On Sparse Approximations of Curves and Functions

B.K. Natarajan and J. Ruppert
Hewlett-Packard Laboratories
1501 Page Mill Road
Palo Alto, CA 94304

1. Abstract

Applications such as data compression and geometric modeling make use of low resolution approximations to more complex objects. We study two problems concerning the sparse approximation of curves and functions. The first problem requires the construction of a piecewise linear curve B, given a planar piecewise linear curve A and $\epsilon > 0$, such that the parametric distance $P(A,B) \leq \epsilon$, and B has the fewest break points over all such curves. If $A(t)$ denotes a parametrization of A, and $\| \|$ denotes a metric, $P(A,B)$ is the minimum over all parametrizations $A(t)$ and $B(t)$ of $\max_t \|A(t),B(t)\|$. For the special case of the problem in which every piece of A is longer than 2ϵ, and the L_∞ metric, we give a linear time algorithm; for the general case and the L_∞ metric, we give a linear time pseudo-optimal algorithm. As for the second problem, given integers t and d, construct a C^t piecewise polynomial function of degree d, satisfying a given set of discrete constraints on value and derivative, and consisting of the fewest pieces over all such functions. We give an efficient pseudo-optimal algorithm for the problem.

2. Introduction

First we consider the approximation of planar curves with respect to the parametric distance. Intuitively, the parametric distance between two curves A and B can be thought of as the minimum length leash necessary for a person on curve A to walk a dog along curve B, where neither may "back up". Formally, let $A(t)$ denote a parametrization of a curve A, and let $\| \|$ be a metric on $\mathbf{R}^2$. For two curves A and B, the parametric distance $P(A,B)$ is defined as the minimum over all parametrizations $A(t)$ and $B(t)$ of $\max_t \|A(t),B(t)\|$. The parametric distance is a natural measure of similarity between curves. Determining the distance between two given curves was investigated in Godau (1991), Natarajan (1991a, b), and Alt and Godau (1992). This is useful in handwriting analysis, for example, where a sample may be checked against those in a database to find the closest match. Here, we consider the problem of *computing* a close approximation to a given curve, which finds application to data compression schemes and in systems where geometric data is stored at different levels of resolution.

Given an error tolerance ϵ, and assuming A is a piecewise linear function $A(x)$, Imai and Iri (1986b) and Natarajan (1991a) give linear time algorithms for computing a piecewise linear function $B(x)$ with fewest possible pieces such that $A(x)-\epsilon \leq B(x) \leq A(x)+\epsilon$ for all x. Natarajan uses the general visibility algorithm of Suri (1988), whereas Imai and Iri develop a simpler algorithm for monotone polygons. Alt et al. (1990) consider the approximation of closed curves by simple convex polygons such as squares and rectangles. Here, we generalize the work of Imai and Iri (1986b) and Natarajan (1991a) to the case of curves in the plane:

Problem 1: Given a piecewise linear curve A in $\mathbf{R}^2$ with m break points, and $\epsilon > 0$, compute a piecewise linear function B such that $P(A,B) \leq \epsilon$, and B has the fewest break points over all such curves.

Godau (1991) gives an $O(m^3)$ time pseudo-optimal algorithm for Problem 1. By pseudo-optimal we mean that the number of pieces in the output is within a constant factor of optimal, for the given ϵ. Here, we give a linear time algorithm, for the L_∞ metric and the restriction that every piece of A is longer than 2ϵ. If this restriction is relaxed, we have a linear-time but pseudo-optimal algorithm. Independently, Guibas et al. (1991) report a linear-time algorithm for the problem for general metrics, but under the restriction that no two break points of A are within 2ϵ of each other.

We also consider a variant problem which adds the restriction that the break points of the constructed curve must be a subset of the break points of the given curve. Toussaint (1985), Imai and Iri (1986b), Melkman and O'Rourke (1988) approach this problem using dynamic programming. Of these, Melkman and O'Rourke (1988) offer the best run time of $O(m^2\log(m))$. Ihm and Naylor (1991) offer an $O(m^3)$ extension to curves in $\mathbf{R}^3$. In contrast, our linear time algorithm for Problem 1 can be extended to give a pseudo-optimal solution for this problem.

The second problem we study allows approximation by a function that is a piecewise polynomial of degree d. This richer set of representations may allow for greater compression ratios for certain types of data. Efficient piecewise polynomial approximation of piecewise *linear* functions appears difficult, instead we assume the input is a set of discrete samples, possibly including derivative constraints.

Let $\mathbf{R}$ be the reals, and $\mathbf{N}$ the natural numbers. For a function $B:\mathbf{R}\rightarrow\mathbf{R}$ and non-negative integer t, B^t denotes the t^{th} derivative of B. B is said to be $\mathbf{C}^t$ if B is continuous up to and including the t^{th} derivative.

Problem 2: Given are (1) m constraints of the form (x,y^-,y^+,k) where $x \in [0,1]$, $y^-,y^+ \in \mathbf{R}$, and k is a non-negative integer, and (2) non-negative integers d and t. Construct a function $B:[0,1]\rightarrow\mathbf{R}$, if such exists, such that (1) B is piecewise polynomial of degree d, (2) B is $\mathbf{C}^t$, (3) for each of the m input constraints, $y^- \le B^k(x) \le y^+$, and (4) B has the fewest number of pieces over all such functions.

The above problem has a variety of applications, including geometric modeling and data compression. See Lyche and Morken (1987), Netravali and Haskell (1989), Rosenberg (1990), Ishijama et al. (1983) amongst others. In this paper, we use the results of Rivlin (1964) to obtain a pseudo-optimal algorithm for Problem 2. The algorithm is pseudo-optimal in the sense that the number of pieces in the constructed piecewise polynomial function is optimal within a factor of $\left(1+\lceil(t+1)/(d-t)\rceil\right)$, assuming that $t < d$. For $t = 0$, this factor reduces to 2 for all d, and for $t = d-1$, the factor reduces to $d+1$. Our algorithm uses linear programming as a basic step. Assuming d is fixed, we can use a fixed dimension linear programming algorithm that runs in time linear in m, the number of constraints, to get an overall run time of $O(m\log(m))$.

The problem of simplifying B-splines is closely related to Problem 2 except that in the former the input is a piecewise polynomial function rather than a discrete sample. Lyche and Morken (1987) present a heuristic for the simplification of B-splines, in the sense of seeking a spline on a subset of the knots of the original spline, with respect to a given error bound. Nurnberger (1989) presents a more general procedure for the simplification of splines.

3. Approximation Under the Parametric Distance

In this section, we consider Problem 1, the parametric approximation of planar piecewise linear curves. Our algorithm builds upon the techniques of Imai and Iri (1986a), who show how to approximate piecewise linear *functions*. The basic idea of their algorithm is to construct a polygon by offsetting the input function, and then to use a visibility algorithm to compute a minimum link path within the polygon. The minimum link path, computable in linear time, is output as the approximation function. For the more general case of piecewise linear *curve* inputs, we show that a simple offset polygon does not suffice, and instead construct a particular type of self-overlapping polygon, in which we can use a modified visibility algorithm to produce an approximation.

The input to be approximated is a piecewise linear curve A with m breakpoints $a_1,\ldots,a_m$, and m-1 segments $A_1,\ldots,A_{m-1}$. Our algorithm widens the curve A by ϵ to get an error tunnel A^ϵ. The goal is that any path in A^ϵ, starting within ϵ of the first point of A and ending within ϵ of the last point of A, should be a valid parametric approximation to A. We compute a minimum-link path from one end of

A^ϵ to the other, using visibility polygon techniques. The resulting path will be a parametric approximation B within ϵ of A having the fewest segments possible. Throughout this section, our error ϵ will be measured using the L_∞ metric.

An idea that fails is to let A^ϵ be the ϵ-offset polygon of A. The ϵ-offset of A is the polygon consisting of all points within a distance ϵ of A, as shown in Figure 1. The problem with the ϵ-offset polygon is that it contains paths that "shortcut" the parametric approximation. For example, curve C in Figure 1 contains no point within ϵ of a_2. In fact, there is no simple polygon that contains all valid approximations and rules out all invalid ones. Instead, we construct a type of self-overlapping polygon.

Our "polygon" A^ϵ will offset the breakpoints and segments of A separately. For each breakpoint a_i, let S_i be the square containing all points within ϵ of a_i. (S_i is an L_∞ unit circle.) For each segment A_i of A start with the set of points within ϵ of A_i and then subtract the squares squares S_i and S_{i+1} representing the endpoints of A_i, to leave a tube T_i as shown in Figure 2. If segment A_i is short, the subtraction may disconnect T_i. We rule this out for now by requiring all segments of A to have length $\geq 2\epsilon$. Later, we will say how to get around this restriction.

Now imagine the S_i's and T_i's as being cut out of separate sheets of paper, and "stitch" them back together as follows. As shown in Figure 3, each T_i shares two corresponding edges with each of S_i and S_{i+1}. Join T_i to both squares along these edges. At any overlaps of S_i's and T_i's we arbitrarily say that the higher indexed item is "on top". In general, A^ϵ cannot be realized as a simple planar polygon, but we can view it as a collection of sheets, stitched together along certain segments. Paths in A^ϵ may switch from sheet to sheet when going between attached S_i's and T_i's, but may not otherwise cross the boundaries of S_i's or T_i's. We refer to A^ϵ as the *error tunnel*. The main result of this section is to show that the polygonal visibility algorithm of Imai and Iri (1986a) can be adapted to work in these error tunnels.

Any path that stays within A^ϵ certainly stays within ϵ of A, and we would like to say that any path from S_1 to S_m that stays within A^ϵ is a parametric approximation to A. Unfortunately, such a path might "back up", preventing a valid parametrization of A. We will be able to show that the path we construct is a valid approximation to a parametrization of A, but we postpone this until after the description of our algorithm.

Visibility polygon techniques will be used in constructing the approximation. Two points p and q in A^ϵ are said to be mutually visible if there is a straight path between them that stays within A^ϵ. The path may switch sheets where they are stitched together. The visibility region $V(p)$ of a point p is given by

$$V(p) = \{q \in A^\epsilon | p, q \text{ are mutually visible}\},$$

and similarly, the visibility region $V(S)$ of any set S of points (such as a segment or a square S_i) is $V(S) = \bigcup\{V(p) | p \in S\}$. Two distinct points q_1 and q_2 can have the same "position", but be on different sheets. It is possible that only one is visible from some other point.

3.1 The Algorithm

We use a greedy algorithm to construct an approximation B to the curve A. At each step, we "look" as far as possible down the tunnel A^ϵ, and add one segment to B. The algorithm is fairly simple in concept, but requires quite a few bookkeeping details. Since these details are mostly the same as in Imai and Iri (1986a), we highlight only the significant differences in this extended abstract.

We first illustrate the algorithm with an example. In Figure 4 are shown the input curve A and the resulting error tunnel A^ϵ. Our approximation must start within the square S_1, and we would like the first segment to reach as far as possible. The visibility region V_1, shown shaded in Figure 5, contains all points in A^ϵ that are visible from some point in S_1. Next is shown V_2, consisting of all points visible from some point in V_1. Note that any point in V_2 is visible from the *window* separating V_1 from V_2. Since the final square S_4 is in V_2, we are done, and we can output the approximation shown in Figure 6.

An outline of our algorithm is as follows.

Algorithm 1

```
let the initial window $w_0$ be $S_1$
$i=0$
repeat
    compute visibility region $V_{i+1}$ from
        current window $w_i$
    get new window $w_{i+1}$ from $V_{i+1}$
    $i=i+1$
until $V_i$ contains part of final square $S_m$
/* place a breakpoint of $B$ on each
    window, working backwards */
let $b_{i+1}$ be any point on $S_m$ visible from $w_i$
for $j = i, i-1, \ldots, 1$ do
    let $b_j$ be a point on $w_j$ that sees $b_{j+1}$
end
```

Next we describe how to implement the steps that compute visibility regions and windows. Visibility regions are computed by working down the tunnel, considering each square S_i in turn as something that may delimit the lines of sight. Eventually, some S_l will be invisible from the window. At that point, we can determine the visibility region and the window from which to start computing the next visibility region. This is illustrated in Figure 7, where lines of sight originating in S_1 can reach S_5, but not S_6. The two rays t_l and t_r delimit the lines of sight that pass through all of $S_1, \ldots, S_5$. We refer to them as the *left extremal tangent* t_l and the *right extremal tangent* t_r. They determine the "most clockwise" and "most counter-clockwise" lines of sight that pass through all of the S_i's considered.

Efficient maintenance of the extremal tangents is the crux of the approximation algorithm. Each tangent remains "tight" at corners of two S_i. As each S_i is considered in turn, the tangents are swiveled around one of these tight points so they pass through S_i and remain tight at two points. Imai and Iri (1986a) maintain "upper" and "lower" hulls while advancing down the polygon, which allows them to quickly determine the next tight point. Their technique relies on the fact that the polygon is monotone and simple. Our error tunnel is neither, but we show in the full paper how to maintain left and right "pseudo-hulls", which allow the same maintenance operations. The overall linear time bound results because any edges of the current pseudo-hull that are examined when a new S_i is considered will never be part of the hull again, and hence the cost per edge is a constant number of operations, since it is only looked at when inserted or deleted.

3.2 Optimality of the approximation

To show that our parametric approximation has the fewest possible segments of any approximation within ϵ of the input curve, we use the fact that windows divide the error tunnel. In the full paper, we use this to show that any parametric ϵ-approximation has at least one breakpoint in each visibility region, and hence at least as many breakpoints as our approximation.

We must also show that the minimum link path can be parametrized as needed. This seems reasonable intuitively, but a rigorous proof that the minimum link path doesn't "back up" too much involves a number of details that are omitted here.

The following summarizes the approximation algorithm.

Proposition 1: Given a piecewise linear curve A with m breakpoints, having all segments longer than 2ϵ in the L_∞ metric, Algorithm 1 constructs in $O(m)$ time, a piecewise linear curve B with a minimum number of pieces *such that the parametric distance $P(A,B) \leq \epsilon$* in the L_∞ metric. We can also describe the parametrizations $A(t)$ and $B(t)$ such that $\|A(t),B(t)\| \leq \epsilon$ for all t.

In the full paper, we describe pseudo-optimal linear time algorithms for the following variants: (1) the restriction to input segments longer than 2ϵ is removed; (2) the output break points are required to be a subset of the input breakpoints; (3) the output break points are required to lie on the input curve.

4. Approximating Functions

Proposition 2: Rivlin (1964). Given are (1) m constraints of the form (x,y^-,y^+,k) where $x \in \mathbf{R}$, $y^-,y^+ \in \mathbf{R}$ and k is a non-negative integer, and (2) a natural number d. We can construct an instance of linear programming with $d+2$ variables and $2m+1$ constraints, whose solution determines the coefficients of a polynomial $P:\mathbf{R}\rightarrow\mathbf{R}$ of degree d, if such exists, such that for each of the m input constraints, $y^- \leq P^k(x) \leq y^+$.

Proof: Omitted in this extended abstract.

□

Proposition 2 leads us to an algorithm for Problem 2. Let (x_1,y_1^-,y_1^+,k_1), (x_2,y_2^-,y_2^+,k_2), ... , (x_m,y_m^-,y_m^+,k_m), be the m input constraints such that $x_1 \leq x_2 \leq x_3 \cdots \leq x_m$. Starting with x_1, find the greatest index i_1 such that $x_{i_1+1} \neq x_{i_1}$ and all constraints for $x_1,x_2...,x_{i_1}$ can be satisfied by a polynomial of degree d. This can be achieved by invoking Proposition 2 in a binary search. Determine such a polynomial, say P_1, and delete all constraints for x_1 through x_{i_1}. Then proceed iteratively with the remaining constraints. We will then have a sequence of polynomials $P_1,P_2,...,P_l$, where P_1 satisfies all constraints for x_1 through x_{i_1}, P_2 satisfies all constraints for x_{i_1+1} through x_{i_2} and so on. Consider the piecewise polynomial C that consists of P_1 on the interval $[x_1,x_{i_1}]$, P_2 over the interval $[x_{i_1+1},x_{i_2}]$ and so on. C is almost the piecewise polynomial that we desire, except that it is undefined on the intervals $[x_{i_1},x_{i_1+1}]$, $[x_{i_2},x_{i_2+1}]$ etc. Proposition 3 helps remedy the situation.

Proposition 3: Given positive integers d and $t < d$, and $x_1,y_1^0,y_1^1 \cdots y_1^t$ and $x_2,y_2^0,y_2^1 \cdots y_2^t$, where $0 \leq x_1 < x_2 \leq 1$ and the y's are from **R**. We can construct a C^t continuous piecewise polynomial Q of degree d consisting of $\lceil(t+1)/(d-t)\rceil$ pieces such that for $0 \leq i \leq t$, $Q^i(x_1) = y_1^i$ and for $0 \leq i \leq t$, $Q^i(x_2) = y_2^i$. Furthermore, this construction can be carried out in time $O((d+1)^4)$.

Proof: In this abstract, we omit the proof, which involves setting up a simple system of linear equations.

□

We can now combine Propositions 2 and 3 to give Algorithm 2, a pseudo-optimal algorithm for Problem 2. Within the algorithm, we represent a piecewise polynomial C as the pair (π_C,X_C), where π is a sequence of polynomials $P_1,P_2 \cdots$, and X is a sequence of pairs of reals of the form (α_i,β_i), with the significance that

$$C(x) = P_i(x),\ x \in [\alpha_i,\beta_i].$$

Proposition 4: Algorithm 2 (1) outputs a function B that is piecewise polynomial of degree d, satisfies the input constraints, and has the fewest number of pieces of any such function within a factor of $\left(1+\lceil(t+1)/(d-t)\rceil\right)$, and (2) runs in time $O\left(T(d+2,2m+1)\log(m)+m(d+1)^4\right)$, where $T(v,c)$ is the time required to solve a linear programming problem with v variables and c constraints.

Proof: Omitted in this extended abstract.

□

```
Algorithm 2
input: an instance of Problem 2;
begin
    Let
    $(x_1,y_1^-,y_1^+,k_1)$,
    $(x_2,y_2^-,y_2^+,k_2)$, ... ,
    $(x_m,y_m^-,y_m^+,k_m)$,
    be the $m$ input constraints such that
    $x_1 \leq x_2 \leq x_3 \cdots \leq x_m$.
    let $B = (\pi_B,X_B)$, $\pi_B$ and $X_B$ initially null;
    and $C = (\pi_C,X_C)$, $\pi_C$ and $X_C$ initially null;
    let $i = 0$;
    While $i \leq m$ do
        Using Proposition 2 in binary search and an
            algorithm for linear programming, find
            the largest index $j \geq i$ such that
            (1) $x_{j+1} \neq x_j$ and
            (2) there exists a polynomial $P$ of degree
            $d$ that satisfies constraints $i$ through $j$;
        append $P$ to $\pi_C$.
        append $(x_i,x_j)$ to $X_C$.
    end
    denote the elements of $X_C$ as $(\alpha_1,\beta_1)$, $(\alpha_2,\beta_2)$, $\cdots$
    denote the elements of $\pi_C$ as $P_1,P_2 \cdots$
    for $P_1,P_2 \cdots ,P_j \cdots \in \pi_C$ do
        $\beta_j = \beta_j+(\alpha_{j+1}-\beta_j)/4$;
        $\alpha_{j+1} = \alpha_{j+1}-(\alpha_{j+1}-\beta_j)/4$;
        Using Proposition 3, compute piecewise
        polynomial $Q = (\pi_Q,X_Q)$ that is $C^t$
        continuous with $P_j$ and $P_{j+1}$ at
        at $\beta_j$ and $\alpha_{j+1}$ respectively.
        append $P_j,\pi_Q$ to $\pi_B$;
        append $(\alpha_j,\beta_j),X_Q$ to $X_B$;
    end
    output $B$;
end
```

5. Conclusion

We presented efficient algorithms for two approximation problems. Both were variants of a previously studied problem, the piecewise linear approximation of piecewise linear functions. In the first case, we generalized the input to include curves. In the second case, we generalized the output representation to allow higher degree polynomials. In addition, we studied some variants of the problems, such as requiring the output's breakpoints to be a subset of those in the input. A number of interesting questions remain to be pursued. For instance, can we achieve an optimal approximation in the curve case (Problem 1) if short input segments are allowed? Can the algorithm in the piecewise polynomial case (Problem 2) be improved to run in linear time? This appears to require maintaining the optimal solution of a linear program as constraints are added, until no feasible solution exists. For piecewise linear output, we can do this in linear total time because the linear program is 2-dimensional, and the constraints are presented in order of increasing slope. For polynomials, the solution polytope is higher-dimensional, which appears to make the maintenance more difficult. Finally, the question of approximating a 2-dimensional surface lying in 3 dimensions is important in practice. Mitchell and Suri (1992) give some initial results on this problem, which seems to require significantly new ideas.

6. References

Alt, H., Blomer, J., Godau, M., and Wagener, M., (1990). Approximation of Convex Polygons. Proc. of the 17th ICALP, pp.703-716.

Alt, H. and Godau, M. (1992). Measuring the resemblance of polygonal curves. To appear 8th ACM Symposium on Computational Geometry.

Ihm, I, and Naylor, B., (1991). Piecewise linear approximations of digitized curves with applications. In Scientific Visualization of Physical Phenomena, pp. 545-568, N.M Patrilakis, editor, Springer-Verlag, New York.

Imai, H., and Iri, M., (1986a). An optimal algorithm for approximating a piecewise linear function. J. of Information Processing, Vol. 9, No. 3, pp. 159-162.

Imai, H., and Iri, M., (1986b). Computational geometric methods for polygonal approximations of a curve. Computer Vision, Graphics and Image Processing, Vo. 36, pp.31-41.

Ishijama M., Shin, S-B., Hostetter, G.H., and Sklansky, J. (1983). Scan-along polygonal approximation for data compression of electrocardiograms. IEEE Trans. on Biomedical Engg., Vol. BME-30, No. 11, pp.723-729.

Godau, M., (1991). A natural metric for curves - computing the distance for polygonal chains and approximation algorithms. Proc. of the 8th STACS, Springer-Verlag, New York.

Guibas, L., Hershberger, J.E., Mitchell, J.S.B, and Snoeyink, J.S., (1991). Approximating polygons and subdivisions with minimum link paths. Second Annual International Symposium on Algorithms, Taiwan, December 1991.

Lyche, T., and Morken, K., (1987). Knot Removal for parametric B-spline curves and surfaces. Computer Aided Geometric Design, Vol. 4, No. 3, pp. 217-230.

Melkman, A. A., and O'Rourke, J. (1988). On polygonal chain approximation. In Computational Morphology, pp. 87-95, G. T. Toussaint, editor, North Holland.

Mitchell, J.S.B., and Suri, S. (1992). Separation and approximation of polyhedral objects. In Proc. ACM-SIAM Symposium on Discrete Algorithms, 1992, pp. 296-306.

Natarajan, B.K., (1991a). On piece-wise linear approximations to curves, Tech. Rep. HPL-91-36, Hewlett Packard Labs, Palo Alto, CA.

Natarajan, B.K., (1991b). On piecewise linear approximations to curves, SIAM Conference on Geometric Design, 1991.

Netravali, A.N., and Haskell, B.G., (1988). Digital Pictures, Plenum Press, New York, NY.

Nurnberger, G., (1989). Approximation by spline functions. Springer Verlag, New York.

Rivlin, T., (1969). An Introduction to the Approximation of Functions. Dover, New York.

Rosenberg, C., (1990). A lossy image compression algorithm based on nonuniform sampling and interpolation of image intensity surfaces. M.S. Thesis, Dept. of Electrical Engineering and Computer Science, Mass. Inst. of Tech.

Suri, S., (1988). On some link distance problems in a simple polygon. IEEE Trans. on Robotics and Automation, vol 6, No.1, pp.108-113.

Toussaint, G. T., (1985). On the complexity of approximating polygonal curves in the plane. In Proc. of the IASTED Int. Symp. on Robotics and Automation, pp.59-62.

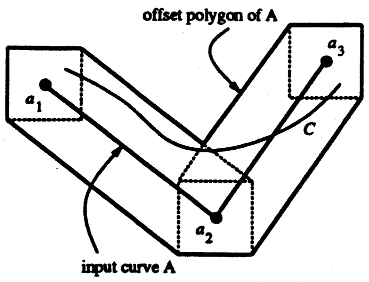

Figure 1: Sample curve A, its ϵ-offset , and curve C, which takes a shortcut at a_2.

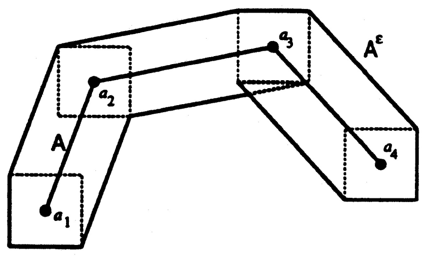

Figure 4: Input curve A and error tunnel A^ϵ.

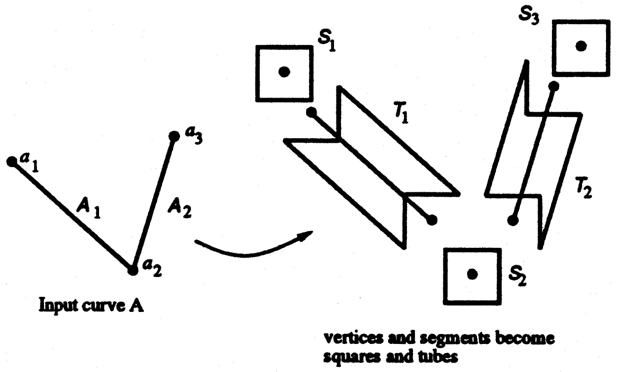

Figure 2

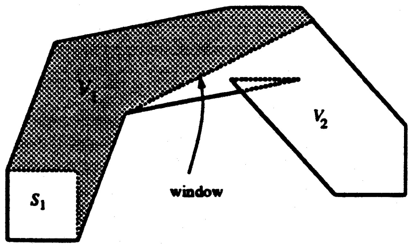

Figure 5: First and second visibility regions, and window separating them.

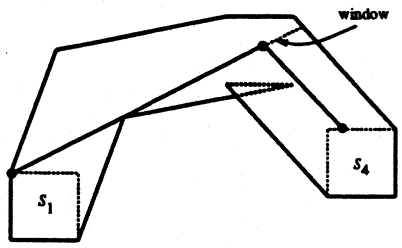

Figure 6: Resulting approximation.

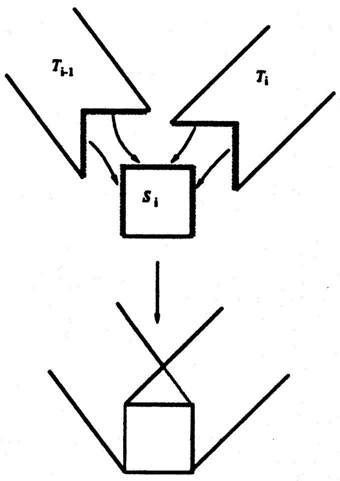

Figure 3: "Stitching" together the tubes and segments to get the (self-overlapping) error tunnel.

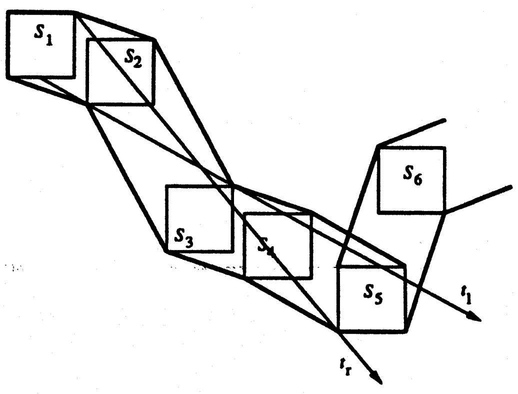

Figure 7: Tangents t_l and t_r, from S_1.