IBM Journal of Research and Development, vol. 34, no. 35, September 1990, pp. 753-769.

Finding Compact Coordinate Representations for Polygons and Polyhedra

Victor Milenkovic
Harvard University
Division of Applied Science
Cambridge, MA 02138

Lee R. Nackman
IBM Research Division
Thomas J. Watson Research Center
P. O. Box 218
Yorktown Heights, NY 10598

Abstract

Practical solid modeling systems are plagued by numerical problems that arise from using floating-point arithmetic. For example, polyhedral solids are often represented by a combination of geometric and combinatorial information. The geometric information might consist of explicit plane equations, with floating-point coefficients; the combinatorial information might consist of face, edge, and vertex adjacencies and orientations, with edges defined by face-face adjacencies and vertices by edge-edge adjacencies. Problems arise when numerical error in geometric operations causes the geometric information to become inconsistent with the combinatorial information. These problems could be avoided by using exact arithmetic instead of floating-point arithmetic. However, some operations, like rotation, increase the number of bits required to represent the plane equation coefficients. Since the execution time of exact arithmetic operators increases with the number of bits in the operands, the increased number of bits in the plane equation coefficients can cause performance problems. One proposed solution to this performance problem is to *round* the plane equation coefficients without altering the combinatorial information. We show that such rounding is NP-complete.

1 Introduction

To achieve reliable programs that implement geometric computations, one must understand and control the effect of numerical error. One approach to controlling numerical error is to eliminate it by working only with objects and transformations that can be represented with numbers in a representable subfield of the reals. For linear objects (e.g., points, lines, planes), using exact (i.e., arbitrary precision) rational numbers would allow exact computation of intersections. Unfortunately, the situation is different with rotation, a commonly used geometric transformation: rotating a line with rational coefficients can yield a line with irrational coefficients. Pythagorean triples can be used to approximate any rotation arbitrarily closely by a *rational rotation*, a rotation that can be represented by a matrix with rational entries [7, Section 4.3.3]. Unfortunately, when rational representations are used, iteration of geometric transformations, like rotation, can cause unbounded growth in the precision needed to represent transformed objects, which can lead rapidly to unacceptable performance of geometric computations. We shall focus here on the precision growth problem.

Precision growth arises from a sequence of geometric transformations. As a simple example, consider a sequence of r rotations about the coordinate axes (any rotation can be expressed as a sequence of rotations about the axes), each of which is to be approximated as above by a rational rotation with an angle of rotation accurate to one part in 2^{-P}. It is easy to show that after applying these r rotations to a point, the precision required per transformed coordinate is $O(rP)$. Precision growth can be limited by interspersing in the sequence *rounding* operations that closely approximate the transformed geometric objects with geometric objects that require less precision to represent. But what does it mean to approximate a geometric object and how should rounding operations be interspersed in the sequence of transformations? These questions are discussed at some length in [7, Section 4.3].

Sugihara [10] has investigated approximating individual hyperplanes in n-dimensional space. He formulates the problem as follows. Given $n+1$ positive integers $Q_1, \cdots, Q_{n+1}$, the hyperplane

$$a_1x_1 + a_2x_2 + \cdots + a_nx_n + a_{n+1} = 0$$

is approximated by the hyperplane

$$b_1x_1 + b_2x_2 + \cdots + b_nx_n + b_{n+1} = 0$$

where $a_1, a_2, \ldots, a_{n+1}$ are real numbers and $b_1, b_2, \ldots, b_{n+1}$ are integers such that $|b_i| \le Q_i$. The Q_i bound the precision of the coefficients of the approximation. The approximation is considered good if b_i/b_j is close to a_i/a_j, for all i and j. Since finding an appropriate set of b_i's is very difficult, Sugihara considers the following problem. Let k be the integer such that the ratio $|a_k|/Q_k$ is maximum. Dividing the original equation by a_k, he obtains the equation

$$w_1x_1 + w_2x_2 + \cdots + w_nx_n + w_{n+1} = 0$$

where $w_i = a_i/a_k$. He then considers the more tractable approximation problem of finding good rational approximations p_i/q to the w_i for some number $q \le Q_k$. If the rational approximations are chosen such that $|p_i/q| \le w_i$, then the equation

$$p_1x_1 + p_2x_2 + \cdots + p_nx_n + p_{n+1} = 0$$

approximates the original hyperplane and satisfies the bounds on the precision of the coefficients. In order to generate a *good* approximation, Sugihara proposed several heuristic methods for finding a set of integers $\{q, p_1, \ldots, p_{n+1}\}$ that minimizes

$$\max_i \left| w_i - \frac{p_i}{q} \right|.$$

He evaluated these approximation methods by applying them to a large number of randomly generated lines. Although each individual line is approximated well, nothing constrains the ensemble of approximations to reproduce any more global structure. This is quite apparent in the pictures shown in [10, Figure 2] in which the rounding process clearly changes the combinatorial structure of the set of lines.

Rounding lines is much more difficult if the rounding must preserve some more global structure imposed on the set of lines. For example, Figure 1 shows a quadrilateral and a triangle before and after a rotation and rounding of individual lines. The rounding has clearly altered the global structure of the set of lines: before rotation and rounding, point **C** is contained inside the quadrilateral, and after, **C** is outside the quadrilateral. Such a change could cause failure of an algorithm that assumed that the triangle is surrounded by the quadrilateral, such as might be the case in a solid modeler with the figure representing a quadrilateral face of a polyhedron with a triangular hole.

Some of the structure of geometric objects can be preserved by using the *object reconstruction* approach [7]: if the structure of the set of geometric objects is defined in terms of operations on geometric primitives (e.g., lines, planes, solids), then the primitives can be rounded individually and the structure reimposed. For example, suppose that a simple polygon (a polygon is simple if non-consecutive edges do not share points) is represented by a constructive solid geometry (CSG) tree of half-planes, that is, as a boolean combination of half-planes. (Any simple polygon can be represented in this manner [2].) The polygon could then be rounded by rounding each half-plane boundary (line) individually and then re-evaluating the CSG tree to obtain the rounded polygon. Although this sequence at least ensures that the rounded polygon is also a simple polygon, it need not preserve other aspects of the polygon's structure, such as the number of sides.

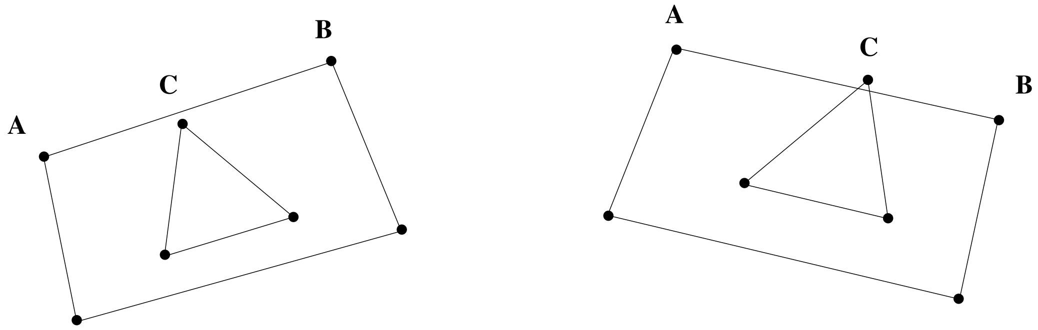

Figure 1: Result of Finite Precision Rotation

Another approach to avoiding precision growth in a sequence of operations is to compose the sequence of operations, round the composed operation, and then apply the rounded, composed operation. Again, consider rotations as an example. Let $\mathrm{rat}(A)$ denote a rational rotation that approximates the rotation A. If we are to compute an approximation to $A_r A_{r-1} A_{r-2} \cdots A_1 \mathbf{p}$, where $\mathbf{p}$ is some point, we should use $\mathrm{rat}(A_r A_{r-1} A_{r-2} \cdots A_1)\mathbf{p}$ instead of $\mathrm{rat}(A_r)\mathrm{rat}(A_{r-1})\mathrm{rat}(A_{r-2}) \cdots \mathrm{rat}(A_1)\mathbf{p}$, thus reducing the precision required from $\mathrm{O}(rP)$ to $\mathrm{O}(P)$. This is not always possible. For example, if $\mathbf{p}$ and $\mathbf{q}$ are vertices of different polygons, we might apply rotation A to $\mathbf{p}$ and B to $\mathbf{q}$. After taking the union of the two polygons, we then apply rotation C to it. Replacing $\mathrm{rat}(C)\mathrm{rat}(A)\mathbf{p}$ and $\mathrm{rat}(C)\mathrm{rat}(B)\mathbf{q}$ by $\mathrm{rat}(CA)\mathbf{p}$ and $\mathrm{rat}(CB)\mathbf{q}$ may change the structure of the resulting polygon.

All three of the approaches we have discussed avoid precision growth by rounding, but provide no guarantees that any notion of structure among a set of geometric objects is preserved. In this paper, we consider the problem of rounding sets of polygons or polyhedra while preserving a certain notion of combinatorial structure (defined below). Informally, the problem can be formulated as follows. Assume that a polygon is represented by a set of real numbers, called *coordinates*, and let $\mathcal{S}$ be a set of polygons. We would like to round the coordinates of the polygons of $\mathcal{S}$ to obtain a set of polygons $\mathcal{S}'$ that have the same combinatorial structure as $\mathcal{S}$, but with coordinates that can be represented with P bits of precision and are close to the corresponding coordinates of $\mathcal{S}$. By close, we mean that each rounded coordinate must be accurate to $P-K$ bits, for some integer K, $0 \leq K < P$, that is, that the magnitude of the difference between a coordinate and its corresponding rounded coordinate must be less than 2^{K-P}. If it is possible to do this, we call $\mathcal{S}'$ a (P, K)-approximation to $\mathcal{S}$.

Now let us discuss a practical scenario in which the ability to find (P, K)-approximations to sets of polygons would be useful. Suppose one starts with a set of P-bit polygons (i.e., polygons with P bit coordinates) and applies some operation to them (possibly, but not necessarily, a rigid motion) with the output being a set of M-bit polygons, where $M > P$. One then wishes to round the output polygons back to a set of P-bit polygons while preserving their combinatorial structure. Clearly, one can't just truncate coordinates to P-bits and expect to preserve the combinatorial structure. So, we are willing to sacrifice a small amount of geometric accuracy in exchange for preserving the structure. Namely, we seek a set of P-bit polygons with the most significant $P-K$ bits correct and the same combinatorial structure, in other words, a (P, K)-approximation to the set of M-bit polygons.

Our main result is that determining the existence of a (P, K)-approximation is NP-complete. This result is stated formally and its applicability and relation to other work discussed in Section 2. The result is proved in Section 3 by reducing the problem of three-coloring a planar graph of degree four to the problem of finding a (P, K)-approximation to a set of simple polygons. Section 4 generalizes from two-dimensional polygons to three-dimensional polyhedra. Since our result says that simultaneously rounding and preserving combinatorial structure is hard, in Section 5 we discuss polynomial-time techniques for generating compact representations that simultaneously round and preserve *nearly* the same combinatorial structure.

2 Rounding is Hard

In this section we show that the problem of determining whether a collection of simple polygons has a (P, K)-approximation is NP-complete. This result generalizes easily to polyhedra. We first define the problem precisely and state the result and then discuss the applicability of the result in practice and its relation to other work in the area of coordinate representations.

2.1 Combinatorial Structure

A *simple polygon* is a list of distinct lines $\mathbf{L}_1, \mathbf{L}_2, \mathbf{L}_3, \ldots, \mathbf{L}_k,\ k \geq 3$, such that the polygon with vertices $\mathbf{L}_1 \cap \mathbf{L}_2$, $\mathbf{L}_2 \cap \mathbf{L}_3$, …, $\mathbf{L}_{k-1} \cap \mathbf{L}_k$, $\mathbf{L}_k \cap \mathbf{L}_1$ is not self-intersecting and is oriented counter-clockwise. A *set of simple polygons* is

- a list of n distinct lines $\mathbf{L}_1, \mathbf{L}_2, \mathbf{L}_3, \ldots, \mathbf{L}_n$, $n \geq 3$;
- a list of $m+1$ integers $0 = n_0 < n_1 < n_2 < \cdots < n_m = n$ such that for $i = 1, 2, \ldots, m$, the sublist of lines numbered from $n_{i-1}+1$ to n_i forms a simple polygon $\mathcal{P}_i$;
- a forest of trees on the values $1, \ldots, m$ such that j is a descendant of i in some tree if and only if polygon $\mathcal{P}_j$ lies inside polygon $\mathcal{P}_i$.

Two sets of simple polygons S_1 and S_2 have the same *combinatorial structure* if and only if there is a one-to-one correspondence between the simple polygons of S_1 and the simple polygons of S_2 such that corresponding simple polygons have the same number of lines and such that the forests are isomorphic under the correspondence. Intuitively, if two sets of simple polygons have the same combinatorial structure, corresponding polygons in the two sets have the same number of edges and the polygons in one set are nested in each other in the same manner as the corresponding polygons in the other set. For example, the two sets of polygons shown in Figure 1 do not have the same combinatorial structure because the triangle is contained in the quadrilateral in one set and not in the other set.

2.2 (P, K)-Approximation

Assume P and K are positive integers such that $0 \leq K < P$, and define $\epsilon = 2^{-P}$ and $\eta = 2^{K-P}$. The set $\mathbf{BF}_P$ of *P-bit binary fractions* (or more simply *P-bit numbers*) is defined,

$$\mathbf{BF}_P = \{2^{-P}q | q \text{ is an integer and } -2^P \leq q \leq 2^P\}.$$

This is the set of numbers in the range $[-1, 1]$ that terminate within P bits of the "binary point." A *line* is defined by the equation $ax + by = c$, where a, b, and c are real. A *P-bit line* has coefficients $a, b, c \in \mathbf{BF}_P$.

A (P, K)-*approximation* to a real number a is a P-bit number a' satisfying, $|a' - a| < \eta$. It follows that a and a' have the same $P - K$ most significant bits. A line $a'x + b'y = c'$ is a (P, K)-*approximation* to a line $ax + by = c$ if a', b', and c' are (P, K)-approximations to a, b, and c, respectively. A simple polygon $\mathcal{P}'$ is a (P, K)-*approximation* to a simple polygon $\mathcal{P}$ if it has the same number of lines and if each line of $\mathcal{P}'$ is a (P, K)-approximation to the corresponding line in $\mathcal{P}$. A set of simple polygons S' is a (P, K)-*approximation* to a set S if S' has the same combinatorial structure as S and if each simple polygon of S' is a (P, K)-approximation to the corresponding polygon of S.

2.3 *NP*-Completeness

Theorem 1 *The language of sets of simple polygons with at least one (P, K)-approximation is NP-complete.*

This theorem is proved in Section 3 and generalized to polyhedra in Section 4.

2.4 Applications of Theorem 1

It is important to understand what Theorem 1 implies with regard to practical applications. Suppose we wish to rotate (or apply some other Euclidean transformation to) a polygon or polyhedron. We start with a P-bit polygon, rotate it exactly using rational arithmetic, and then round the higher precision result to a nearby (P, K)-approximation. The theorem does not necessarily imply that this rounding step will be difficult because, in this situation, we are restricting the input to those sets of simple polygons *that result from the Euclidean transformation of P-bit polygons.* It may be that this is an easier problem than the more general situation covered in the theorem, in which the set of input polygons is not restricted. Nevertheless, the theorem does imply that any procedure we might be able to devise for rounding restricted sets of polygons (i.e., those that result from Euclidean transformation of P-bit polygons) must exploit properties of the Euclidean transformation. There can be no general polynomial time procedure for rounding polygons (unless, of course, the $P \neq NP$ conjecture is false).

Unfortunately, a transformation-dependent rounding procedure might be hard to find for the following reason. The distance between closest *representable* points varies greatly from one part of the plane to another, where we define a point to be representable if it is the intersection of two P-bit lines. Thus even a pure translation might cause problems if it moves a cluster of vertices with representable locations to a region of the plane of low "density." As we shall see in Section 3.1.1, the proof of Theorem 1 depends on the existence of dense objects that must expand greatly when approximated. A sequence of transformations and set operations on polygons might lead to the construction of such a dense object, and the next transformation might move this object to a low-density region of the plane where it would be forced to expand.

To see how much the distance between closest representable points can vary, consider first the origin. Only lines of the form $ax + by = 0$ pass through the origin. Every other P-bit line $ax + by = c$ with $c \neq 0$ lies at least

$$\frac{|c|}{\sqrt{a^2 + b^2}} \geq 2^{-P}$$

distant. Therefore, the origin is the only representable point in a circle of radius 2^{-P}. Now consider nine integers, A_i, B_i, C_i, $i = 1, 2, 3$, with magnitude approximately equal to but not exceeding 2^P and such that:

$$\Delta_{123} = \begin{vmatrix} A_1 & B_1 & C_1 \\ A_2 & B_2 & C_2 \\ A_3 & B_3 & C_3 \end{vmatrix} = 1.$$

Such values may be rare, but they exist, for the points $\langle A_i, B_i, C_i \rangle$, $i = 1, 2, 3$, form a basis for the integer lattice in three dimensions [3, Section 13.9] and there are an infinite number of such bases. It can be shown that the three P-bit lines $2^{-P} A_i x + 2^{-P} B_i y = 2^{-P} C_i$, $i = 1, 2, 3$, form a triangle of diameter roughly 2^{-3P}. In particular, the distance from the intersection of the first two lines ($i = 1, 2$) to the third line (i=3) is

$$2^{-P} \Delta_{123} \begin{vmatrix} A_1 & B_1 \\ A_2 & B_2 \end{vmatrix}^{-1}.$$

Assuming that A_1, B_1, A_2, B_2 have the appropriate signs, this distance has magnitude close to 2^{-3P}. Thus, the minimum distance between representable points can vary from 2^{-3P} to 2^{-P}. There is clearly considerable unevenness in the distribution of representable points.

2.5 Relation to Other Work

Before proceeding to the proof, let us compare this result to other work. Mnev [9] and Goodman, Pollack and Sturmfels [6] examine the problem of finding coordinate representations for order types. An *order type* is

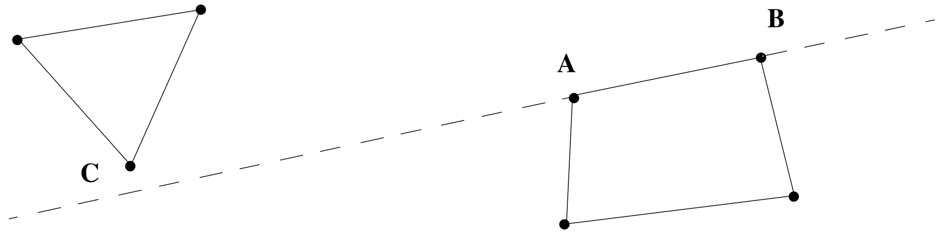

Figure 2: Order Type vs. Combinatorial Structure

an assignment of an orientation for every triple of points **A**, **B**, and **C**: the orientation is *positive* if triangle **ABC** is counter-clockwise, *negative* if **ABC** is clockwise, and *zero* if **A**, **B**, and **C** are collinear. Mnev shows that determining whether there exists a set of points with a given order type is equivalent to the existential theory of the reals. Goodman et al. show that even if such a set of points exists, a coordinate representation may require storage exponential in the number of points.

In contrast to the work of Mnev and of Goodman et al., our work considers lines instead of points, uses the notion of combinatorial structure instead of order type, and seeks a (P, K)-approximation to some given representation instead of seeking any coordinate representation of a combinatorial structure. These differences are motivated by practical considerations.

We consider lines instead of points because we wish to generalize to polyhedra, which are commonly represented by the plane equations of their faces. By well-known duality relationships [4], there is a direct connection between results on lines and results on points, so this difference is of little consequence.

On the other hand, the concept of combinatorial structure of a set of simple polygons is less stringent than the order type of points. For example, in Figure 2, point **C** could be moved to the other side of line **AB** without changing the combinatorial structure. The order type would be changed in this case because the orientation of triangle **ABC** would change from positive to negative. We believe that our definition of combinatorial equivalence is more natural for most practical problems.

One way to pose the problem we are considering is: Given a combinatorial structure for a set of simple polygons, find a coordinate representation for the polygons subject to the constraint that the coordinate representation is a (P, K)-approximation to some specified representation. This constraint is imposed because in practice rounding must both preserve combinatorial structure and incorporate some notion of "nearness" to the un-rounded object. Indeed, rounding would be easy without this constraint, for it can be shown easily that a combinatorial structure has a coordinate representation if and only if each polygon appears exactly once in the forest and, furthermore, that an $O(\log n)$-bit coordinate representation can be found in linear time.

Finally, we note that, like the results of Mnev and of Goodman et al., determining the existence of a (P, K)-approximation is difficult in two or more dimensions and easy in one dimension. The one-dimensional problem, generating a (P, K)-approximation to a forest of nested closed intervals, can be solved in linear time.

3 Proof

To prove Theorem 1, we must show that the problem of finding a (P, K)-approximation to a set of simple polygons is contained in NP and that it is NP-hard. Establishing that the problem is in NP is easy. Given a set of simple polygons with n distinct lines, there are up to 2^{3nK} possible (P, K)-approximations to the set, each of size $O(nP)$ bits. To prove that a particular set of simple polygons has a (P, K)-approximation, we non-deterministically generate one of the approximations and check it in polynomial time. To check an approximation, we first verify in time $O(n)$ that each coordinate is indeed a (P, K)-approximation of the corresponding coordinate in the original set of polygons. Next in time $O(n \log n)$ we verify that the set of polygons is simple and determine the nesting relationships among the polygons in the approximation (using a standard sweep-line algorithm [1]). Since the correspondence between the lines in the original set of simple polygons and the lines in a (P, K)-approximation to it is known, it is then only necessary to test that the two forests are isomorphic. This can be done in $O(n)$ time. Therefore, the problem is in NP.

We show that the problem is NP-hard, thus completing the proof of Theorem 1, by reducing the NP-complete problem of three-coloring planar graphs having no vertex degree greater than four [5, Section A1.1] to the problem of finding a (P, K)-approximation to a set of simple polygons. For any such graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, the reduction consists of two steps:

1. Embed $\mathcal{G}$ in an orthogonal grid, with graph vertices placed at grid vertices and graph edges following grid edges. Using the algorithm of Tamassia and Tollis [11], we can embed $\mathcal{G}$ in a cV by cV grid in $O(V)$ time, where c is a constant and $V = |\mathcal{V}|$.
2. Construct in time polynomial in V a set of simple polygons S such that there exists a (P, K)-approximation S' of S if and only if $\mathcal{G}$ can be three-colored.

The polygons constructed in the second step are instances of a small number of components. To help explain the construction, we group the components into levels in a manner analogous to the way components are organized in a computer chip. We shall use four levels, with a number of components defined at each level:

DEVICE: sponge, slider, adder;

SSI (small scale integration): transmission line, splitter-inverter, AND gate;

MSI (medium scale integration): transmission line crossing;

LSI (large scale integration): graph.

The purpose of embedding $\mathcal{G}$ in a grid is to simplify the "chip wiring."

A component at any level is characterized by its *terminals*, its *footprint*, and its input-output behavior. Each component has a set of zero or more terminals that are used in integrating components; the terminals are not part of the set of simple polygons, but are points that have a specified geometric relationship to the polygons. A terminal is said to be *covered* if it is contained in the interior of a polygon; otherwise it is *uncovered*. If a (P, K)-approximation forces a terminal to become uncovered, we will say that the approximation is *pushing* on that terminal of the component; conversely, if a (P, K)-approximation forces a component to cover a terminal, we will say that the component *pushes* on the terminal.

The set of simple polygons S will be constructed so that every (P, K)-approximation of S will cover some terminals and leave others uncovered; the coloring of the simulated graph $\mathcal{G}$ is represented by the covered/uncovered state of certain terminals. We shall describe the construction bottom-up, presenting at each level the "data sheets" and constructions for the components at that level.

3.1 Device Level

At the lowest level, the graph implementation consists of three types of collections of simple polygons: sponges, sliders, and adders.

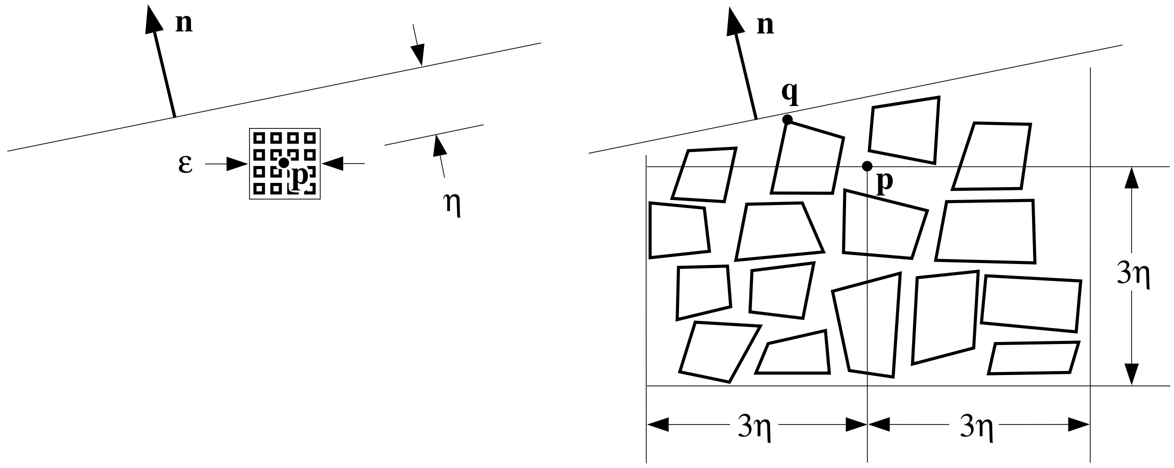

Figure 3: A Sponge and its (P, K)-Approximation

3.1.1 Sponges

Intuitively, a sponge is a tiny object that expands when approximated. See Figure 3. Let $\mathbf{p}$ be a point and $\mathbf{n}$ a non-zero vector such that $|\mathbf{n}_y| \geq |\mathbf{n}_x|$. A *horizontal* $(\mathbf{p}, \mathbf{n})$*-sponge* is a set of simple polygons with the following properties:

- The sponge's polygons lie in a square of side ϵ aligned with the coordinate axes and centered at $\mathbf{p}$.
- In every (P, K)-approximation to the sponge, there exists a polygon vertex $\mathbf{q}$ at least $\frac{(\eta-\epsilon)|\mathbf{n}_y|}{|\mathbf{n}|}$ distant from $\mathbf{p}$ along the direction of the vector $\mathbf{n}$. Expressed analytically, $(\mathbf{q} - \mathbf{p}) \cdot \mathbf{n} \geq (\eta - \epsilon)|\mathbf{n}_y|$.
- Every (P, K)-approximation to the sponge lies in a square of width 6η aligned with the coordinate axes and centered at $\mathbf{p}$.
- There exists at least one (P, K)-approximation to the sponge that extends no farther than $\frac{(\eta+\epsilon)|\mathbf{n}_y|}{|\mathbf{n}|}$ from $\mathbf{p}$ in the direction $\mathbf{n}$. That is, for all vertices $\mathbf{q}$ in this approximation, $(\mathbf{q} - \mathbf{p}) \cdot \mathbf{n} < (\eta + \epsilon)|\mathbf{n}_y|$

A *vertical* $(\mathbf{p}, \mathbf{n})$*-sponge* is defined analogously for $|\mathbf{n}_x| \geq |\mathbf{n}_y|$. When we refer to a $(\mathbf{p}, \mathbf{n})$*-sponge*, we mean either a horizontal or vertical $(\mathbf{p}, \mathbf{n})$-sponge, as appropriate.

A $(\mathbf{p}, \mathbf{n})$-sponge is implemented by a set of squares with sides parallel to the coordinate axes. We now give an algorithm for constructing a sponge for the case $\mathbf{n}_y \geq \mathbf{n}_x \geq 0$; constructions for the other cases can be obtained straightforwardly by reflections and rotations by multiples of $\pi/2$. Let $\xi = 2^{-(P+3K+4)}$ and define the sets of horizontal and vertical lines:

$$\begin{array}{ll} \mathbf{hl}_i\colon & y = \mathbf{p}_y + i\xi, \quad i = 0, 1, \ldots, 2^{3K+3}; \\ \mathbf{vl}_j\colon & x = \mathbf{p}_x + j\xi, \quad j = 0, 1, \ldots, 2^{3K+3}. \end{array}$$

Because ξ is so small, the point $\mathbf{p}$ lies within ϵ of each of these lines, and each line $\mathbf{hl}_i$ has the same set of (P, K)-approximations as the line $y = \mathbf{p}_y$, and each line $\mathbf{vl}_j$ has the same set of (P, K)-approximations as the line $x = \mathbf{p}_x$. Since the K least significant bits of all three coefficients can be changed, $x = \mathbf{p}_x$ and $y = \mathbf{p}_y$ each have 2^{3K} (P, K)-approximations.

The following algorithm generates a $(\mathbf{p},\mathbf{n})$-sponge.

```
sponge-set = ∅
for i = 0 to 2^{3K+1}
    for j = 0 to 2^{3K+1}
        /* (η − ε)-invariant */
        add the square hl_{2i} vl_{2j} hl_{2i+1} vl_{2j+1} to sponge-set
        /* (η + ε)-invariant */
        if sponge-set is a (p, n)-sponge, return it
```

The set *sponge-set* can contain no more than 2^{6K+2} squares. Each square has no more than 2^{12K} (P,K)-approximations (because each side has no more than 2^{3K} approximations). Therefore, there are no more than $(2^{12K})^{(2^{6K+2})}$ (P,K)-approximations to the squares in *sponge-set*. Since K is a constant, we can test in (a somewhat daunting) constant time whether the current *sponge-set* is a $(\mathbf{p},\mathbf{n})$-sponge.

We prove correctness using two invariants:

$(\eta-\epsilon)$-invariant: there exists at least one (P,K)-approximation to *sponge-set* that does not extend farther than $\frac{(\eta-\epsilon)|\mathbf{n}_y|}{|\mathbf{n}|}$ in the direction of $\mathbf{n}$ (for all vertices $\mathbf{q}$ in the approximation, $(\mathbf{q}-\mathbf{p})\cdot\mathbf{n} < (\eta-\epsilon)|\mathbf{n}_y|$);

$(\eta+\epsilon)$-invariant: there exists at least one (P,K)-approximation to *sponge-set* that extends at least $\frac{(\eta-\epsilon)|\mathbf{n}_y|}{|\mathbf{n}|}$, but no farther than $\frac{(\eta+\epsilon)|\mathbf{n}_y|}{|\mathbf{n}|}$, in the direction of $\mathbf{n}$.

The proof of correctness is as follows.

- The first invariant clearly holds for the empty set.
- If the first invariant holds for *sponge-set*, then the second invariant holds when square $\mathbf{hl}_{2i}\mathbf{vl}_{2j}\mathbf{hl}_{2i+1}\mathbf{vl}_{2j+1}$ is added to *sponge-set*. The (P,K)-approximation implied by the first invariant does not intersect the square bounded by the lines $x=\mathbf{p}_x$, $y=\mathbf{p}_y+\eta-\epsilon$, $x=\mathbf{p}_x+\epsilon$, and $y=\mathbf{p}_y+\eta$, because this square lies at least $\frac{(\eta-\epsilon)|\mathbf{n}_y|}{|\mathbf{n}|}$ in the direction of $\mathbf{n}$ and no farther than $\frac{(\eta+\epsilon)|\mathbf{n}_y|}{|\mathbf{n}|}$. But this square is a (P,K)-approximation to the square $\mathbf{hl}_{2i}\mathbf{vl}_{2j}\mathbf{hl}_{2i+1}\mathbf{vl}_{2j+1}$, so the second invariant is satisfied.
- If the *sponge-set* satisfies the second invariant but fails to be a $(\mathbf{p},\mathbf{n})$-sponge, it must be true that *not all* (P,K)-approximations have a vertex lying $\frac{(\eta-\epsilon)|\mathbf{n}_y|}{|\mathbf{n}|}$ in the direction of $\mathbf{n}$. In other words, the first invariant holds.

The proof of termination is as follows. Each time around the inner loop, the second invariant implies that there exists at least one (P,K)-approximation to *sponge-set*. If the algorithm terminates without concluding that *sponge-set* is a $(\mathbf{p},\mathbf{n})$-sponge, then there are 2^{6K+2} squares in *sponge-set*. But this set is too large to have a valid (P,K)-approximation. To see why, consider that there are only 2^{3K} possible approximations each for the set of horizontal and for the set of vertical lines. Therefore only, 2^{6K} points in the plane are eligible to be vertices of polygons in the (P,K)-approximation. A set of 2^{6K+2} disjoint polygons needs more vertices than that.

It remains for us to verify that each point of every (P,K)-approximation to the sponge lies within the square of width 6η centered at $\mathbf{p}$. Let $ax+by=c$ be a line where $a^2+b^2=1$. Let $a'x+b'y=c'$ be a (P,K)-approximation to that line, and let $\langle X,Y\rangle$ be a point on the latter line inside the unit square centered at the origin. How far can this point lie from the original line? We know that,

$$|aX+bY-c| \le |a'X+b'Y-c'| + |(a-a')X| + |(b-b')Y| + |(c-c')| \le 0+\eta+\eta+\eta = 3\eta.$$

Since each vertex in the (P,K)-approximation to the sponge lies on a (P,K)-approximation to a vertical line and a horizontal line through $\mathbf{p}$, each vertex lies within the desired square.

3.1.2 Sliders

This section defines a *slider*, which is a component that can transmit "information" from a terminal $\mathbf{p}_1$ to another terminal $\mathbf{p}_2$. Sliders will be used below to construct *transmission lines*, which transmit colors among vertices. As depicted in Figure 4 (Figure 4 is foreshortened),[1] a $\mathbf{p}_1\mathbf{p}_2$-*slider* is a hexagon with two sponges inside. The hexagon has the form of a long shaft with a right triangular "speartip" at each end. The length of the shaft is arbitrary, and it is chosen so that the hexagon has the following properties:

- the slider covers neither terminal $\mathbf{p}_1$ nor $\mathbf{p}_2$ and lies at least η distant from each terminal;
- every (P, K)-approximation to the slider covers at least one of the terminals;
- there exists at least one (P, K)-approximation that covers $\mathbf{p}_1$ but not $\mathbf{p}_2$, and at least one that covers $\mathbf{p}_2$ but not $\mathbf{p}_1$.

In other words, if we push on one terminal, the slider pushes on the other terminal.

Intuitively, the slider is a computational device that amplifies the expansion of the slider's sponges under (P, K)-approximation. Amplification is achieved through the use of speartips with a high aspect ratio, one part in eight in this case (Figure 4 is foreshortened). Displacing either the hypotenuse or the longer leg parallel to itself by η has the effect of displacing their intersection by 8η in the x-direction. Initially, the shorter legs have length about 4η and the longer legs, about 32η. In any (P, K)-approximation, the sponges force each hypotenuse to be displaced parallel to itself by about η, causing the spearheads to extend about an additional 8η to cover the terminals as shown in Figure 5. However, the shaft can be displaced upward by η, restoring the right spearhead to its original length and uncovering $\mathbf{p}_2$, as shown in Figure 6, or the shaft can be displaced downward to uncover $\mathbf{p}_1$. Because of the length chosen for the shaft, it is not possible to uncover both terminals at the same time.

Let us first define the slider for $\mathbf{p}_1 = \langle -\lambda, 0\rangle$ and $\mathbf{p}_2 = \langle \lambda, 0\rangle$; generalization is straightforward. For simplicity, we shall assume that λ is an integral multiple of η. Since there must be room for both sponges to expand simultaneously in the horizontal direction by 3η, the tip length is about 32η, and the distance from the tip point to terminal must be at least η. Thus, λ must be at least 36η. To allow for the necessary space to position the sponges properly, we require that $\lambda \geq 38\eta$. The six lines, as labeled in Figure 4, have the following equations:

$$
\begin{array}{lrcccccl}
\text{line} & & & & & & & \\
(1) & 0 & x & + & -1 & y & = & \epsilon \\
(2) & \frac{1}{8} & x & + & 1 & y & = & \frac{\lambda-2\eta}{8} + \epsilon \\
(3) & -1 & x & + & 0 & y & = & -\lambda + 34\eta + \epsilon \\
(4) & 0 & x & + & 1 & y & = & \epsilon \\
(5) & -\frac{1}{8} & x & + & -1 & y & = & \frac{\lambda-2\eta}{8} + \epsilon \\
(6) & 1 & x & + & 0 & y & = & -\lambda + 34\eta + \epsilon
\end{array}
$$

The sponge in the uppermost corner of the hexagon is a $(\langle \lambda - 34\eta, 4\eta\rangle, \mathbf{n})$-sponge, and the one in the lowermost corner is a $(\langle -\lambda + 34\eta, -4\eta\rangle, -\mathbf{n})$-sponge, with $\mathbf{n} = \langle \frac{1}{8}, 1\rangle$.

Assuming that the lines and their (P, K)-approximations are parallel, we can easily verify that the slider has the desired properties. The sponges expand by η (actually $\sqrt{64/65}\eta$, the amount required to shift line (2) *vertically* by a distance η) in the desired direction and expand by no more than 3η horizontally and vertically. Because the spearheads have height 4η initially, the vertical expansion will not interfere with the motion of the shaft. Without loss of generality, we always choose the shaft to have length greater than 6η, and thus the horizontal displacement cannot eliminate the shaft.

We claim that any change in orientation of the lines has negligible effect for the length of sliders we will consider, a few hundred η at most. The magnitude of the slope of line (2) is $\frac{1}{8}/1$. Approximation can increase

[1] In this and all other figures, dimensions are considered to be accurate within a small multiple of ϵ. Thus $3\eta \pm 4\epsilon$ is labeled as 3η. The correct line equations are given in the text.

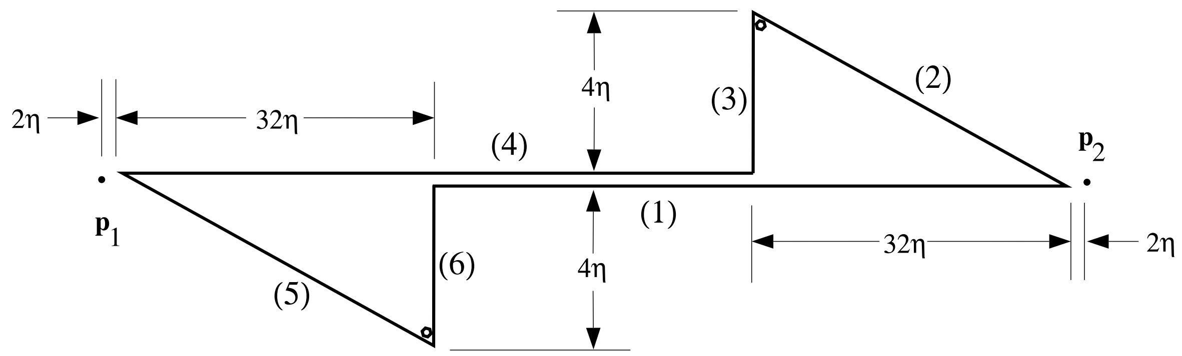

Figure 4: A Slider

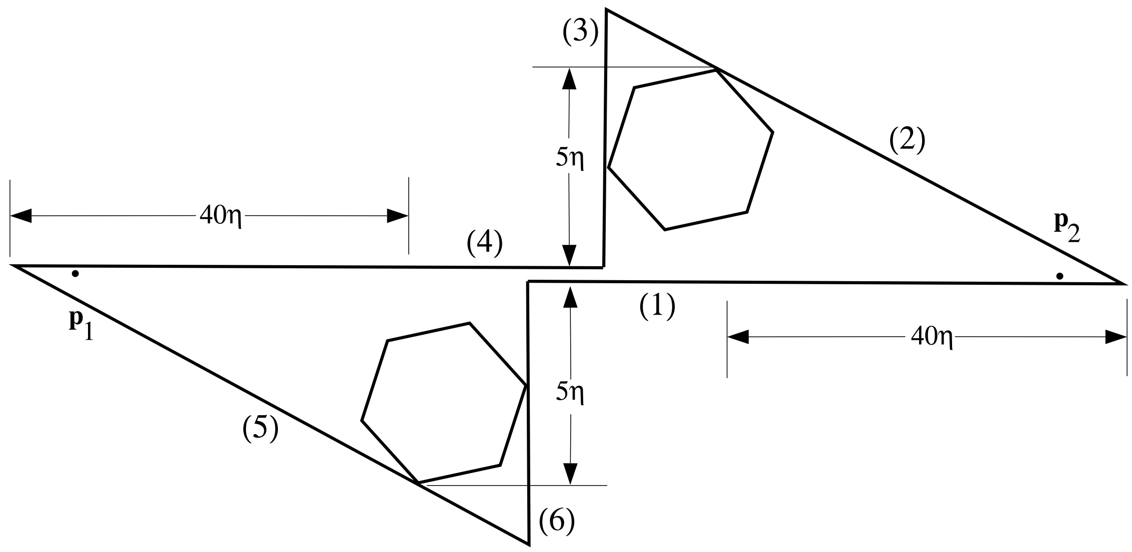

Figure 5: One Possible (P, K)-Approximation to a Slider

the slope magnitude to $(\frac{1}{8} + \eta)/(1 - \eta)$ at most, which is $\frac{1}{8}(1 + 9\eta + 9\eta^2 + \cdots)$. Neglecting terms of higher than first order, this change in slope corresponds to a change in orientation of $\frac{9}{8}\eta = 1.125\eta$ radians. Similar reasoning shows that the change in orientation of line (1) is bounded by η radians. Thus, for $\lambda = 1000\eta$, a change in the orientation of lines (1) and (2) moves the intersection of lines (1) and (2) by no more than $2.123\eta\lambda = 2125\eta^2$. For $\eta = 2^{-20}$, this error amounts to about 0.002η, which is negligible, as claimed.

In general, a slider can be positioned anywhere in the plane and oriented either horizontally or vertically. Translating the slider to be centered at a point $\langle X, Y\rangle$ simply requires changing each of its lines $ax + by = c$ to $ax + by = c + aX + bY$.

3.1.3 Adder

An *adder*, as depicted in Figure 7 (foreshortened), has four *input* terminals $\mathbf{u}_1$, $\mathbf{u}_2$, $\mathbf{v}_1$, and $\mathbf{v}_2$, and two *output* terminals $\mathbf{w}_2$ and $\mathbf{w}_4$. For a (P, K)-approximation to the adder, we say that its first input is zero if both $\mathbf{u}_1$ and $\mathbf{u}_2$ are covered, one if only $\mathbf{u}_2$ is covered, and two if neither $\mathbf{u}_1$ nor $\mathbf{u}_2$ is covered. The second input is defined analogously for $\mathbf{v}_1$ and $\mathbf{v}_2$. The adder has the property that if the sum of the inputs is at least two, $\mathbf{w}_2$ is covered, and if the sum is at least four, $\mathbf{w}_4$ is also covered.

An adder is implemented by a nine-sided polygon with four sponges. Let us first consider an adder centered

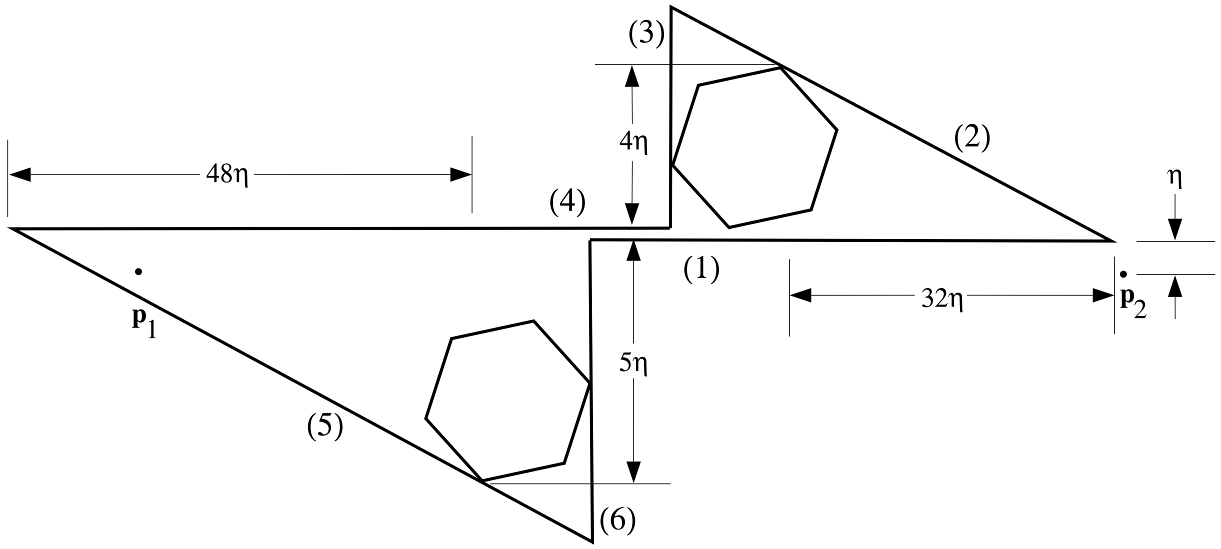

Figure 6: Another (P, K)-Approximation to a Slider

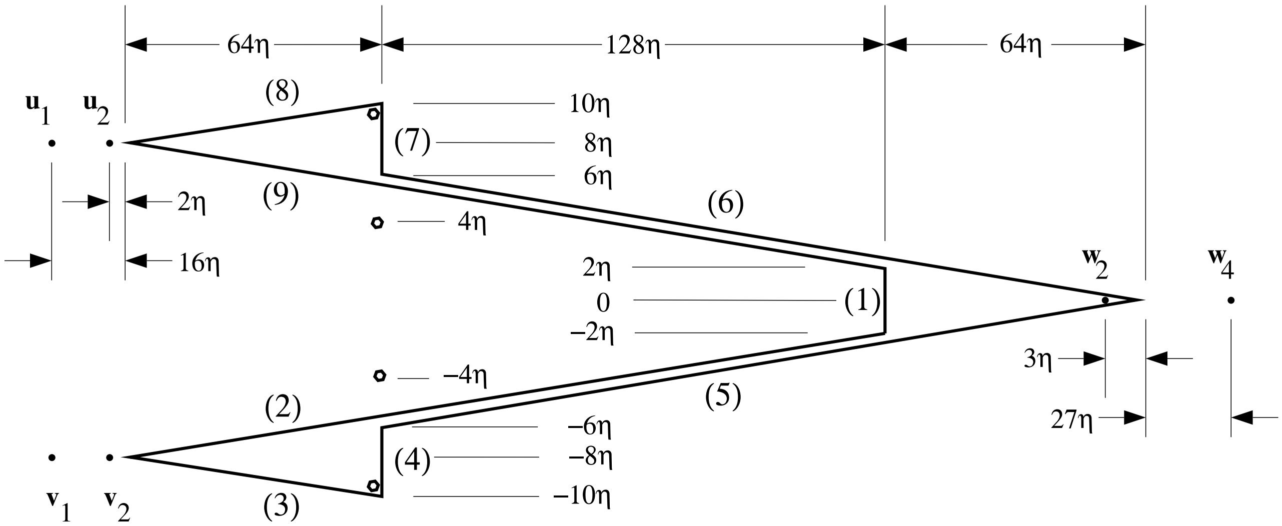

Figure 7: An Adder

at the origin. It is bounded by the lines,

$$
\begin{array}{lrcccccl}
\text{line} & & & & & & & \\
(1) & -1 & x & + & 0 & y & = & -64\eta \\
(2) & -\frac{1}{32} & x & + & 1 & y & = & -4\eta+\epsilon \\
(3) & -\frac{1}{32} & x & + & -1 & y & = & 12\eta+\epsilon \\
(4) & 1 & x & + & 0 & y & = & -64\eta+\epsilon \\
(5) & \frac{1}{32} & x & + & -1 & y & = & 4\eta+\epsilon \\
(6) & \frac{1}{32} & x & + & 1 & y & = & 4\eta+\epsilon \\
(7) & 1 & x & + & 0 & y & = & -64\eta+\epsilon \\
(8) & -\frac{1}{32} & x & + & 1 & y & = & 12\eta+\epsilon \\
(9) & -\frac{1}{32} & x & + & -1 & y & = & -4\eta+\epsilon
\end{array}
$$

Wedged in the uppermost and lowermost corners are two sponges—a $(\langle -64\eta, 10\eta\rangle, \langle -\frac{1}{32}, 1\rangle)$-sponge and a $(\langle -64\eta, -10\eta\rangle, \langle -\frac{1}{32}, -1\rangle)$-sponge, respectively—which displace lines (8) and (3) outwards and parallel to themselves by about η in each (P, K)-approximation. To keep lines (2), (5), (6), and (9) from moving more than η inwards, we add two more sponges: a $(\langle -64\eta, 4\eta\rangle, \langle \frac{1}{32}, 1\rangle)$-sponge and a $(\langle -64\eta, -4\eta\rangle, \langle \frac{1}{32}, -1\rangle)$-sponge. Now, suppose that we push on $\mathbf{u}_2$ (i.e., some other polygon covers $\mathbf{u}_2$). Since lines (6) and (9) are prevented from moving inwards, they must move outward so that the intersection of (the approximations of) lines (8) and (9) lies to the right of $\mathbf{u}_2$. This then causes the intersection of (the approximations of) lines (5) and (6) to move to the right of $\mathbf{w}_2$, pushing on $\mathbf{w}_2$. By analogous reasoning, if we also push on $\mathbf{v}_2$, line (5) will also be forced to move outward, moving the intersection of lines (5) and (6) still farther rightward, pushing on $\mathbf{w}_4$. Similar arguments can be used to show that the desired behavior can be obtained for the other combinations of inputs.

To make the adder perform as described, the terminals must be placed appropriately. This requires consideration both of the geometry of the adder itself and of the way it will be integrated into higher-level "circuits." We shall see in the next section that $\mathbf{u}_2$ and $\mathbf{v}_2$ must be placed sufficiently far to the left of the intersections of lines (8) and (9) and lines (2) and (3), respectively, so that they can be covered by slider tips; it suffices to shift $\mathbf{u}_2$ and $\mathbf{v}_2$ left by 2η. Terminals $\mathbf{u}_1$ and $\mathbf{v}_1$ are placed 16η to the left of the intersections of lines (8) and (9) and lines (2) and (3), respectively. To compensate for shifting $\mathbf{u}_2$ and $\mathbf{v}_2$, $\mathbf{w}_2$ is shifted 3η to the left of the intersection of lines (5) and (6). The coordinates of the control points are:

$$
\begin{array}{lcl}
\mathbf{u}_1 & = & \langle\ -144\eta,\ \ 8\eta\ \rangle, \\
\mathbf{u}_2 & = & \langle\ -130\eta,\ \ 8\eta\ \rangle, \\
\mathbf{v}_1 & = & \langle\ -144\eta,\ -8\eta\ \rangle, \\
\mathbf{v}_2 & = & \langle\ -130\eta,\ -8\eta\ \rangle, \\
\mathbf{w}_2 & = & \langle\ 125\eta,\ \ 0\ \rangle, \\
\mathbf{w}_4 & = & \langle\ 155\eta,\ \ 0\ \rangle.
\end{array}
$$

Centering the adder at some point other than the origin is accomplished as with sliders.

3.2 SSI Level

The device-level components described in Section 3.1 are integrated into higher level components by placing horizontal and vertical devices so that they share a common terminal. Since a terminal can be covered by only one device at a time (since polygons are not permitted to intersect), a terminal can have three logic values, denoted as 0, h, or v, depending on whether no device, a horizontal device, or a vertical device is covering it.

The top row of Figure 8 shows symbols used in SSI diagrams for sliders and adders. The output of each of these devices can be duplicated by placing extra terminals appropriately. This is illustrated on the bottom row of Figure 8: an extra terminal $\mathbf{p}_3$ is placed 3η to the right of $\mathbf{p}_2$ and an extra terminal is placed 3η to the left of each of the original control points $\mathbf{w}_2$ and $\mathbf{w}_4$. The specifications given in Sections 3.1.2 and 3.1.3 are designed to permit these extra terminals.

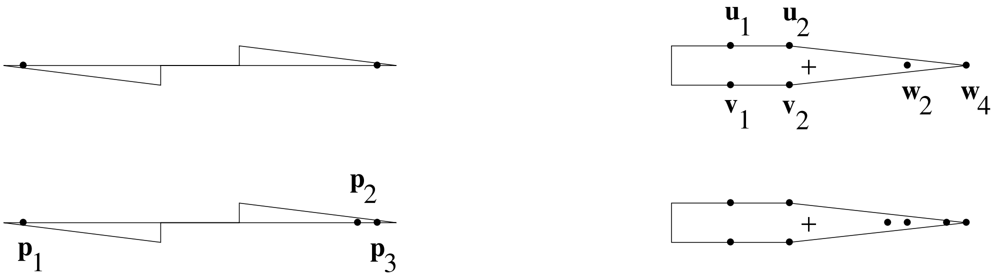

Figure 8: SSI Symbols: Slider and Adder

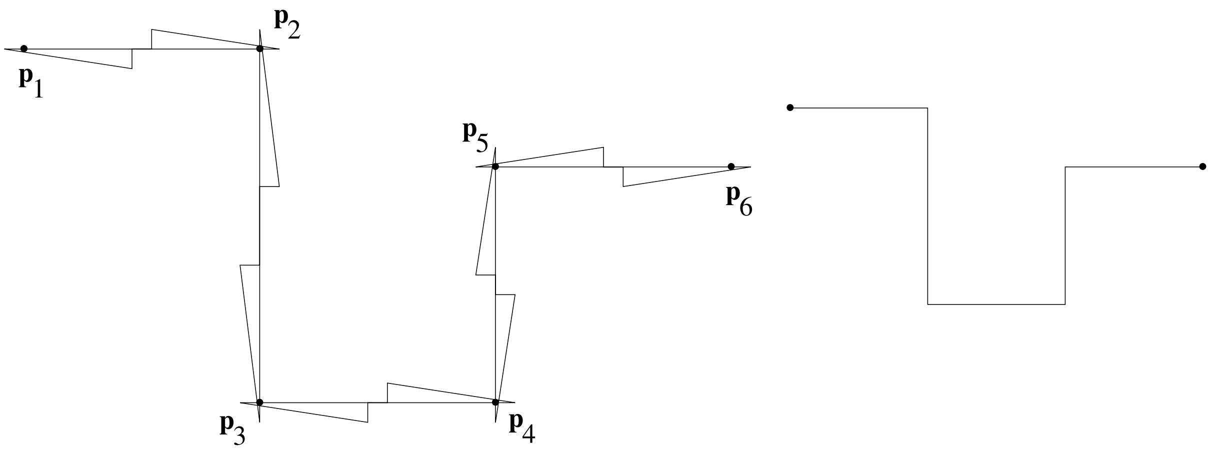

Figure 9: Transmission Line

3.2.1 Transmission Lines

Figure 9 depicts a *transmission line*, which consists of a string of horizontal and vertical sliders sharing common control points. If the value of $\mathbf{p}_1$ is 0, then $\mathbf{p}_2$ must have value h, $\mathbf{p}_3$ must have value v, $\mathbf{p}_4$ must have value h, and so on. Thus asserting a zero value for $\mathbf{p}_1$ transmits information over the line. Transmission lines have no "tensile strength," however; if $\mathbf{p}_1$ is h, we can conclude nothing about the other values; for example, $\mathbf{p}_2$ can take on any of the three logic values. The MSI symbol for a transmission line is a sequence of alternating horizontal and vertical line segments, like the ones shown on the right side of Figure 9.

3.2.2 AND Gates

Figure 10 illustrates the SSI implementation of an *AND gate* and its MSI symbol. Pushing on $\mathbf{p}_1$ and $\mathbf{p}_2$ causes the gate to push on $\mathbf{p}_3$. In fact, the gate is symmetric with respect to its three terminals: pushing on any two causes the gate to push on the third. In other words, any (P, K)-approximation of an AND gate must cover at least one and at most two of its terminals.

3.2.3 Splitter-Inverters

Figure 11 illustrates the SSI implementation of a *splitter* and its MSI symbol. Pushing on $\mathbf{p}_1$ causes this component to push on $\mathbf{p}_2$ and $\mathbf{p}_3$, thus duplicating its input.

If we think of $\mathbf{p}_2$ as the input, then pushing on $\mathbf{p}_2$ has the effect of freeing $\mathbf{p}_3$ to be pushed on. Since we cannot actually *pull* on a terminal, this is close as we can come to an *inverter*.

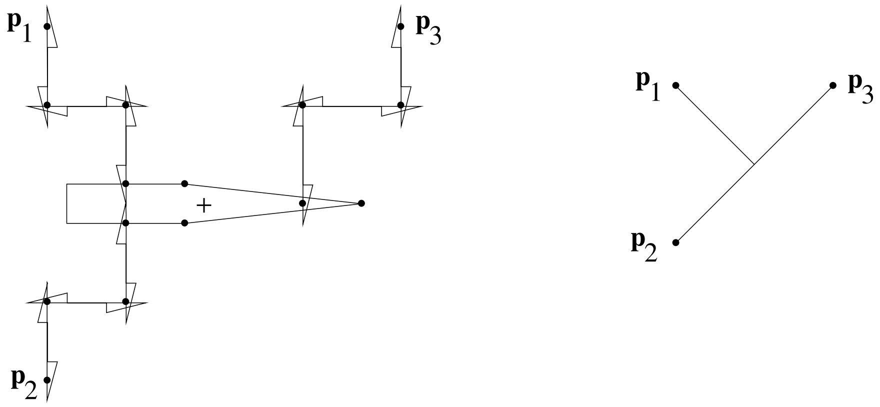

Figure 10: AND gate

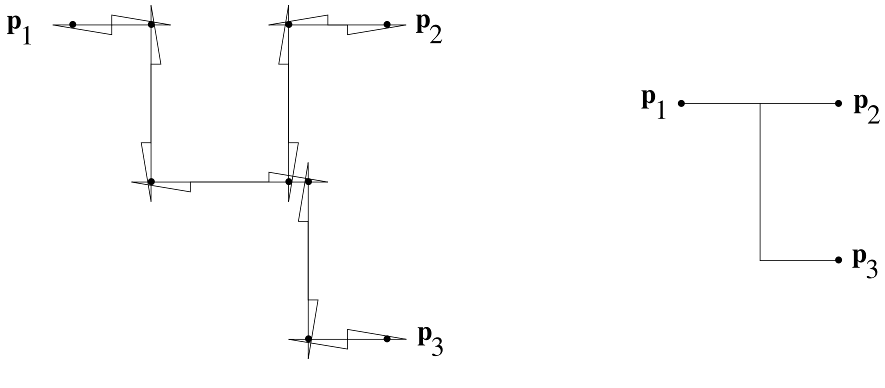

Figure 11: Splitter-Inverter

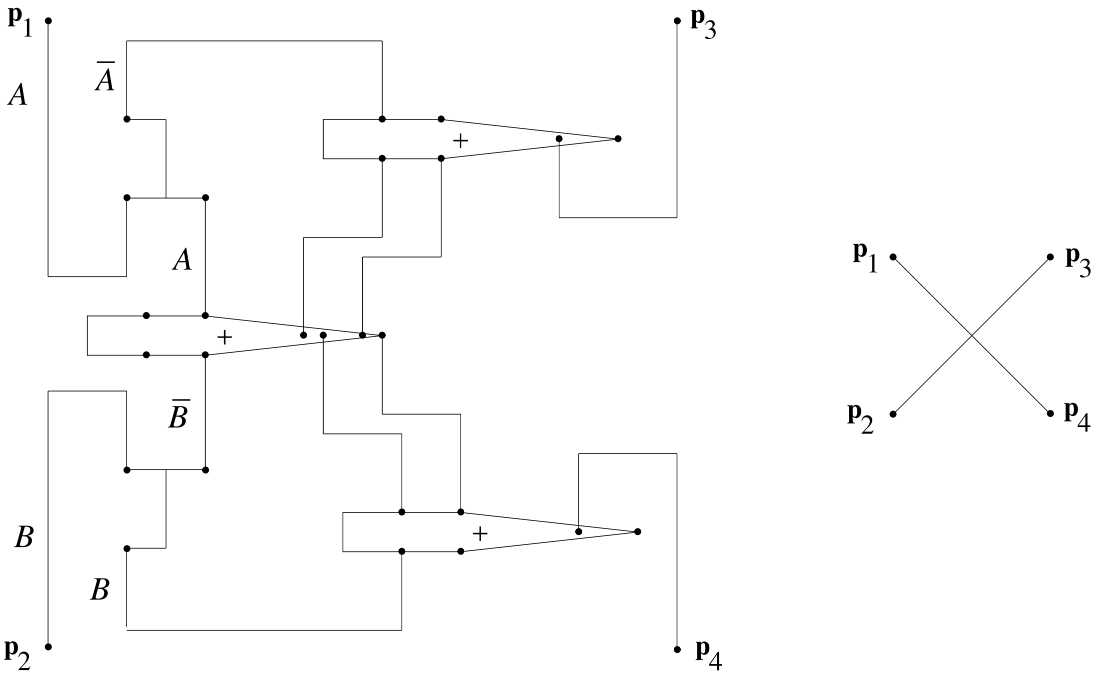

Figure 12: Crossing Two Transmission Lines

3.3 MSI: Transmission Line Crossing

Even though we will be implementing a planar graph at the LSI level, transmitting color information among vertices requires transmission lines to cross. Figure 12 illustrates the MSI implementation of a *transmission line crossing* and its LSI symbol. A crossing consists of three adders, two splitter-inverters, and a number of transmission lines. We now show that if we push on $\mathbf{p}_1$, the crossing pushes on $\mathbf{p}_4$; independently, if we push on $\mathbf{p}_2$, the crossing pushes on $\mathbf{p}_3$. Let A be the first input of the left adder, and let B be its second input. If we push on $\mathbf{p}_1$, $A = 1$; otherwise $A = 0$. Similarly for $\mathbf{p}_2$ and B. The left adder computes $A + B$: it pushes on $\mathbf{w}_2$ if either is one, and it pushes on $\mathbf{w}_4$ if both are one. The upper adder computes $\overline{A} + A + B = 1 - A + A + B = 1 + B$, which depends on B only. Similarly, the lower adder computes $1 + A$.

3.4 LSI: Implementing a Graph

Each vertex in $\mathcal{V}$ is represented by an AND gate surrounded by circuitry required to transmit its state to other vertices. It follows from the definition of the logical AND that an AND gate must cover at least one of its three terminals. Let us label its terminals R, B, and G. If the gate covers terminal R, the vertex is colored red; if it covers B but not R, its color is blue; else it covers only G, and it is green. Information about the color of a vertex is transmitted to neighboring vertices by three transmission lines. For example, if a vertex is blue, it pushes on the transmission line leading out of the B terminal. Since it is not possible to push on both ends of a transmission line simultaneously, none of the neighboring vertices can be blue.

Figure 13 illustrates the LSI implementation of two neighboring vertices of a graph. Nine splitters and nine crossings suffice to transmit information about a vertex's color in four directions (remember, this graph has degree four).

We now argue that P and K can be chosen so that the appropriate vertices and transmission lines can be constructed to simulate any graph $\mathcal{G}$. Recall that the algorithm of Tamassia and Tollis [11] can be used to embed $\mathcal{G}$ in a $cV \times cV$ grid, for some constant c. For some constants k_1 and k_2, the "circuitry" for a vertex takes no more space than $k_1\eta \times k_1\eta$, and the "circuitry" for a transmission line is no wider than

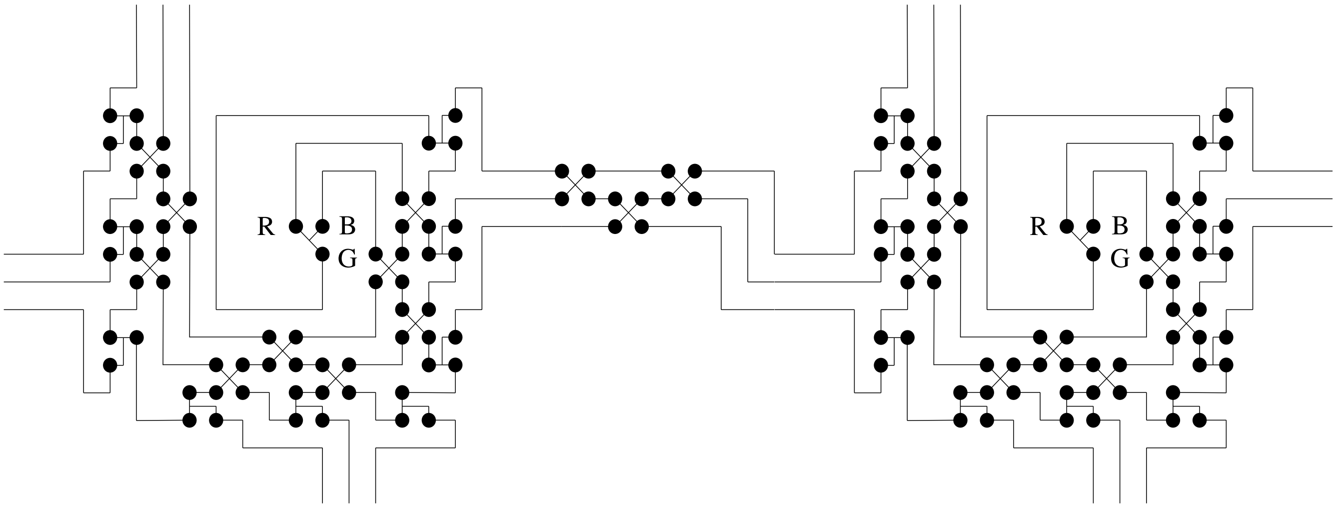

Figure 13: Implementation of Graph

$k_2\eta$. In total, the width of the embedding is $\max(k_1, k_2)\eta cV$, which must be less than unity. Therefore, $\eta = \min(2^{-20}, (\max(k_1, k_2)cV)^{-1})$, since, from Section 3.1.2, $\eta < 2^{-20}$. The value of ϵ must be small enough that the addition of ϵ to line coefficients has a negligible effect. Setting $\epsilon = 2^{-20}\eta$ easily suffices. This leads to $P = \lceil -\log \epsilon \rceil$ and $K = \lceil -\log \epsilon/\eta \rceil = 20$.

In summary, we point out that before any rounding, none of the terminals is covered by any slider. If a valid (P, K)-approximation exists, the set of covering choices for the slider terminals corresponds to a choice of colors. Thus, an approximation exists if and only if a coloring exists.

4 Generalization to Polyhedra

With suitable generalizations of the definitions, Theorem 1 also applies to sets of simple polyhedra. To see this, observe that if it did not, membership in the language of sets of simple polygons could be determined by "thickening" polygons into polyhedra in the z-direction.

In particular, we replace each line $ax + by = c$ by the *plane* $ax + by = c$. This replacement transforms each simple polygon into an infinite cylinder in the z-direction. To make each cylinder finite, we terminate it at the plane $z = 1$ and the plane $z = -1$. If a set of simple polygons has a (P, K)-approximation, then the resulting set of cylinders has a (P, K)-approximation: simply convert each polygon in the planar (P, K)-approximation into a cylinder. Conversely, if the set of cylinders has a (P, K)-approximation, then we can generate a (P, K)-approximation to the original set of polygons by taking the cross-section $z = 0$. Thus, the set of polygons has a (P, K)-approximation if and only if the corresponding set of cylinders has a (P, K)-approximation. Therefore, the polyhedral rounding problem is NP-hard.

To show that the polyhedral rounding problem is in NP, we have to be able to check a potential (P, K)-approximation in polynomial time. Even using naive methods, it requires no more that $O(n^3)$ time to verify that a set of polyhedra are simple and to determine the nesting relationship. Comparing the forest of nesting relationships with the original set of polyhedra can be done in $O(n)$ time.

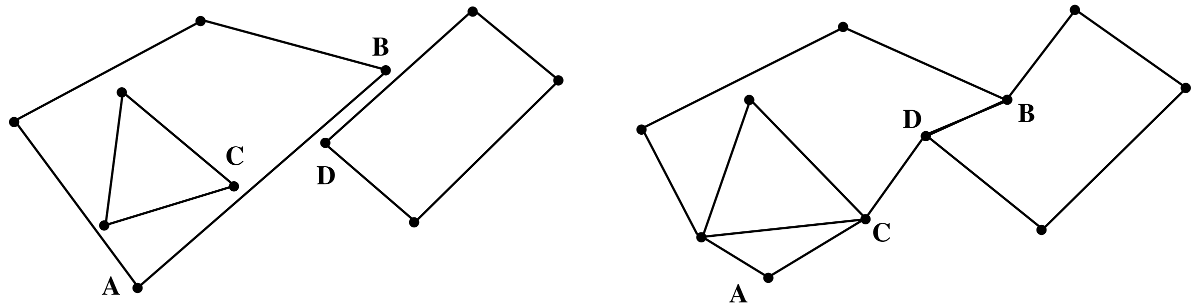

Figure 14: Polygon Rounding Based on Shortest Paths

5 Approximation Methods

Since finding a (P, K)-approximation to a set of simple polygons is NP-complete, it is important to consider approximate methods. This section describes several approaches. An obvious approach, which was discussed in Section 1, is to avoid the need to round a set of polygons. The object reconstruction technique rounds individual geometric elements independently and then computes a new combinatorial structure. The other approach discussed is to compose and then round the transformation, not the transformed set of polygons.

Probabilistic methods seem more promising. Clearly, the sets of simple polygons generated by the reduction, particularly the sponges, are not typical. If we could define a natural distribution on sets of simple polygons, we might create techniques that work well on this distribution.

Another good approximation, discussed elsewhere [8], is to replace each edge of a polygon with the shortest polygonal path that has nearly the same combinatorial relationship with every edge. By "nearly" we mean that some vertices may end up on a path that did not lie on an edge. This process is illustrated in Figure 14, in which edge **AB** becomes path **ACDB**.

6 Conclusion

We have shown that to round polygons and polyhedra while preserving combinatorial structure is a difficult problem. We have not shown, however, that rotation with limited precision growth is necessarily difficult, because this would require showing that the polygon used in the reduction can result from the rotation of a P-bit set of simple polygons. However, we have shown that one cannot solve the problem of rotations by solving the general problem of rounding. Any technique for rotation must either increase the number of bits required to represent each coefficient, or it must change the combinatorial structure, or it must exploit the fact that the output polygons or polyhedra result from a rotation. A technique based on the third approach may exist, but it will not generalize to other operations that require rounding.

References

[1] Jon L. Bentley and Thomas Ottmann. Algorithms for Reporting and Counting Geometric Intersections. *IEEE Transactions on Computing*, Vol. C29, 1230-1234, 1972.

[2] David Dobkin, Leonidas Guibas, John Hershberger, and Jack Snoeyink. An Efficient Algorithm for Finding the CSG Representation of a Simple Polygon. SIGGRAPH'88 Conference Proceedings, *Computer Graphics*, Vol. 22, No. 4, 31-40, August 1988.

[3] H.S.M. Coxeter. *Introduction to Geometry*, John Wiley & Sons, New York, 1969.

[4] Herbert Edelsbrunner. *Algorithms in Combinatorial Geometry*, Springer-Verlag, Berlin, 1987.

[5] Michael R. Garey and David S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness*, W.H. Freeman and Company, New York, 1979.

[6] Jacob E. Goodman, Richard Pollack, Bernd Sturmfels. Coordinate Representation of Order Types Requires Exponential Storage. *21st Annual ACM Symposium on the Theory of Computing*, ACM Press, May 1989.

[7] Christoph M. Hoffmann. *Geometric and Solid Modeling: An Introduction*, Morgan Kaufmann Publishers, Inc., San Mateo, CA, 1989.

[8] Victor Milenkovic. Rounding Face Lattices in the Plane. *First Canadian Conference on Computational Geometry*, August 21-25 1989.

[9] N. E. Mnev. The universality theorems on the classification problem of configuration varieties and convex polytopes varieties. In *Topology and Geometry: Rohlin Seminar, Lecture Notes in Mathematics, No. 1346* (O. Ya. Viro, ed.), Springer-Verlag, New York, 1988.

[10] Kokichi Sugihara. On Finite-Precision Representations of Geometric Objects. *Journal of Computer and System Sciences*, 39:236-247, 1989.

[11] Roberto Tamassia and I. G. Tollis. Planar Grid Embedding in Linear Time. *IEEE Transactions on Circuits and Systems*, Vol. CAS-36, No. 9, 1230-1234, September 1989.