Lemma 6.3 at vertex v_i, $|bb'|^2 \leq Q(u, L'_u)$. So we have $|aa'|^2$, $|bb'|^2 \leq \max(Q(u, L'_u), |pp_l|^2)$. Symmetrically, for $|aa'|^2 > |bb'|^2$, by considering subchain L'_v, we have $|aa'|^2$, $|bb'|^2 \leq \max(Q(v, L'_v), |qq_l|^2)$.

So we have $H(x, L) \leq \max(Q(u, L'_u), |pp_l|^2, Q(v, L'_v), |qq_l|^2)$. Since both $Q(u, L'_u)$ and $Q(v, L'_v)$ are at most the max quadric error ϵ^2, and both $|pp_l|^2$ and $|qq_l|^2$ are at most ϵ^2 by our construction, we have $H(x, L) \leq \epsilon^2$. $\square$

Chapter 7

Experimentation and Results

7.1 Preliminaries

In this chapter, we present the experimental results comparing the several variants of our simplification algorithm. They are:

Q: no extra lines are added at sharp turns, no complementary lines are added, the next edge to be contracted is chosen greedily based on its quadric error.

QGG: extra lines at sharp turns and complementary lines are added, the matrix update at each edge contraction is performed greedily, the next edge to be contracted is chosen greedily. (See Section 5.3.)

QG: extra lines at sharp turns and complementary lines are added, the matrix update at each edge contraction is performed according to a prescribed order of preference, the next edge to be contracted is chosen greedily. (See Section 5.3.)

QLG: extra lines at sharp turns and complementary lines are added, the matrix update at each edge contraction is performed greedily, the leftmost edge with quadric error within ϵ^2 is to be contracted, where ϵ is the input error tolerance. (See Section 5.3.)

Note that main difference of QLG from Q, QG and QGG is that it requires an input error tolerance ϵ to work. Although Q, QG and QGG are designed for simplifying the given polygonal line down to a user specified size, one can equally specify a tolerance ϵ and then run Q, QG or QGG until the quadric error of the next edge to be contracted is larger than ϵ^2.

The implementations of Q, QG, QGG and QLG are based on the 3D surface simplification implementation by Garland and Heckbert [1]. For Q, QG and QGG, a heap is employed to support the greedy selection of the next edge to be contracted. The heap stores the edges in the current simplified line with their quadric error as key. After contracting an edge uu', the quadric error for contracting the two edges adjacent to uu' have to be updated. Thus, contracting an edge takes $O(\log m)$ time, where m is the size of the current simplified line. In contrast, QLG takes constant time for each edge contraction.

We expect QGG to produce the best approximation, followed by QG, QLG and then Q. But the processing time of QGG is expected to be higher, particularly compared to QLG. Our experimental results generally agree with these expectations, but there are also exceptions. We cannot really explain these exceptions and they may be data-dependent.

We also compare the above algorithm with the Douglas and Peucker algorithm (DP) [6]. It is reputed to perform very well in practice in terms of the quality of approximation. Given an input error tolerance ϵ, the output simplified polygonal line of algorithm DP has the property that the directed Hausdorff distance from the simplified polygonal line to the original polygonal line is within ϵ. The implementation of DP is from [32].

Both DP and our algorithms can take given error tolerance as input parameter. However, DP does not provide an interactive and direct way to increase the simplification, unlike QGG and QG. Thus, our algorithms can work for applications in different contexts than DP. We emphasize that our algorithms are not proposed as substitutes of DP but rather as a new method for doing the simplification task, which can work for other settings as well. Nevertheless, we use DP as a benchmark to check the quality of approximation obtained by our algorithms.

7.2 Test data and measurements

We use three categories of test data:

1. 30 sets of coastlines (geographical data) from [30],
2. One set of circle data (discretized as closed polygonal chains), and
3. The Dow Jones HK index 1998 [31].

The coastlines allow us to find out how our algorithms work for real data. For spiral chains, the working of DP uses only the vertices of the original polygonal line, which suggests that it may perform worse than quadric error based algorithms (which relocates new destination vertices when doing edge contractions). We take the discretized circle data as an example for this test case. The data set of Dow Jones HK index contains wild zigzags, which is a natural extreme test case, and we suspect the straightforward quadric error algorithm (Q) will perform the worst for this test case. An example in each category is shown below.

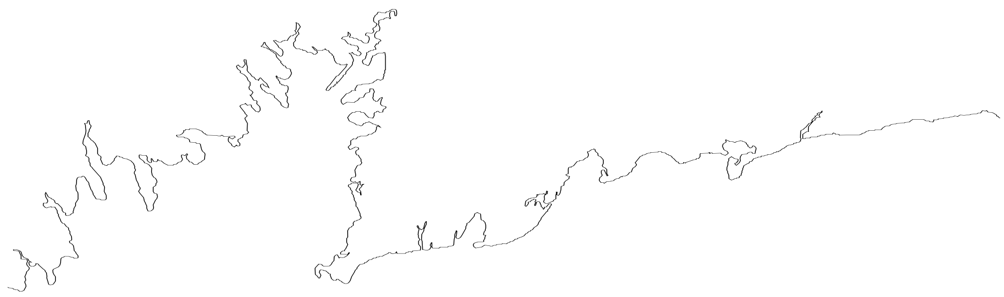

Figure 7.1: A coastline (1787 edges).

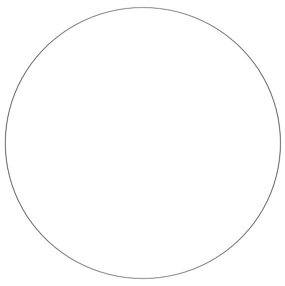

Figure 7.2: A circle (100 edges).

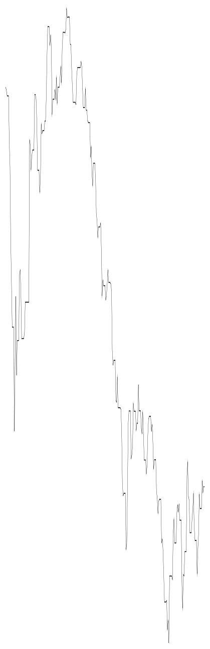

Figure 7.3: The Dow Jones HK index 1998 (274 edges).

For each category, we measure two quantities that reflect the approximation error: *mean vertex distance* (MeanVD), and *maximum vertex distance* (MaxVD). Let L be the input polygonal line and let L_s be the simplified line. MeanVD is defined to be:

$$(\sum_{v \in L} \sqrt{H(v, L_s)} + \sum_{v \in L_s} \sqrt{H(v, L)})/\text{total number of vertices in } L \text{ and } L_s$$

MaxVD is defined to be:

$$\max\{\max_{v \in L_s} \sqrt{H(v, L)}, \max_{v \in L} \sqrt{H(v, L_s)}\}$$

Note that MaxVD is to simulate/approximate the Hausdorff distance between L and L_s. In order to estimate the significance of the approximation error relative to L_s, we also measure the *mean edge length* of L_s (MeanSEL).

7.3 Summary of experimental results

We performed two classes of experiments. In class I, for each data set in each category, we plotted MeanVD, MaxVD, and MeanSEL against the percentage of edges eliminated in the simplified line for each algorithm. In class II, for each data set in each category, we plotted MeanVD and MaxVD against ϵ, the input error tolerance.

Recall that both DP and QLG require an input error tolerance to work (for reasons as described before). So for experiments in class I, we attempted by trial and error to obtain the tolerance for DP to obtain a particular size of the output simplified line. We repeat the same trial and error process for QLG.

For data sets in the same category, we obtain similar experimental results and we describe the general trend in the following.

7.3.1 Class I: Controlling the percentage of edges eliminated

The coastline

For the coastlines data, DP obtains the best MeanVD, compared to all the four quadric error algorithms. The difference among the quadric error algorithms are insignificant. See a typical plot in Figure 7.4. DP also obtains the best MaxVD compared to all the quadric error algorithms. Q and QG yield larger MaxVD than QGG and QLG over a wide range (for example, from 30% to 80% of edges eliminated in Figure 7.5). In particular, it is more common for QG than Q to produce larger MaxVD than QGG and QLG. See Figure 7.5 and figures in Appendix **??**.

Figure 7.6 shows the mean edge length of the simplified line which is similar for all the algorithms tested from 0% to 90% of edges eliminated. It can be seen that the difference in MeanVD relative to MeanSEL among Q, QG, QGG and QLG are insignificant from 0% to 90% of edges eliminated. Though DP obtains the smallest MeanVD, it is no more than 5% of MeanSEL smaller than the others.

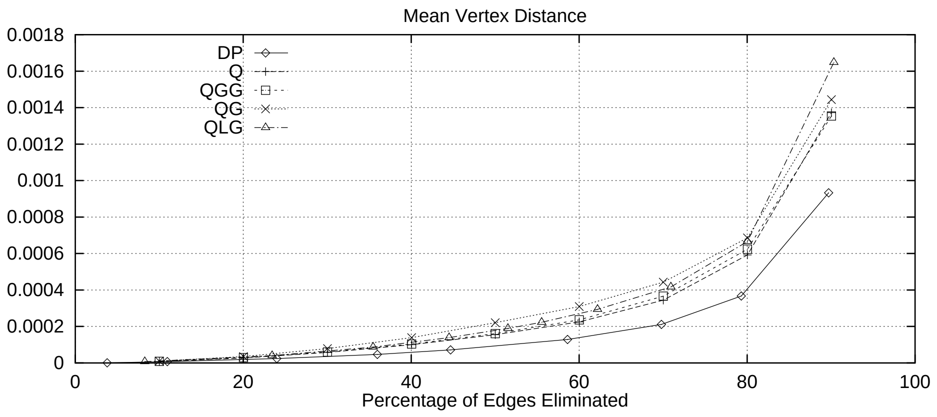

Figure 7.4: Mean Vertex Distance

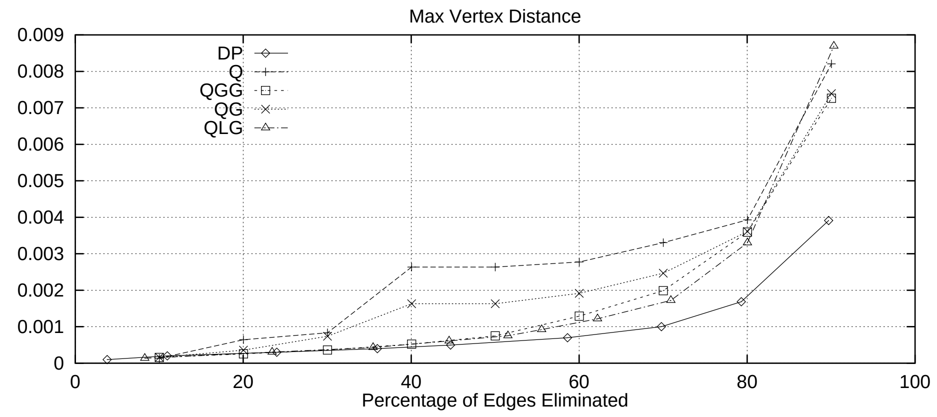

Figure 7.5: Max Vertex Distance

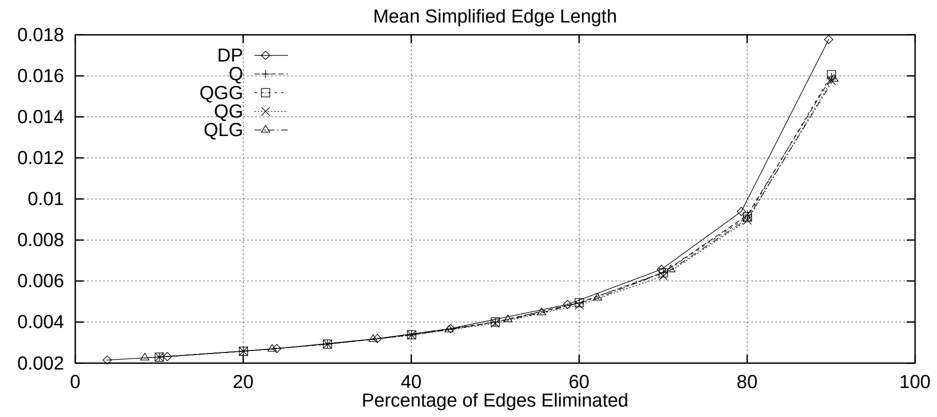

Figure 7.6: Mean Edge Length of the Simplified Polyline

The circle

For the circle data, the MeanVD of all the four quadric error algorithms are similar and slightly better than the MeanVD of DP. See a typical plot in Figure 7.7. The MaxVD of all the four quadric error algorithms are also better than the MaxVD of DP. However, Q, QGG and QG yield the same MaxVD while the MaxVD of QLG is 25% to 50% less. See a typical plot in Figure 7.8. It is surprising that QLG produces much better MaxVD than Q, QGG and QG, which we cannot explain.

Figure 7.9 shows the mean edge length in the simplified line which is similar for all the algorithms tested from 0% to about 90% of edges eliminated. It can be seen that the difference between DP and all the four quadric error algorithms in MeanVD is very negligible relative to MeanSEL before 90% of edges are eliminated, which is less than 2% of MeanSEL. For MaxVD, DP is up to 2.5% of MeanSEL larger than Q, QGG and QG, and up to 4% of MeanSEL larger than QLG. The relative difference is again not significant.

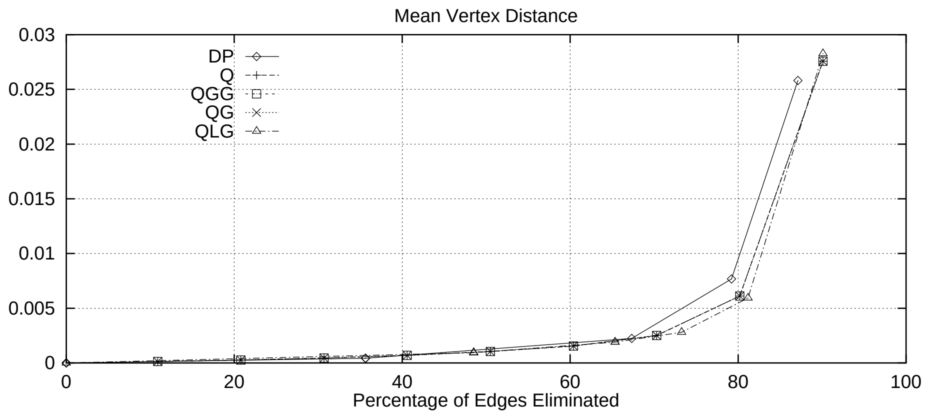

Figure 7.7: Mean Vertex Distance

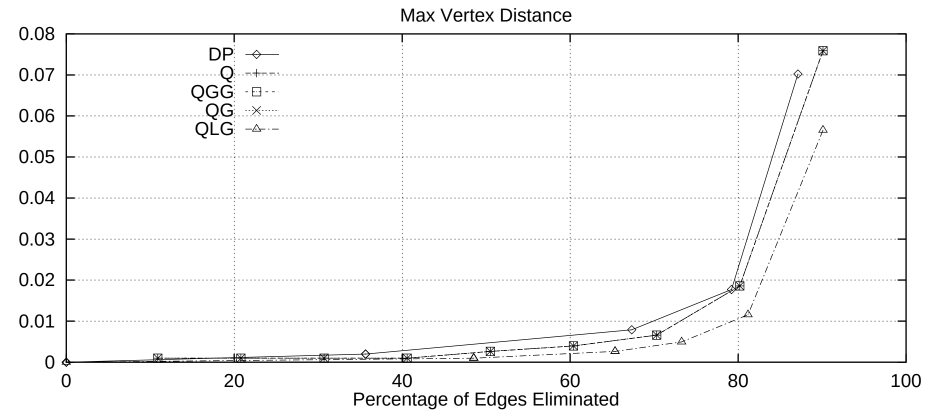

Figure 7.8: Max Vertex Distance

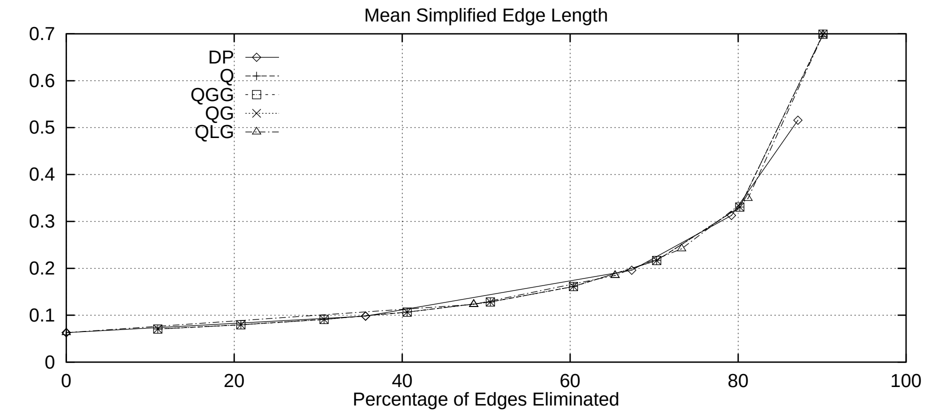

Figure 7.9: Mean Edge Length of the Simplified Polyline

The Dow Jones HK index 1998

For the Dow Jones HK index 1998, DP again has the best MeanVD. The MeanVD of QG, QGG and QLG are similar and better than the MeanVD of Q. See a typical plot in Figure 7.10. The same trend is observed in MaxVD. Notice that Q performs very badly in MaxVD compared with QGG, QG and QLG. See a typical plot in Figure 7.11. Figure 7.12 shows that the mean edge lengths produced by our three improved algorithms (QGG, QG and QLG) are similar, while the mean edge length produced by DP becomes larger beyond 63% of edges eliminated and significantly larger beyond 80% of edges eliminated. ¿From 40% to 90% of edges eliminated, the mean edge length of Q is up to 23% smaller than that of QGG, QG and QLG. We will take the MeanSEL of QGG, QG and QLG as the reference for us to analyze the significance of the error differences among all algorithms.

The differences in MeanVD among Q, QG, QGG and QLG relative to MeanSEL are not significant (up to 1.5%). However, the MaxVD of Q can be 155% of MeanSEL larger than the MaxVD of QG, QGG and QLG.

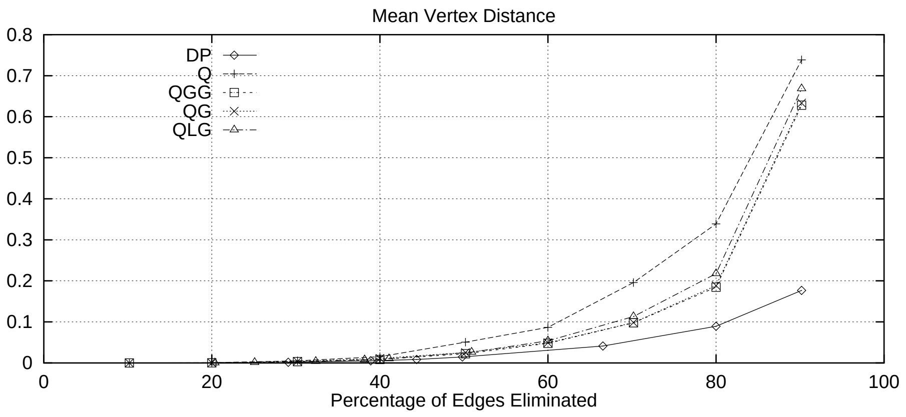

Figure 7.10: Mean Vertex Distance

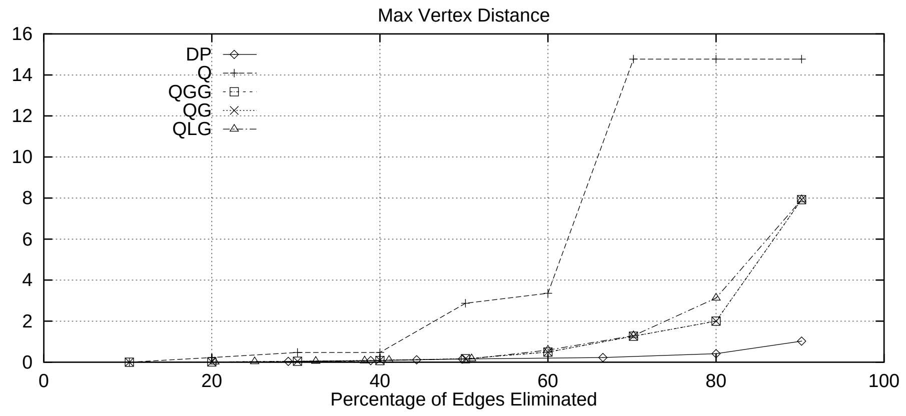

Figure 7.11: Max Vertex Distance

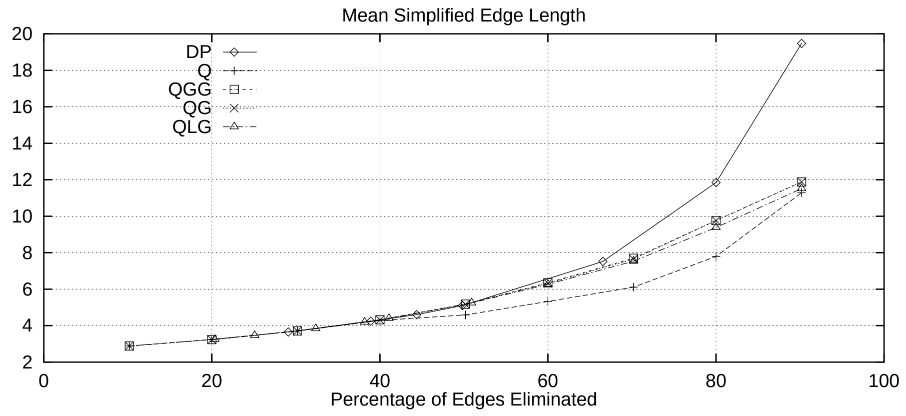

Figure 7.12: Mean Edge Length of the Simplified Polyline

The above results on coastlines and the Dow Jones HK index 1998 lead us to conjecture that our improved quadric error computation (adding extra lines at sharp turns and complementary lines) can make a difference in the Hausdorff distance between the simplified line and the input polygonal line when compared to the basic quadric error algorithm.

7.3.2 Class II: Controlling the input error tolerance

The coastline

The performance of Q and QGG are almost the same in both graphs.
Generally, with the same input error tolerance, DP has larger MeanVD and percentage of edges eliminated. That means DP does more simplification than the other four for the same input error tolerance. In particular, at 90% of edges eliminated, the input error tolerance of both QGG and QG are about 4.5 times that of DP, Q about 4.3 times, and QLG about 3.3 times. It can be explained that DP only takes the distance from the farthest point to compare to the input error tolerance whereas quadric error includes all related distances to the related edges.
Among the quadric error algorithms, QLG is up to about 54% higher than Q and QGG in MeanVD under the same input error tolerance whereas only about 5.4% higher in percentage of edges eliminated. This means that QLG does a little bit more simplification but simultaneously introduce much more error.
Between 0% and 80% of edges eliminated, QLG is up to about 37% lower than Q and QGG in MeanVD under the same input error tolerance, and up to about 26% less in percentage of edges eliminated. This means that QLG does less simplification and thus introduces less error at a particular input error tolerance.
It is worthwhile to point out that the shapes of graphs of MeanVD against the input error tolerance for all algorithms follow linear relation such that the curve for DP has slope of about 1/4, QLG about 1/8, while the other three about 1/12.

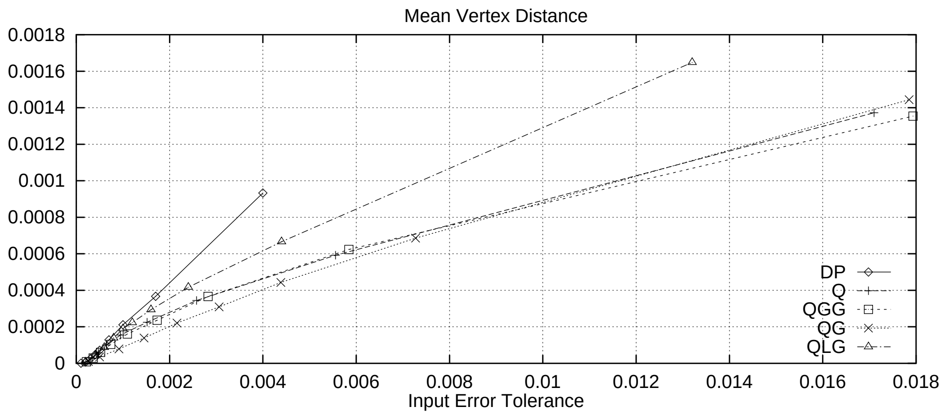

Figure 7.13: Mean Vertex Distance vs Input Error Tolerance

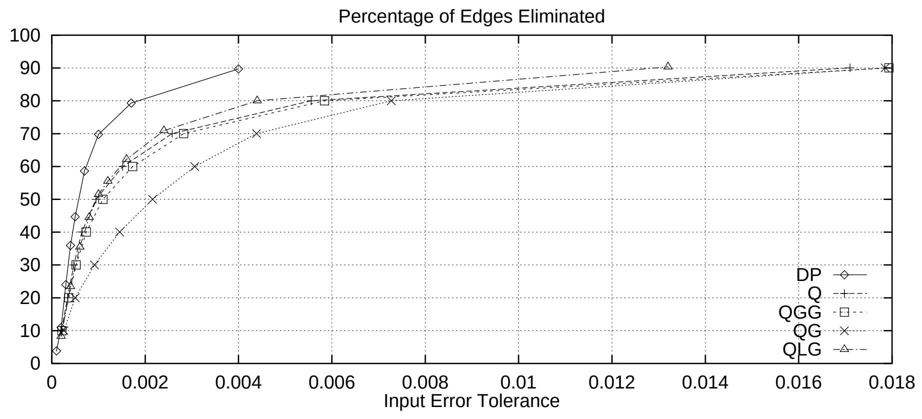

Figure 7.14: Percentage of Edges Eliminated vs Input Error Tolerance

The circle

The graphs of MeanVD and percentage of edges eliminated for Q, QGG and QG are identical. The reason is that no extra line is added since the input polygonal line is nice.

The graphs of MeanVD and percentage of edges eliminated for QLG are than those for Q, QGG and QG. This is because QLG contracts more edges and introduces more error as it processes contractions from one end to the other sequentially in a single pass.

The MeanVD obtained by the quadric error algorithms grows linearly with the input error tolerance. The slope of the graph for QLG is about 1/5 and the slopes of the graphs for Q, QGG and QG are about 1/7.

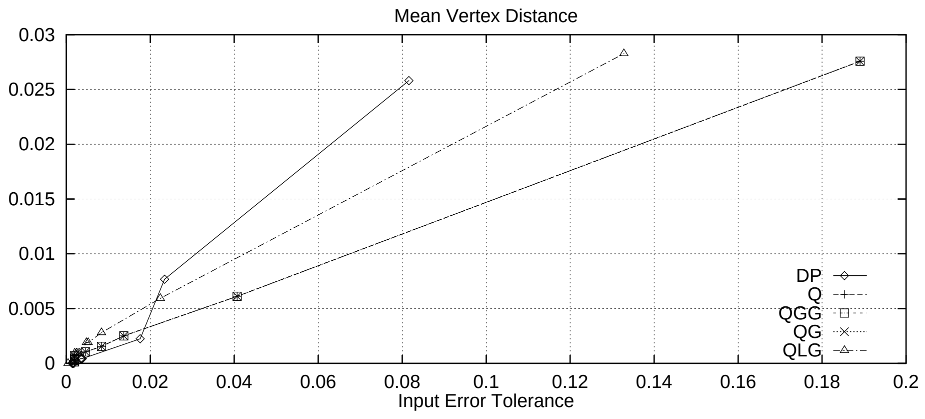

Figure 7.15: Mean Vertex Distance vs Input Error Tolerance

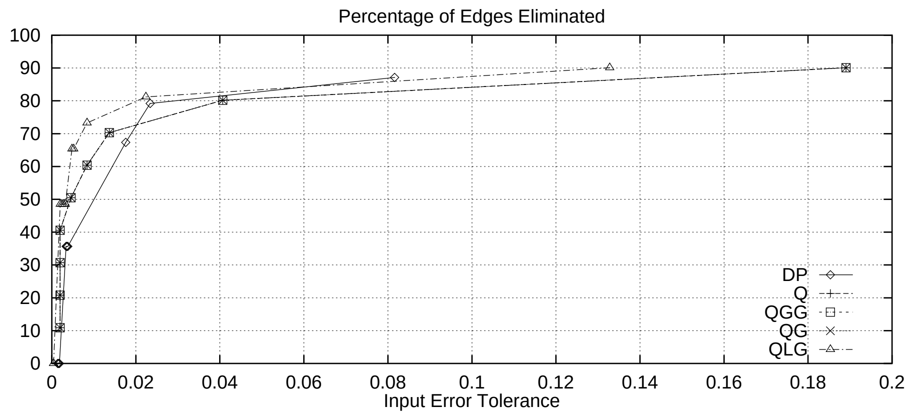

Figure 7.16: Percentage of Edges Eliminated vs Input Error Tolerance

The Dow Jones HK index

The graphs of MeanVD and percentage of edges eliminated for QGG and QG are almost identical.

The graphs of MeanVD and percentage of edges eliminated for DP are almost vertical when compared with the other four algorithms. This shows that it eliminates many vertices even when the input error tolerance is extremely small. The reason is that DP is good in selecting critical point and eliminating those small zigzags even if the input error tolerance is small. In contrast, the zigzags introduce much quadric error for all quadric error algorithms.

The MeanVD produced by Q is up to 6 times higher than QG, QGG, and QLG at a given input error tolerance. Q also eliminates up to 40% more edges than QG, QGG, and QLG at a given input error tolerance. This is due to the extra lines added by QG, QGG, and QLG. On one hand, these extra lines pull the new vertices produced by edge contractions closer to the original polyline. On the other hand, these extra lines also drive the quadric error higher and so fewer edge contractions are performed.

The MeanVD produced by QLG is up to 15% higher than QGG and QG at a given input error tolerance. The percentage of edges eliminated by QLG is also the same as QGG and QG for all input error tolerance. This means QLG does not contract more but introduces up to 15% more error in MeanVD.

The MeanVD obtained by QG, QGG, and QLG grows linearly with the input error tolerance. The slope of the graph for QLG is about 1/21, and the slopes of the graphs for QGG and QG are about 1/25.

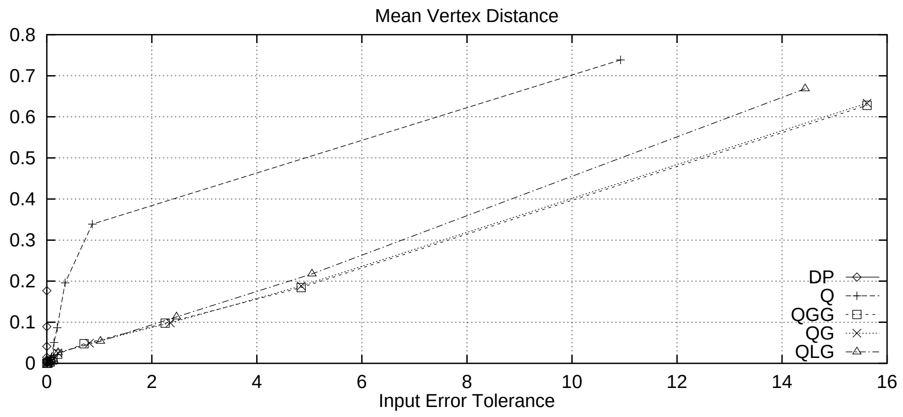

Figure 7.17: Mean Vertex Distance vs Input Error Tolerance

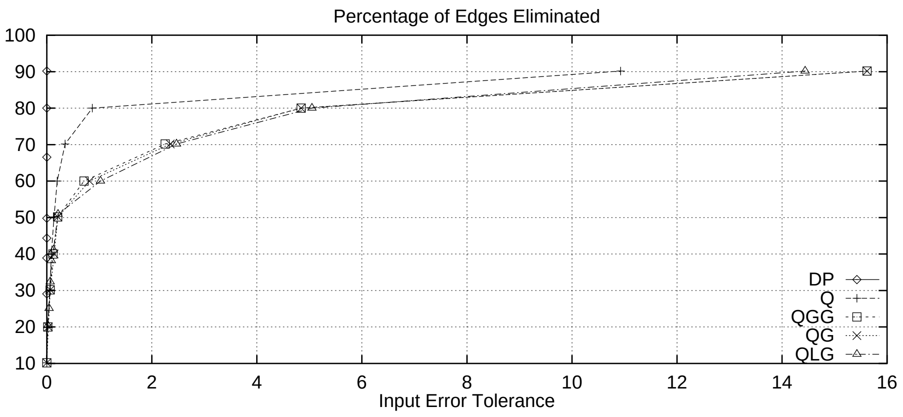

Figure 7.18: Percentage of Edges Eliminated vs Input Error Tolerance

7.4 Some examples of simplification output

In this section, we only take 3 algorithms, DP, Q, and QGG, for consideration. The outputs of QG and QLG are similar to QGG.

7.4.1 The coastline

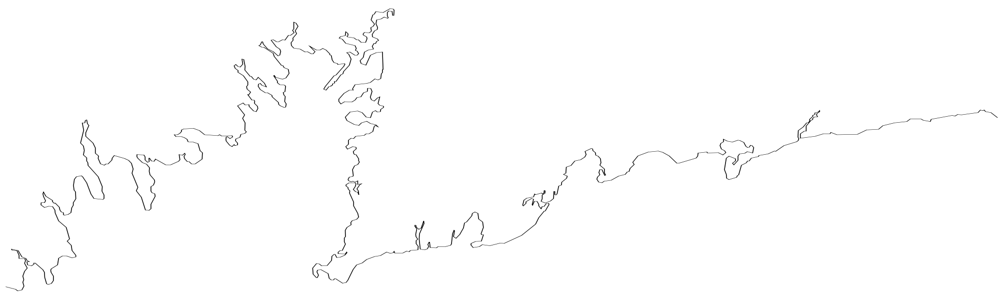
Figure 7.19: DP: Simplified to 50 percent.

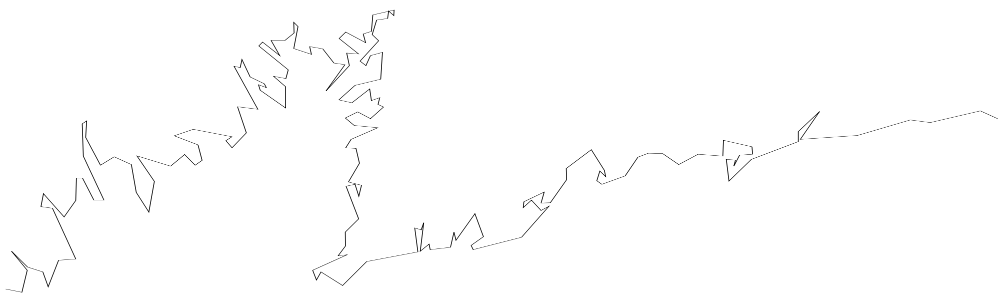
Figure 7.20: DP: Simplified to 10 percent.

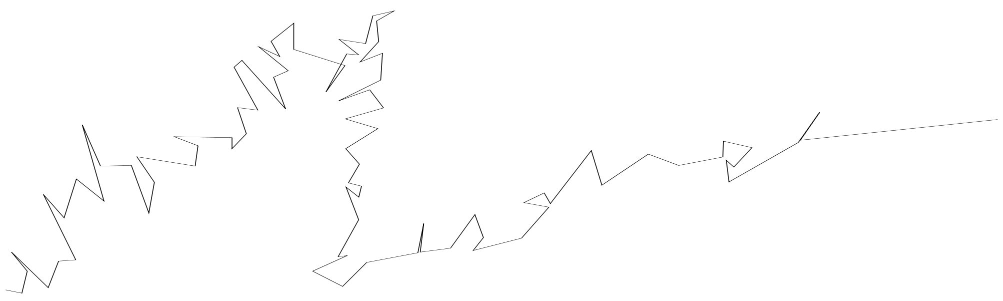
Figure 7.21: DP: Simplified to 5 percent.

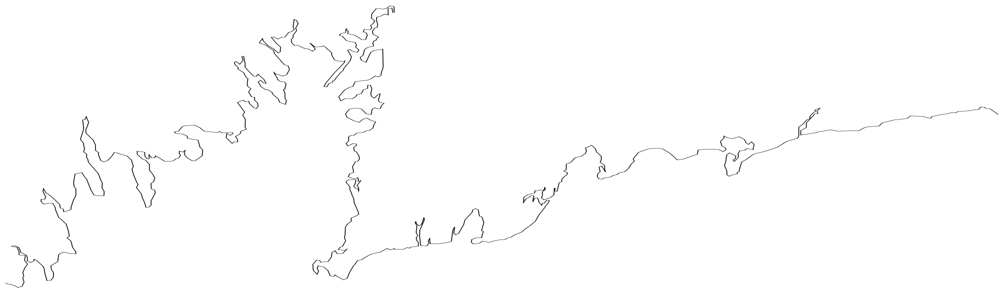
Figure 7.22: Q: Simplified to 50 percent.

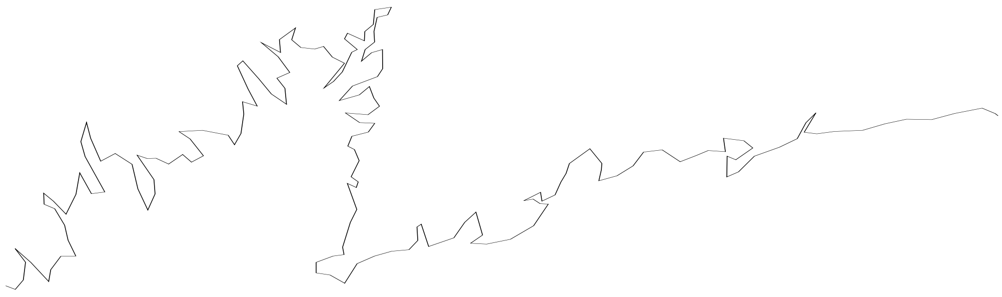
Figure 7.23: Q: Simplified to 10 percent.

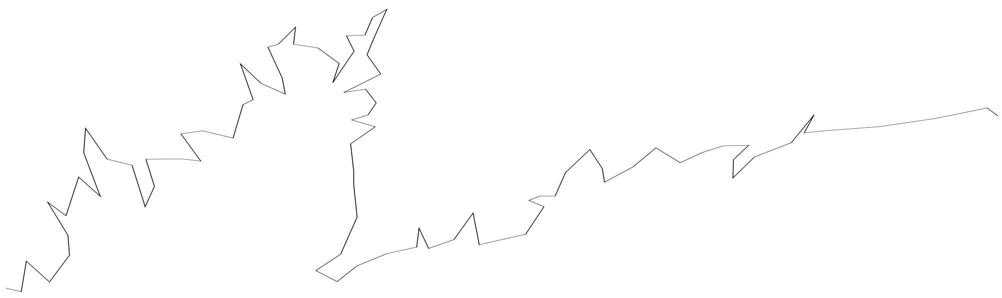
Figure 7.24: Q: Simplified to 5 percent.

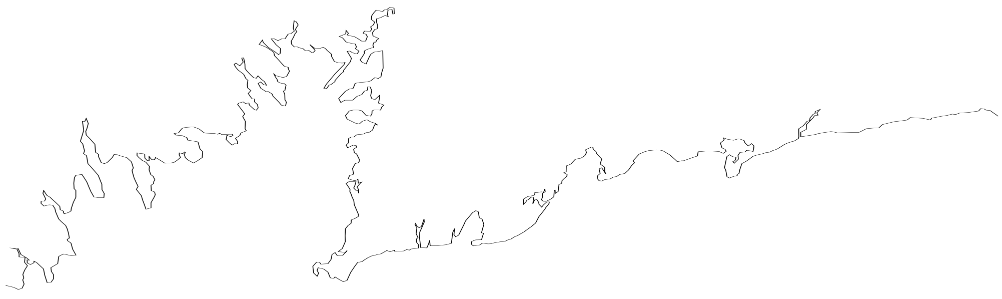
Figure 7.25: QGG: Simplified to 50 percent.

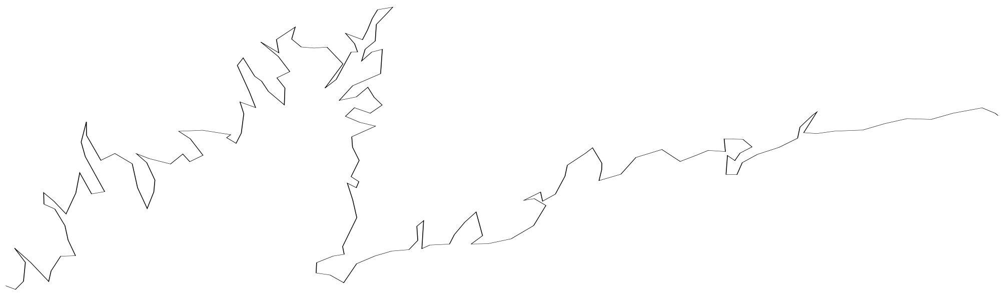
Figure 7.26: QGG: Simplified to 10 percent.

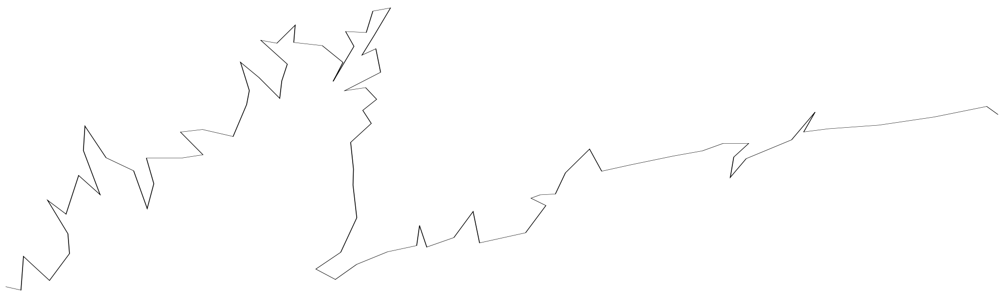
Figure 7.27: QGG: Simplified to 5 percent.

7.4.2 The circle

Two quadric error algorithms produce better results than DP when the circle is simplified to about 10%. This is due to DP's lack of freedom to position the vertices of the simplified polygonal line.

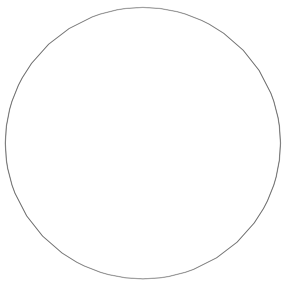
Figure 7.28: DP: Simplified to 50 percent.

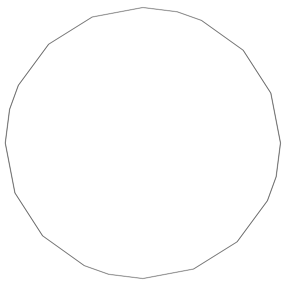
Figure 7.29: DP: Simplified to 20 percent.

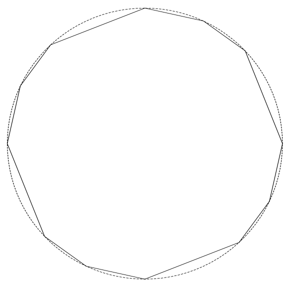
Figure 7.30: DP: Simplified to 10 percent (with original circle).

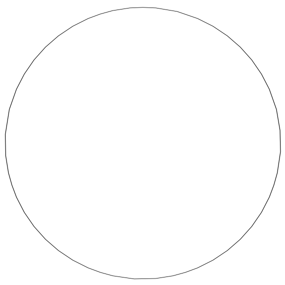
Figure 7.31: Q: Simplified to 50 percent.

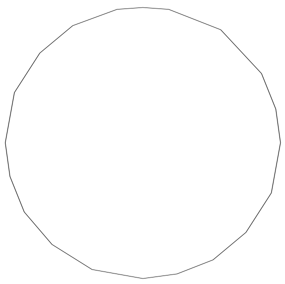
Figure 7.32: Q: Simplified to 20 percent.

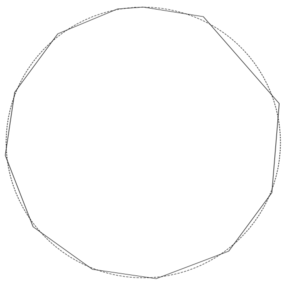
Figure 7.33: Q: Simplified to 10 percent (with original circle).

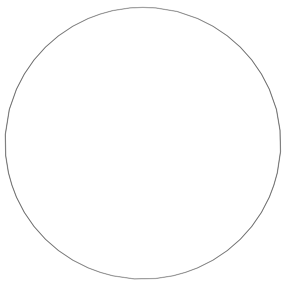
Figure 7.34: QGG: Simplified to 50 percent.

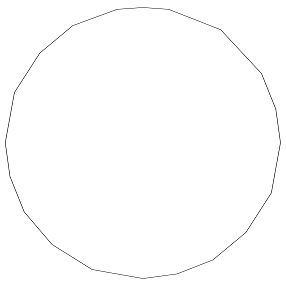
Figure 7.35: QGG: Simplified to 20 percent.

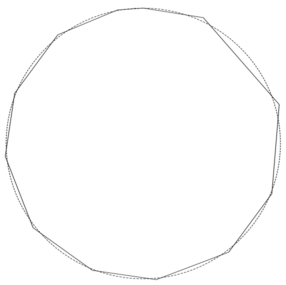
Figure 7.36: QGG: Simplified to 10 percent (with original circle).

7.4.3 The Dow Jones HK index

This is an example of the worst case for algorithm Q. Some important sharp features have already been eliminated even when the percentage of edges eliminated is relatively small. When simplified to 10%, the output quality of QGG is still acceptable although it is inferior to DP.

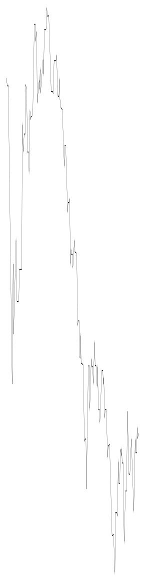

Figure 7.37: DP: Simplified to 50 percent.

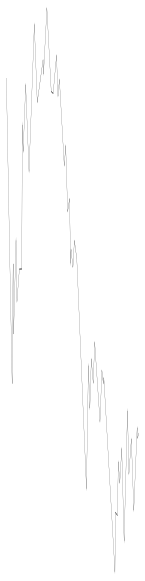

Figure 7.38: DP: Simplified to 20 percent.

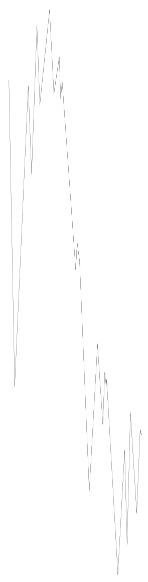

Figure 7.39: DP: Simplified to 10 percent.

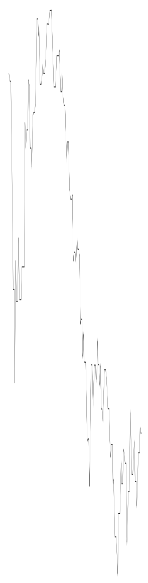

Figure 7.40: Q: Simplified to 50 percent.

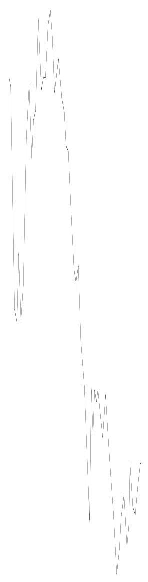
Figure 7.41: Q: Simplified to 20 percent.

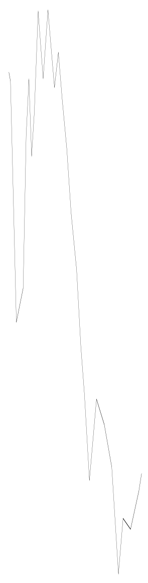
Figure 7.42: Q: Simplified to 10 percent.

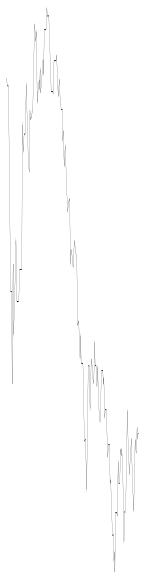
Figure 7.43: QGG: Simplified to 50 percent.

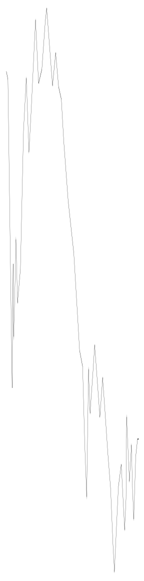
Figure 7.44: QGG: Simplified to 20 percent.

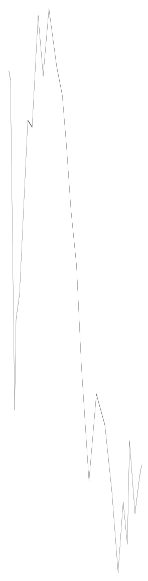

Figure 7.45: QGG: Simplified to 10 percent.

Chapter 8

Conclusion

We propose an edge contraction approach for polyline simplification. The position of the new vertex produced by each edge contraction is computed using the quadric error metric. The basic algorithm Q using this approach does not guarantee any bound on the distance between the simplified polyline and the original polyline. We present improvements to this basic algorithm so that the quadric error actually bounds the directed Hausdorff distance from the simplified polyline to the original one. We derive three improved algorithms QG, QGG, and QLG. QG and QGG use a min-heap to store the quadric error of candidate edge contractions and the one with the minimum quadric error is to be performed next. Each edge contraction takes $O(\log m)$ time, where m is the size of the current simplified polyline. QLG does not use any heap and basically performs edge contractions from one end to the other. Each edge contraction takes constant time. All of our algorithms are simple and easy to implement. Our approach allows the user to directly and interactively increase the simplification.

Although the Douglas and Peucker algorithm produces less error than our algorithms in most of our experiments, the difference is usually insignificant when compared with the mean edge length of the simplified polyline. Among our algorithms, QLG and QGG show similar performance and QG seems to produce inferior output quality.

Chapter 9

Future ork

In fact, we originally got the idea of quadric error metric from surface simplification algorithm by Garland and Heckbert [1]. But their algorithm cannot make sure that there is an error bound for the simplified the surface. We use the same idea to apply the approach into 2-dimensional polyline simplification. With some modification of the algorithm, we finally can result in an error bounded algorithm. This gives us a hope that with some modification of the original surface simplification algorithm, we might be possible to come to an error bounded solution. This is the immediate extension of the research work of this thesis.

In 2-dimensional plane, the distance from a point to a line can be represented by a 3×3 matrix. In 3-dimensional space, the distance from a point to a plane can be represented by a 4×4 matrix. And it is obvious that in k-dimensional space, the distance from a point to a hyperplane can be represented by a $(k+1) \times (k+1)$ matrix. But whether our algorithm can be extended to a surface from any dimension still is a question mark. But it might be an interesting problem. Surface simplification in k-dimension is very useful in scientific visualization. Suppose given a function of $(k-1)$ variables, which we can consider as a k-dimensional surface. Scientists from many disciplines frequently would like to project to surface to one variable or two variables so that they can visualize the trend of the surface with respect to those variables. Surface simplification in k-dimension can result in that we always can do the projection from the simplified surface instead of the original surface. This can save them much of the time and make visualization of scientific data in a real time stance possible.

Simplification for a subdivision in a plane such that the topology does not change is an interesting topic. Furthermore, if a sets of points S also have been given, then it is very challenging task to simplify the subdivision such that not only the topology does not change, but also prevents the point in S crossing over boundaries of the subdivision. However, whether our approach using quadric error metric to do simplification can be applied to the two problems stated above is still open.

Apart from further extension of our algorithms, more thorough and extensive experimentations needs to be taken out to compare our algorithms with other algorithms, not only Douglas and Peucker algorithm.

Bibliography

[1] M. Garland and P.S. Heckbert. Surface Simplification using using quadric error metrics. *SIGGRAPH '97 Proc.*, Aug. 1997. http://www.cs.cmu.edu/˜garland/.

[2] M. Garland and P.S. Heckbert. Survey of Polygonal Surface Simplification Algorithm. *Tech. report, CS Dept.,Carnegie Mellon U.*, May 1997. http://www.cs.cmu.edu/˜ph/.

[3] R. Ronfard and J. Rossignac. Full-range approximation of triangulated polyhedra. *Computer Graphics Forum*, 15(3), Aug. 1997.

[4] R.B. McMaster. Automated Line Generalization. *The Canadian Cartographer*, 24(2), 74-111, 1987.

[5] R.B. McMaster. The Geometric Properties of Numerical Generalization. *Geographical Analysis*, 19(4), 330-346, 1987.

[6] D.H. Douglas and T.K. Peucker. Algorithms for the reduction of the number of points required to represent a digitized line or its caricature. *The Canadian Cartographer*, 10(2), 112-122, 1973.

[7] B. Buttenfield. Treatment of the cartographic line. *Cartographica*, 22(2), 1-26, 1985.

[8] R.G. Cromley. A Vertex substitution approach to numerical line simplification. *Proc. 3rd Symp. on Spatial Data Handling*, 57-64, 1988.

[9] J. Hershberger and J. Snoeyink. Speeding up the Douglas-Peucker line simplification algorithm. *Proc. 5th Symp. on Spatial Data Handling*, 134-143, 1992.

[10] Z. Li and S. Openshaw. Algorithms for automated line generalization based on a natural principle of objective generalization. *Int. J. Geographical Information Systems*, 6, 373-389, 1992.

[11] T. Asano and N. Katoh. Number theory helps line detection in digital images - an extended abstract. *Proc. 4th ISAAC'93*, Lect. Notes in Comp. Science 762, 378-387, 1992.

[12] J.D. Hobby. Polygonal approximations that minimize the number of inflections. *Proc. 4th ACM-SIAM Symp. on Discrete Algorithms*, 93-102, 1993.

[13] Y. Kurozumi and W.A. Davis. Polygonal approximation by the minimax method. *Computer Graphics and Image Processing*, 19, 248-264, 1982.

[14] J. Hershberger and J. Snoeyink. Computing minimum length paths of a given homotopy class. *Computational Geometry: Theory and Applications*, 4, 63-97, 1994.

[15] S.L. Hakimi and E.F. Schmeichel. Fitting polygonal functions to a set of points in the plane. *CVGIP: Graphical Models and Image Processing*, 53(2), 132-136, 1991.

[16] S. Kahan and J. Snoeyink. On the bit complexity of minimum link paths: Superquadratic algorithms for problems solvable in linear time. *ACM Symposium on Computational Geometry*, 151-158, 1996.

[17] D.P. Wang, N.F. Huang, H.S. Chao and R.C.T. Lee. Plane sweep algorithms for the polygonal approximation problems with applications. *Proceedings of International Symposium on Algorithms and Computation*, 515-522, 1993.

[18] M.T. Goodrich. Efficient piecewise-linear function approximation using the uniform metric. *Proceedings of ACM Symposium on Computational Geometry*, 322-331, 1994.

[19] W.S. Chan and F. Chin. Approximation of polygonal curves with minimum number of line segments. *Proc. 3rd ISAAC'92*, Lect. Notes in Comp. Science 650, 378-387, 1992.

[20] W.S. Chan and F. Chin. Approximation of polygonal curves with minimum number of line segments or minimum error. *International Journal of Computational Geometry and Applications*, Vol 6, No 1, 59-77, 1996.

[21] D. Eu and G. Toussaint. On approximating polygonal curves in two and three dimensions. *Graphical Models and Image Processing*, 5, 231-246, 1994.

[22] L.J. Guibas, J.E. Hershberger, J.S.B. Mitchell, and J.S. Snoeyink. Approximating polygons and subdivisions with minimum-link paths. *International Journal of Computational Geometry and Applications*, 3, 383-415, 1993.

[23] H. Imai and M. Iri. Polygonal approximations of a curve-formulations and algorithms. In: G.T. Toussaint(Ed.), *Computational Morphology*, Elsevier Science Publishers, 71-86, 1988.

[24] A. Melkman and J. O'Rourke. On polygonal chain approximation. In: G.T. Toussaint (Ed.), *Computational Morphology*, Elsevier Science Publishers, 87-95, 1988.

[25] J. Hershberger and J. Snoeyink. Cartographic line simplification and polygon CSG formulae and in i $O(n \log^* n)$ time. *WADS 1997*, 93-103.

[26] J.S. Marino. Identification of characteristic points along naturally occurring lines: An empirical study. *The Canadian Cartographer*, 16, 70-80, 1979.

[27] E.R. White. Assessment of line-generation algorithms using characteristic points. *The American Cartography*, 12(1), 17-27, 1985.

[28] J. Hershberger and J. Snoeyink. An $O(n \log n)$ implementation of the Douglas-Peucker algorithm for line simplification. *Proc. of the Tenth Annual Symp. on Computational Geometry*, 383-384, 1994.

[29] D.M. Mark, Conceptual basis for geographic line generalization. *Proc. Auto-Carto*, 9, 68-77, 1989.

[30] USGS, Coastline Extractor. http://crusty.er.usgs.gov/coast/getcoast.html.

[31] Dow Jones Company. http://www.dowjones.com/.

[32] P. Wessel. Version 1.1. wessel@soest.hawaii.edu.

[33] M.L. Bittinger and B.B. Morrel. *Applied Calculus(2nd Edition)*, Addison-Wesley Publishing Co., 441, June 1989.