Lecture 5
GEOMETRIC APPROACHES

We look at some geometric approaches to nonrobustness. An intriguing idea here is the notion of "fixed precision geometry". After all, if nonrobustness is a geometric phenomenon, it makes sense to modify the geometry to reflect the fixed precision we find in numbers. There are many ways to create such ersatz geometries*, which in various ways try to recover the basic properties of standard Euclidean geometry. We shall illustrate several such geometries through a discussion of "fixed precision lines".*

§1. What is a Finite Precision Line?

Since qualitative errors is geometric in nature, we can attempt to modify the underlying geometry to be achieve robust behavior. E.g., with finite precision arithmetic, it seems reasonable to replace standard Euclidean geometry by some finite precision geometry. We have already met such a geometry in Chapter 1: we interpreted the standard epsilon-tweaking trick as manipulating new kinds of "points" and "lines", namely, fat points and fat lines.

But there are many potential candidates for finite-precision geometries, even for the simple concept of lines. We illustrate with some common answers [24]. to the question "what is a finite precision line?" Each answer can be viewed as representative of a general approach to finite precision geometry.

- Interval Geometry: We use the term "interval geometry" to refer any approach which replaces standard geometric objects by "fat geometric objects". A "fat point" can be a disc, but it can be generally be any simply connected bounded region. A "fat line" can be any region bounded by two infinite polygonal paths (probably with other properties as well). This is illustrated in Figure 1(a). This is the geometric analogue of interval arithmetic, and has been proposed in many different settings, e.g., [18, 12]. Guibas and Stolfi [6] investigated a form of such geometry ("ε-geometry") by focusing on geometric ε-predicates that can have yes/no/uncertain values. Interval geometry is also attractive from another viewpoint: in the field of mechanical design and manufacture, the tolerancing of geometric objects [23, 13] is a critical problem. The computational aspects of this field is being pursued under the name of computational metrology [9, 25, 14].

- Topologically Consistent Geometry: A line can be distorted into a curve, which in practice would be a polyline. This is illustrated in Figure 1(b). The goal of distortion is to preserve some desired topological property of the original lines. For instance, polylines arising in when computing the arrangements of lines aim to avoid introducing extraneous intersections. Consider the following grid model studied by Greene and Yao [4]. Let G

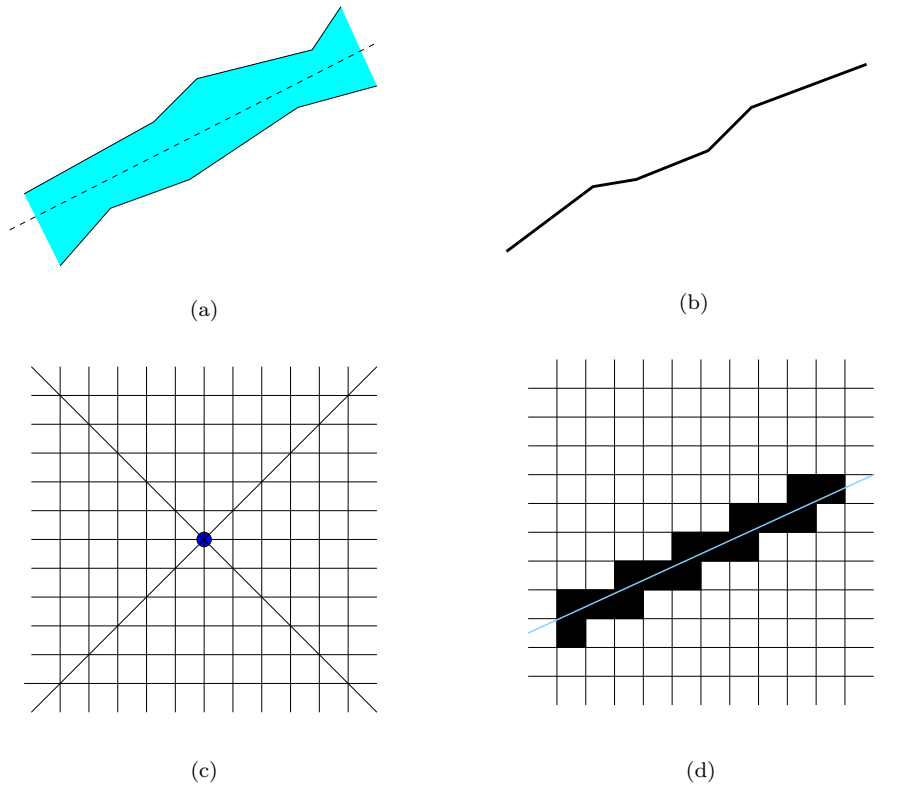


Figure 1: Finite Precision Line Geometry

be a grid, typically $G = \mathbb{Z} \times \mathbb{Z}$. Assume the input are segments whose end points lie on this grid. For each input pair (s, s') of intersecting line segments, we move the intersecting point to a close grid point $p \in G$. The segment s, s' are now converted into polysegments with the same end points as before, but now passing through p. Greene and Yao require the transition from s into a polysegment not to cross any gridpoint, leading to polysegments with continued fraction type properties.

Sugihara and Iri [21, 22] propose topological consistency as a fundamental principle for achieving robustness.

- Rounded Parameter Geometry: Assume our geometric objects live in some parametric space. For instance, lines can be represented in the form $L = \mathsf{Line}(a, b, c)$ as in Chapter 1. This corresponds to the Euclidean line with equation $aX + bY + c = 0$. We want to round the line parameters (a, b, c) to some (a', b', c') where a', b', c' comes from some finite set of representable values (say L-bit integers). We call $\mathsf{Line}(a', b', c')$ a **rounded line**. Sugihara [19] [CHECK!] discusses several approaches to rounding lines. For instance, we might like to restrict $|a'|, |b'|, |c'|$ to lie in the range $[0, 2^L)$. But in fact, it turns out that we get a "nicer class" of such rounded lines is obtained if we let $|c'|$ lie in the slightly larger range $[0, 4^L)$. Such a set of lines is illustrated in Figure 1(c).

- Discretized Geometry: This is well-known in computer graphics: a line is a suitable set of pixels on a finite screen. In contrast to the previous rounding in parameter space, in discretized geometry we discretize the underlying Euclidean space where the geometric object lives. Figure 1(d)

illustrates a discretized line represented as a set of darkened pixels. This is called a Bresenham line in computer graphics. In the field of image processing, an area called "digital geometry" investigates properties of such discretized representation of objects.

Software and Language Support. Regardless of the approach to robustness, it is an important to provide software support for the particular approach. Thus, the development of `Pascal SC` as an extension of `Pascal` for scientific computation supports robust arithmetic with facilities for rounding, overloading, dynamic arrays and modules [1]. These facilities have been extended and made available in other languages: `PASCAL-XSC`, `C-XSC` and `ACRITH-XSC`. The language **Numerical Turing** [8, 10, 11] has a concept of "precision block" whereby the computation within such a block can be repeated with increasing precision until some objective is satisfied. While there are well-known numerical libraries such as `LAPACK`, newer efforts such as `LEDA` and `CGAL` aim to provide robust primitives and basic algorithms and data structures as building blocks for geometric applications. Important primitives that ought to be supported by languages or libraries include scalar product [16] and accurate determinant evaluation. More generally, we could extend programming languages to support geometric objects and their manipulation [17]. In the area of programming languages, the proper treatment of arithmetic exceptions [2] ought to become a standard part of high-level language support.

[NOTE: We should considerably expand the details in the four bullets.]

§2. Robust Line Segment Intersection

We consider another fundamental problem in computational geometry: given a set S of line segments in the plane, we want to compute their arrangement. This is a very basic problem that arises in VLSI circuit design, hidden line elimination, clipping and windowing, physical simulations, etc. The arrangement of S is a **1-skeleton** (or network) determined by the line segments and their intersections: a 1-skeleton is basically a graph that is embedded in the plane: its vertices are the endpoints of segments and the intersection points. Its edges are the open portions of segments bounded by two vertices. This is illustrated in Figure 2.

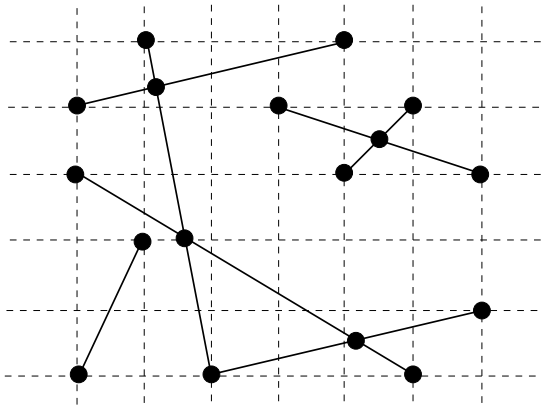


Figure 2: 1-Skeleton of a set of line segments

There are several algorithms for this problem. The asyptotically best algorithm is from Chazelle and Edelsbrunner, with running time $O(k+n\log n)$ where

k is the number of intersection points and n is the number of segments. Here k is the output parameter and ranges from 0 to n^2. More practical algorithms based on randomization have been given Clarkson and also Mulmuley. Algorithms that takes into account robustness issues have been studied by Greene and Yao [4], Milenkovic [15], and Sugihara [20]. The approach of Hobby [7], Guibas and Marimont [5] and Goodrich, et al [3] are based on the concept of "snap rounding", which we will treat in detail.

Recently, Preparata et al considered another issue, namely modifying the primitives using in these algorithms so that the algebraic degree is as small as possible. Although this clearly can help FP computations, it is useful even in AP computation.

Bentley-Ottman Algorithm. The classic algorithm for this problem is from Bentley and Ottmann. This algorithm takes time $O((n+k)\log n)$, and is quite easy to describe. Later we will implement Hobby's snap rounding using this algorithm as its basis [7].

Let us briefly review the Bentley-Ottmann algorithm. We assume a vertical sweepline L with equation $x = x_0$. The line sweeps from left to right (so x_0 is increasing). We have priority queue Q, the **event queue**, initialized to store the end points of each input segment. Each point in Q represents an **event**. An point (x, y) in Q has priority x. Initially, all the points are either **start events** or **stop events**, corresponding to the leftmost or rightmost endpoints of a segment. As L sweeps from left to right, we pull events from Q. At any moment, we keep track of the set of segments that intersect L. These segments are sorted by their the y-coordinate of their intersection with L. We store them in a balanced binary tree T. Moreover, for each pair of adjacent segments in T, if they intersect at a point q to the right of L, then we insert q into Q. Such a q represents the third kind of event, an **intersection event**.

Here is how we update T with each event point. If stop event, we delete the corresponding segment from T If a start event, we insert the corresponding segment into T If intersection event, we exchange the relative order of the two intersecting segments in T.

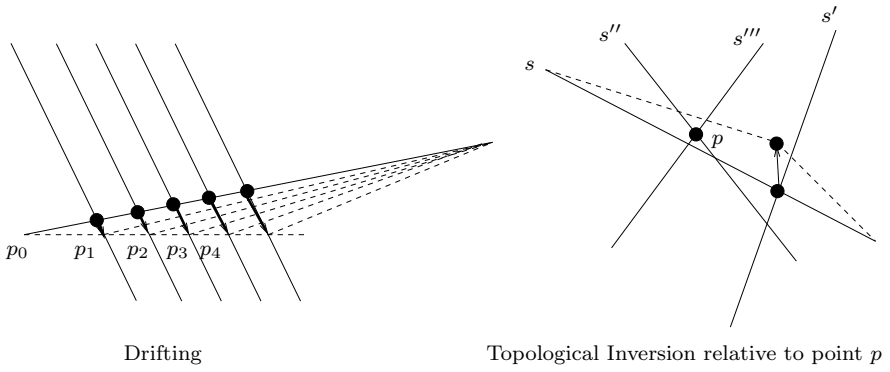
Yao-Greene Grid Model. We consider the following finite-precision model of plane geometry: let $G_2 = \mathbb{Z} \times \mathbb{Z}$ be the **(unit) grid**. All representable points must come from G_2. Suppose s, s' are two representable line segments (i.e., their endpoints are in G_2). If they intersect at a point p, this point is generally not representable. So we must round it to some nearby grid point p'. If $s = [a, b]$, then p' no longer lie on $[a, b]$ – failing a fundamental identity we discussed in Lecture 1:

$$OnLine(Intersect(s, s'), s)$$

is identically true whenever $Intersect(s, s')$ is defined. To restore this identity, we now replace line segments with **polysegments** which is just a finite polygonal path $[a_1, a_2, \ldots, a_n]$. Thus s is now the polysegment $[a, p', b]$ and there is a similar modification for s'. In general, for a polysegment s, we can "snap" a point p in s to a grid point p' and change s accordingly.

This defines our basic model, first studied by Greene and Yao for this problem. What are the issues?

- Unbounded Change: when an intersection point is snapped, other intersection points may move as a result. It is not hard to see that this movement may be unbounded even if the snap distance is bounded.



Drifting Topological Inversion relative to point p

Figure 3: Issues in Snap Rounding

- Cascaded and New Intersections: when we snap a point, we may generate new intersections. If we now snap these new intersections points, we may obtain other new ones. Thus we have a cascading effect. How often do we need to do this? Does this terminate?
- Drifting: Each time we modify a polysegment, the result may drift further and further away from its original position. Is there a bound on this drift? See Figure 3(a) for an illustration of drifting.
- Topological changes: See Figure 3(b) illustrates the possibility of a polysegment moving past an intersection point. We call this an "inversion". A milder version of topological change is to have two points collapse into one (this is called "degeneration").
- Braiding: Two segments intersect in a connected component. If our polysegments intersect is several connected components, we call this "braiding". Even if there were no inversion, braiding can happen.

This list of problems seems formidable [4]. So it is somewhat surprising to see the simple solution described next.

What is an acceptable topological change? Topological change is inevitable in finite precision computation. But we distinguish between **degeneration** (which is acceptable) versus **inversion** (which is not). For instance, if an edge in a planar arrangement of segments is collapsed to a point and it does not affect other relations (except for the obvious ones forced by this collapse), this is an acceptable "degeneration". On the other hand, if a vertex inside a cell is modify to lie outside the cell, this is an unacceptable "inversion". In the presence of degeneration, there is a trivial solution for any segment arrangement: collapse everything to a point. So, our requirements ought to be supplemented by some metric properties such as a bound of the amount of degeneration. Greene and Yao imposed a very strong one, that the deformation of a segment must not cross any grid point. The resulting polysegments has nice properties related to the continued fraction process. We will study another approach from Hobby.

Snap Rounding. It is convenient to introduce the half-open interval $R_1 = (-\frac{1}{2}, \frac{1}{2}]$. Also let $R_2 = R_1 \times R_1$. Call R_1 and R_2 the **rounding interval** or **square**, respectively. It defines the rounding model for us: a number x is rounded to the unique integer $(x + R_1) \cap \mathbb{Z}$, sometimes denoted $\lfloor x \rceil$. In other

words, we round up in case of ties: 2.5 rounds to 3. Similarly, a point p is rounded to the unique point $(p + R_2) \cap G_2$, also denoted $\lfloor p \rceil$. Note that "+" here is the Minkowski sum.

We can view rounding from the view point of the grid: let $\overline{R}_1$ denote the interval $[-\frac{1}{2}, \frac{1}{2})$ (thus, $\overline{R}_1 = -R_1$). If $i \in \mathbb{Z}$, then $\lfloor x \rceil = i$ iff $x \in i + \overline{R}_1$. Similarly, we may define $\overline{R}_2 = \overline{R}_1 \times \overline{R}_1$ and define the square of influence of a grid point $g \in G_2$ to be $g + \overline{R}_2$.

Let S be a set of segments, where each pair of segments intersect transversally (we do not allow overlap). Define H to be the set comprising all the end points of segments in S and also all intersection points. So H is finite. Let the rounded point set $\lfloor H \rceil$: these are called the "hot points".

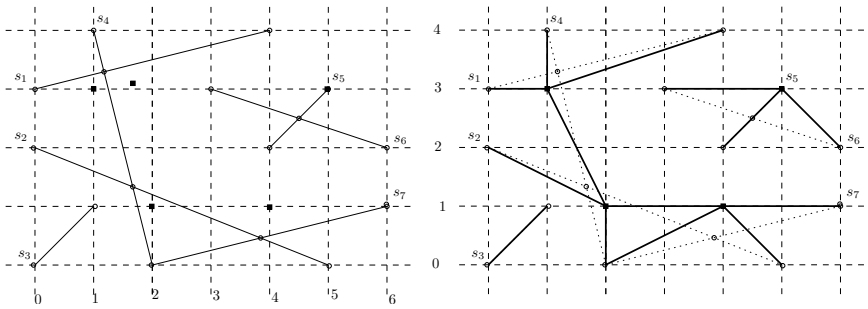


Figure 4: Snap Rounding Based on Hot Points

A polysegment s is said to be **near** a point p if $p + \overline{R}_2 \cap s$ is non-empty. If a polysegment s that passes near a hot point p, we snap s towards p. More precisely, If $q \in p + \overline{R}_2 \cap s$, then we snap q to p (recall the definition of this snap operation above). It is not hard to see that this does not depend on our choice of q. But in the sequel, we will assume that q is the point in s closest to p. The number of vertices in s increases by 1.

Hobby's Theorem on Snap Rounding. Given a segment $s \subseteq \mathbb{R}^2$, define the map $D_T(s)$ to comprise all the segments $[a, b]$ such that there exists $[a', b'] \subseteq s$ such that

$$\lfloor H \rceil \cap ([[a', b'] + R_2) = \{a, b\}.$$

If $[a', b']$ is a maximal segment with this property, we call $[a', b']$ the "preimage" of $[a, b]$. Hence, each segment in $D_T(s)$ has at least one preimage in s. Below we will prove that these preimages are very well-behaved (in particular, there is a unique preimage for each segment in $D_T(s)$).

Lemma 1 *For any segment s there is a direction $(1, \delta)$ (where $\delta = \pm 1$) such that no two points in $G_2 \cap (s + R_2)$ have the same $x + \delta y$ value.*

Proof. If s has positive slope, let $\delta = +1$ else $\delta = -1$. Let $(x_i, y_i) \in s$ for $i = 1, 2$, and $(\lfloor x_i \rceil, \lfloor y_i \rceil) \in (s + R_2) \cap G_2$. Then

$$\lfloor x_1 \rceil + \delta \lfloor y_1 \rceil \neq \lfloor x_2 \rceil + \delta \lfloor y_2 \rceil$$

unless the two rounded points are identical. To see this, suppose the slope is positive and $\lfloor x_1 \rceil + \lfloor y_1 \rceil = c$ for some c. If $\lfloor x_2 \rceil + \lfloor y_2 \rceil = c$, then it is easy to see that they must be diagonal points of a grid square. Indeed, they must be the NW and SE corners of some grid square. But this is impossible. **Q.E.D.**

A collection S of segments are **essentially disjoint** if any pair are disjoint or intersect at their end points only. We also write $D_T(S)$ for the set union of $D_T(s)$, ranging over all $s \in S$.

Lemma 2 *If S have endpoints in T and are essentially disjoint, then the set $D_T(S)$ are also essentially disjoint.*

Proof. Let $s \in S$. The previous lemma ensures that there is a coordinate system

$$(\xi, \eta) = \left(\frac{x + \delta y}{\sqrt{2}}, \frac{-\delta x + y}{\sqrt{2}} \right)$$

such that the endpoints of segments in $D_T(s)$ all have all different ξ coordinates. The transformation $(x, y) \mapsto (\xi, \eta)$ is given by

$$\begin{pmatrix} \xi \\ \eta \end{pmatrix} = M \begin{pmatrix} x \\ y \end{pmatrix}, \qquad \text{where} \quad M = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 & \delta \\ -\delta & 1 \end{bmatrix} \begin{pmatrix} x \\ y \end{pmatrix}.$$

Q.E.D.

The transformation is a rotation, *i.e.*, $\det M = 1$ and the inverse M^{-1} is given by its transpose M^T. E.g., if $\delta = 1$, the points $(\pm 1, 1)$ map to $(1/\sqrt{2}, 0)$ and $(0, 1/\sqrt{2})$.

Lemma 3 *The set of these preimages forms a cover of s. Each segment $D_T(s)$ has a unique preimage.*

Proof. The fact that they form a cover of s follows from the fact that if any maximal open subsegment $(a, b) \subseteq s$ that is dijoint from any preimages, then there must be a preimage of the form $[b, c] \subseteq s$ (for some c). This implies that $[a, c]$ is contained in a preimage, contradicting the assumption that $[b, c]$ is maximal. Uniqueness is trivial by the maximality requirement. **Q.E.D.**

Lemma 4 *If the endpoints in $D_T(S)$ maps to $q_1 = (\xi_1, \eta_1), \ldots, q_m = (\xi_m, \eta_m)$, and $\xi_1 < \xi_2 < \cdots < \xi_m$, then they describe a monotonic piecewise linear function F on the interval $[\xi_1, \xi_m]$ given by*

$$F(\xi) = \frac{\eta_i(\xi_{i+1} - \xi) + \eta_{i+1}(\xi - \xi_i)}{\xi_{i+1} - \xi_i}$$

when $\xi_i \leq \xi \leq \xi_{i+1}$.

Proof. This follows from the fact that the preimages of $[q_i, q_{i+1}]$ are ordered along the segment s in the expected way. [Incomplete] **Q.E.D.**

If $s' \in S$ is a distinct segment, then a similar function F' based on $D_T(s')$ can be defined.

Note that the slope of s' may be different from that of s. This will lead to different coordinate systems for F and F'. To fix this, we may first rotate the coordinate system so that s and s' have both positive slopes or both negative slopes. It is easy to see this is always possible.

These functions approximate the lines ℓ and ℓ' through s and s' (respectively).

Since s, s' are essentially disjoint, we may assume THAT IN THE (ξ, η)-coordinate system, wlog that s is below s' whenever they intersect a common vertical line $\xi = \xi_0$. So it suffices to show that

$$F(\xi) \leq F'(\xi) \tag{1}$$

for all $\xi \in \{\xi_1, \ldots, \xi_m\} \cup \{\xi'_1, \ldots, \xi'_{m'}\}$.

Parametrize the lines ℓ so that $(\xi, \ell(\xi)) \in \ell$ for all ξ. Similarly, assume $(\xi, \ell'(\xi)) \in \ell'$. Since the segment in $D_T(s) \subseteq s_j + R_2$, and similar for s', the difference $F(\xi) - \ell(\xi)$ is limited to the range of the η coordinates in R_2. This ranges over the intervals $(-\frac{1}{2}, \frac{1}{2})$ or $[-\frac{1}{2}, \frac{1}{2})$. [CHECK]

Then our assumptions that $\ell(\xi) \leq \ell'(\xi)$ for all ξ is the range of interest implies that

$$F'(\xi_i) \geq \eta_i, \qquad \text{if } \ell'(\xi_i) \geq \eta_i + \frac{1}{2}.$$

Otherwise, $\ell'(\xi_i) \in \eta_i + R_1$ and the definition of D_T forces $F'(\xi_i) \geq \eta_i$. incomplete

Similarly, we argue that $F(\xi'_j) \leq \eta'_j$. This proves equation (1).

Snap Rounding Based on Bentley-Ottmann's Algorithm. For simplicity, we may assume that the input segments are already representable.

STEP 1. We modify the Bentley-Ottman algorithm as follows. We first run the original algorithm as a "first pass". The intersection points are symbolically represented (by the pair of defining segments). Let H be the set of all endpoints and the intersection points found in this way.

STEP 2. Compute the set of hot points: $\lfloor H \rceil$.

STEP 3. Now we want to snap each segment $s \in S$ to all the hot points p that it is near to. More precisely, for each s we compute a set of pairs $\{(q_i, p_i) : i = 1, \ldots, k\}$ (for some k) such that p_i is a hot point that s is near to, and q_i is the point in s closest to p_i. Then we snap q_i to p_i, in an natural order determined by the q_i's along s. But how do we detect these p_i's? We can do this by modifying the same sweepline algorithm of Bentley-Ottman: for each hot point p, we add the four unit segments corresponding to the boundary of $p + \overline{R}_2$ to the input set. Thus the segment s is near to p iff it intersects one of these unit segments of p.

We can take advantage of the fact that these unit segments are horizontal or vertical and in special positions relative to the grid G_2. An efficient implementation (using the idea of batching, and keeping the unit segments separate from the original segments) is given in the original paper of Hobby.

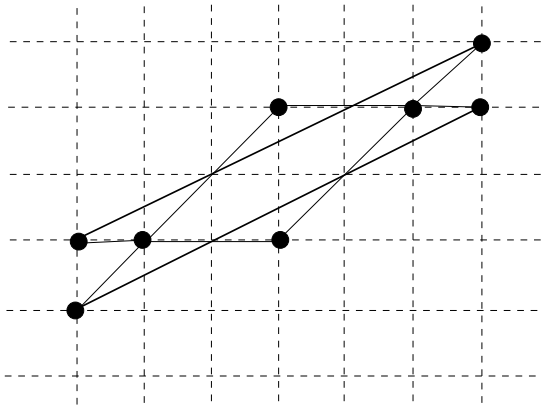


Figure 5: Braiding Behavior in Snap Rounding

Other Approaches. Yao and Greene's. Milenkovic's Sugihara's.

Exercises

Exercise 2.1:
(i) Complete the proof that the ESSA algorithm is correct.
(ii) Describe modifications to the algorithm when exponents are limited in the range $[e_{\min}, e_{\max}]$. ♢

Exercise 2.2: Fix a precision $t \geq 1$. Consider snap rounding in the following "floating point grid" $G_2^t = F(2,t) \times F(2,t)$. Show an example that shows that the snap rounding theorem of Hobby fails in GF_2. ♢

End Exercises

References

[1] G. Bohlender, C. Ullrich, J. W. von Gudenberg, and L. B. Rall. *Pascal-SC*, volume 17 of *Perspectives in Computing*. Academic Press, Boston-San Diego-New York, 1990.

[2] S. Figueroa. *A Rigorous Framework for Fully Supporting the IEEE Standard for Floating-Point Arithmetic in High-Level Programming Languages*. Ph.d. thesis, New York University, 1999.

[3] M. Goodrich, L. J. Guibas, J. Hershberger, and P. Tanenbaum. Snap rounding line segments efficiently in two and three dimensions. In *Proc. 13th Annu. ACM Sympos. Comput. Geom.*, pages 284–293, 1997.

[4] D. H. Greene and F. F. Yao. Finite-resolution computational geometry. *IEEE Foundations of Computer Sci.*, 27:143–152, 1986.

[5] L. Guibas and D. Marimont. Rounding arrangements dynamically. In *Proc. 11th ACM Symp. Computational Geom.*, pages 190–199, 1995.

[6] L. Guibas, D. Salesin, and J. Stolfi. Epsilon geometry: building robust algorithms from imprecise computations. *ACM Symp. on Comp. Geometry*, 5:208–217, 1989.

[7] J. D. Hobby. Practical segment intersection with finite precision output. *Comput. Geometry: Theory and Appl.*, 13:199–214, 1999.

[8] Holt, Matthews, Rosselet, and Cordy. *The Turing Programming Language*. Prentice-Hall, Englewood Cliffs, NJ, 1988.

[9] T. H. Hopp. Computational metrology. In *Proc. 1993 International Forum on Dimensional Tolerancing and Metrology*, pages 207–217, New York, NY, 1993. The American Society of Mechanical Engineers. CRTD-Vol.27.

[10] T. Hull, A. Abraham, M. Cohen, A. Curley, C. Hall, D. Penny, and J. Sawchuk. Numerical Turing. *ACM SIGNUM newsletter*, 20(3):26–34, July, 1985.

[11] T. Hull, M. Cohen, J. Sawchuk, and D. Wortman. Exception handling in scientific computing. *ACM Trans. on Math. Software*, 14(3):201–217, 1988.

[12] D. Jackson. Boundary representation modeling with local tolerances. In *Proc. Symp. on Solid Modeling Foundations and CAD/CAM applications*, pages 247–253, 1995.

[13] N. Juster. Modeling and representation of dimensions and tolerances: a survey. *CAD*, 24(1):3–17, 1992.

[14] K. Mehlhorn, T. Shermer, and C. Yap. A complete roundness classification procedure. In *13th ACM Symp. on Comp. Geometry*, pages 129–138, 1997.

[15] V. Milenkovic. Double precision geometry: a general technique for calculating line and segment intersections using rounded arithmetic. In *Proc. 30th Annual Symp. on Foundations of Comp.Sci.*, pages 500–506. IEEE Computer Society, 1989.

[16] T. Ottmann, G. Thiemt, and C. Ullrich. Numerical stability of geometric algorithms. In *Proc. 3rd ACM Sympos. Comput. Geom.*, pages 119–125, 1987.

[17] M. G. Segal. Programming language support for geometric computations. Report csd-90-557, Dept. Elect. Engrg. Comput. Sci., Univ. California Berkeley, Berkeley, CA, 1990.

[18] M. G. Segal and C. H. Sequin. Consistent calculations for solids modelling. In *Proc. 1st ACM Sympos. Comput. Geom.*, pages 29–38, 1985.

[19] K. Sugihara. On finite-precision representations of geometric objects. *J. of Computer and System Sciences*, 39:236–247, 1989.

[20] K. Sugihara. An intersection algorithm based on Delaunay triangulation. *IEEE Computer Graphics Appl.*, 12(2):59–67, 1992.

[21] K. Sugihara and M. Iri. Two design principles of geometric algorithms in finite precision arithmetic. *Applied Mathematics Letters*, 2:203–206, 1989.

[22] K. Sugihara, M. Iri, H. Inagaki, and T. Imai. Topology-oriented implementation – an approach to robust geometric algorithms. *Algorithmica*, 27:5–20, 2000.

[23] H. B. Voelcker. A current perspective on tolerancing and metrology. In *Proc. 1993 International Forum on Dimensional Tolerancing and Metrology*, pages 49–60, New York, NY, 1993. The American Society of Mechanical Engineers. CRTD-Vol.27.

[24] C. K. Yap. Robust geometric computation. In J. E. Goodman and J. O'Rourke, editors, *Handbook of Discrete and Computational Geometry*, chapter 41, pages 927–952. Chapmen & Hall/CRC, Boca Raton, FL, 2nd edition, 2004. Expanded from 1997 version.

[25] C. K. Yap and E.-C. Chang. Issues in the metrology of geometric toleranc-ing. In J.-P. Laumond and M. Overmars, editors, *Algorithms for Robot Motion Planning and Manipulation*, pages 393–400, Wellesley, Massachusetts, 1997. A.K. Peters. 2nd Workshop on Algorithmic Foundations of Robotics (WAFR).