Digitized Brush Trajectories

John Douglas Hobby
Department of Computer Science
Stanford University
Stanford, California 94305

Abstract

We consider the problem of finding a discrete set of pixels that approximates the envelope of a convex brush shape with respect to a given trajectory. Let the *digitization* of a planar region be the set of pixels whose centers lie inside of it. We develop mathematical models for the width of digitized brush strokes, and we give a class of polygonal brush shapes such that the width of their envelope with respect to a given trajectory is accurately reflected by the digitization of the envelope. Polygonal brush shapes also have the advantage that it is usually much easier to compute the digitization of the envelope with respect to a given trajectory.

We present fast algorithms for approximating a given brush shape with an appropriate polygon so that the digitization of the envelope of the modified brush will have more accurate and uniform width than the digitization of the exact envelope would. We also present an algorithm for finding a set of pixels that represents the envelope of a dynamically changing brush while preserving accurate and uniform stroke width. This algorithm finds a polygonal path with simple rational slopes that is digitally equivalent to the given trajectory. Other possible applications of this polygonal representation include smoothing digitized curves, data compression, and curve fitting.

This research was supported in part by the National Science Foundation under grants IST-820-1926 and MCS-83-00984 and by the System Development Foundation.

DIGITIZED BRUSH TRAJECTORIES

A DISSERTATION
SUBMITTED TO THE DEPARTMENT OF COMPUTER SCIENCE
AND THE COMMITTEE ON GRADUATE STUDIES
OF STANFORD UNIVERSITY
IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR THE DEGREE OF
DOCTOR OF PHILOSOPHY

by
John Douglas Hobby

August 1985

Acknowledgements

I am greatly indebted to my adviser Donald Knuth for creating the research environment that made this work possible, for creating the METAFONT system, and for being so receptive to my ideas. His advice and encouragement have proved invaluable.

I also thank my readers Leo Guibas and Vaughan Pratt for the time and effort that they have put into this work. Finally I thank Lyle Ramshaw and many others for stimulating discussions that have helped me to refine my ideas.

Table of Contents

Chapter 1

Introduction

In computer graphics and other applications it is often useful to describe shapes as the product of moving a brush along a trajectory. The final output is usually produced by some kind of raster device such as a CRT screen or a laser printer, so that the desired shape must be approximated by a set of discrete pixels. We shall investigate ways to take advantage of the special features of brush-trajectory descriptions in order to deal with the discreteness.

In the simplest applications, the trajectory represents a line to be drawn and the brush determines the desired thickness. Of course the discrete raster limits the set of achievable line thicknesses; e.g., horizontal and vertical lines must be at least one pixel wide. There are many applications where this minimum possible width is desirable, and some important work has been done on the problem of drawing thin lines on raster devices.

Perhaps the most important single work on line drawing is J. E. Bresenham's algorithm for drawing thin straight lines [3]. In addition to its speed and simplicity, this algorithm has the advantage that the set of pixels it selects has a simple mathematical description that leads to superior aesthetic qualities. A generalization of this idea is Knuth's "diamond rule" which he used in his original METAFONT system [13]. The effect of the diamond rule is to ensure that straight or curved lines with slopes between 1 and -1 contain one pixel from each column of the raster, and similarly that lines with steeper slopes contain one pixel in each row. A general treatment of line drawing algorithms that are "optimal" in this way is given by Sproull in [27].

Other line drawing algorithms include the digital differential analyzer described by Armstrong in [1]. A general overview of this and related algorithms can be found in Newman/Sproull [19]. Generalizations to curved lines include Lindgård/Moss [17] and Jordan et al. [12]. (See also Belsner [2] and Ramot [23].) These algorithms have been used successfully in many applications, especially where speed and simplicity are important, but they do not obey the diamond rule; hence the thickness of lines produced depends to some degree on parameters other than the slope. There are other algorithms for drawing curved lines that are much easier to analyze because they do obey rules similar to the diamond rule. Such algorithms are given by Bresenham in [4] and by Horn in [11].

For lines of more than the minimum thickness, the approach depends on the type of output device being used. Some printing devices with limited graphics capabilities can draw lines of various thicknesses by using fonts containing short line segments. This technique may have other applications including color CRT displays. (See Lucas [16].)

When color and gray scales are available, it is possible achieve complex effects such as partially transparent brushes Fishkin [6], but the most important application for gray scales is the technique of "antialiasing" which involves using gray pixels at the boundaries of lines so as to make them appear smoother. Some fast antialiasing algorithms are given by Field in [5] and a brief general summary is given by Foley and Van Dam in [7]. Antialiasing can greatly improve the appearance of lines, but the technique is applicable only to certain output devices. Many CRT displays are not capable of displaying gray pixels, and very few printing devices have this capability.

This leaves the important question of how to cope with raster devices that cannot display gray pixels. The basic algorithms for doing this are much the same as the line drawing algorithms given so far. One way to find a discrete version of a brush-trajectory specification is to determine its boundary, and the process of finding such boundaries is essentially equivalent to line drawing. Our goal will be to obtain a description of the desired set of pixels that is sufficient to drive such a line drawing algorithm. A general purpose plotting algorithm that is well suited to this application is described by Knuth in [15].

The shape represented by a brush stroke is the *envelope* of the trajectory $\mathcal{T}$ with respect to the brush $\mathcal{B}$, i.e., the set of all $z + z'$ where z is a point on $\mathcal{T}$ and z' is a point in $\mathcal{B}$. For simple line drawing applications $\mathcal{B}$ is usually a circle or perhaps a rectangle or an ellipse, and the trajectory is a straight line or some spline curve. In general the brush shape might also be described by spline curves.

Although the definition of envelope makes sense in general, we shall require $\mathcal{T}$ and the boundary of $\mathcal{B}$ to be *piecewise real analytic.* That is, they must be decomposable into a finite number of sets, each of which is the range of a real analytic function from a closed interval of $\mathbb{R}$ into $\mathbb{R}^2$. This ensures that the envelope of $\mathcal{T}$ with respect to $\mathcal{B}$ will also have a piecewise real analytic boundary. Some of our results will also be valid under weaker restrictions, but all commonly used spline curves are piecewise real analytic. This work is based on the concept of convolution defined by Guibas, Ramshaw, and Stolfi in [8] and their generalization to curved tracings [24]. This in turn depends on the restriction to piecewise real analytic curves.

Since the theory of tracings is not yet widely known, the main body of this work will be devoted to results that can be understood without that theory. The appendix gives a brief introduction to the theory of tracings and convolutions, followed by a variety of interesting results that depend on the theory. Some concepts introduced in the main body are generalized and treated more formally in the appendix.

1.1. The Digitization of a Region

If a brush stroke or any other shape is to be represented on a raster device such as a CRT display, a laser printer, or a digital typesetter, it is necessary to approximate the shape with a set of discrete pixels. This can be done by merely taking those pixels whose centers lie inside of the shape. We shall refer to this set of pixels as the *digitization* of the shape. We shall never use any other definition for the digitization of a region; when a different set of pixels is desired, we shall adjust the shape before digitizing it.

The digitization is not difficult to compute, but the exact algorithm depends on the family of splines used and the representation of pixels. The best approach is probably to use the generalized concept of digitization described at the end of this section and to adapt one of the line drawing algorithms mentioned previously. A complete implementation of such an algorithm is given by Knuth in [15], but other techniques such as those described by Pratt in [21] are also well suited to the problem. We shall not be concerned with the particular algorithms for digitization or with particular families of spline curves.

We now need a formal definition of digitization. Let us begin by choosing our coordinate system so that pixel centers lie at $(m+\frac{1}{2}, n+\frac{1}{2})$ for integers m and n. We shall represent pixels abstractly as unit squares so that the plane $\mathbb{R}^2$ is divided into discrete pixels

$$P(m,n) = \{\,(x,y) \mid m \le x < m+1 \text{ and } n < y \le n+1\,\} \quad \text{for } m,n \in \mathbb{Z}.$$

A union of such discrete pixels is a *digital region*. A digital region can be thought of as an approximation to where the ink will be placed when the corresponding set of pixels is printed on a raster device. Even though real pixels are seldom uniform or square, the approximation is usually fairly good when averaged over a significant area. Some raster devices tend to erode edges and make black areas smaller, but this distortion is universal and has little effect on comparative analyses of the type that we shall be doing.

The digitization of a region R with piecewise real analytic boundary $B(R)$ is the union of all $P(m,n)$ such that

$$(m+\tfrac{1}{2}, n+\tfrac{1}{2}) \in \big(R \setminus B(R)\big) \cup B_L(R)$$

where $B_L(R)$ is a special subset of $B(R)$ that we shall define shortly.

In the neighborhood of any boundary point z, the boundary $B(R)$ consists of a finite set of real analytic curves leaving z in various directions. There is always a constant $\epsilon > 0$ such that $B(R)$ either includes or does not intersect the line segment $z+(0,\delta)$ for $0<\delta<\epsilon$. Thus for any point $z \in B(R)$, either the following condition holds for $S = R$, or it holds for $S = \bar{R}$, the complement of R: There must exist $\epsilon_1 > 0$ such that if $0 < \delta_1 < \epsilon_1$ then there exists $\epsilon_2 > 0$ possibly depending on δ_1 such that

$$z + (\delta_1, -\delta_2) \in S \quad \text{for } 0 < \delta_2 < \epsilon_2. \tag{1.1.1}$$

The set $B_L(R)$ is exactly those $z \in B(R)$ for which (1.1.1) holds for $S = R$. If R is convex, this means that for the purpose of computing the digitization, we include the left side of R and the all but the rightmost point of the top, but not the right side or the bottom.

The reason that we have been so careful about the boundary of R is that we want to be able to talk about what happens to the digitization when a region is shifted, and we must avoid undesirable behavior when pixel centers fall exactly on the boundary. Such special cases are important in practice and they can considerably detract from the performance of a computer program when they are not handled correctly. Notice that we have made the digitization of R independent of which points of $B(R)$ are contained in R. This is necessary because it is not always possible to select brush shapes whose envelopes will contain the correct boundary points. We have the following lemma:

Theorem 1.1.1. *If R is a finite region bounded by piecewise real analytic curves, then R can be displaced by some amount $(-\delta_1, \delta_2)$ to yield a region R' with the same digitization as R, such that there are no pixel centers on the boundary of R'.*

Proof. Take the minimum of all ϵ_1 from (1.1.1) for all pixel centers on $B(R)$ and choose δ_1 smaller than this. In addition, we also require δ_1 to be smaller than the minimum distance from $B(R)$ to any pixel center not on $B(R)$. For sufficiently small δ_2, we obtain the correct set of boundary pixels without changing the set of pixels whose centers are strictly interior to R. ∎

In practice, we often want to compute the digitization of a region R from a description of its boundary. Let a *path* be a continuous, piecewise real analytic function X from some finite interval I to $\mathbb{R}^2$. If $I = [a, b]$ where $X(a) = X(b)$, we say that X is a *closed path*. The exact correspondence between regions and closed paths is defined in the appendix, but we can assume that a region R is represented by a closed path whose range is the boundary $B(R)$. We can extend the concept of digitization to paths so that if a path X represents a region R then the digitization of X represents the digitization of R. (See Theorem A.4.1.)

We now define the digization of a path $X(t) = \bigl(x(t), y(t)\bigr)$ where t ranges over an interval $[a, b]$. Let $X_1, X_2, \ldots, X_n$ be the sequence of points given by

$$\bar{X}(t) = \bigl(\lceil x(t) - \tfrac{1}{2}\rceil, \lfloor y(t) + \tfrac{1}{2}\rfloor\bigr)$$

as t increases. Build a new sequence $Y_1, Y_2, \ldots, Y_{n'}$ by taking each X_i in order except that for each i such that $X_i = (x_i, y_i)$ and $X_{i+1} = (x_{i+1}, y_{i+1})$ differ in more than one coordinate, insert either (x_i, y_{i+1}) or (x_{i+1}, y_i), whichever has the smallest x-coordinate. Take a polygonal path that connects $Y_1, Y_2, \ldots, Y_{n'}$ in order by line segments.

Note that the case $x_i \neq x_{i+1}$ and $y_i \neq y_{i+1}$ mentioned in the above definition is only possible when $x_{i+1} - x_i = y_i - y_{i+1} = \pm 1$ and $X(t)$ passes through the point $(X_i + X_{i+1})/2$. The choice of the new point to insert ensures that $X(t)$ is treated as being to the left of the pixel center through which it passes. (It is not possible that $x_{i+1} - x_i = y_{i+1} - y_i = \pm 1$ because $\bar{X}(t) = X(t) + (-\frac{1}{2}, \frac{1}{2})$ when $X(t) \in \mathbb{Z}^2 + (\frac{1}{2}, \frac{1}{2})$.)

1.2. Aesthetic Criteria

Perhaps the most straightforward way to represent a brush stroke on a raster device would be to find some mathematical description of the exact envelope and then to take the digitization. There are two problems with this approach: the computation involved is often quite difficult; and the results are often disappointing, especially when the size of the brush is not much more than the resolution of the device. The process of digitization places the edge of the digital region as close as possible to the edge of the ideal envelope, but it is usually more important to represent the width of the envelope accurately.

Figure 1 shows two brush envelopes with their digitizations. In each case the grid lines are pixel boundaries at integer x and y coordinates and the digitization is delimited by bold lines. Figure 1a shows how parallel segments of the trajectory can yield lines of markedly different thickness, depending on the placement of the envelope relative to the pixels.

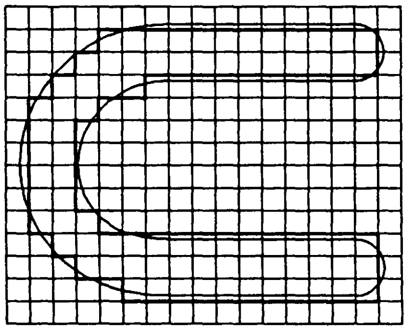

Fig. 1a. A digitized envelope of a circular brush of diameter 2.5

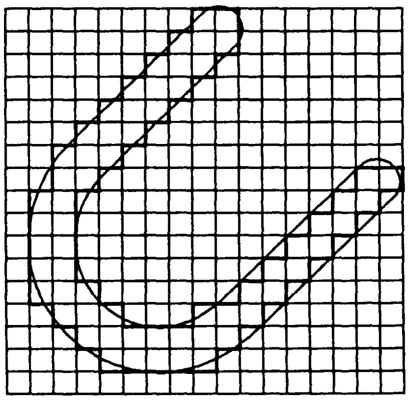

Fig. 1b. A digitized envelope of a circular brush of diameter 2

The asymmetrical appearance of the digitization in Figure 1a could be avoided by forcing the brush diameter to be an integer, but unfortunately this works only for horizontal and vertical lines. As Figure 1b shows, the same problem can occur with circular brushes of integral diameter. One of the legs still comes out half again as thick as the other, even though the exact envelope is symmetrical.

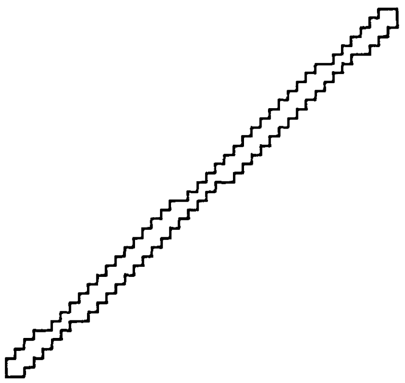

Fig. 2a. The digitization of a straight stroke with a circular brush of diameter 2

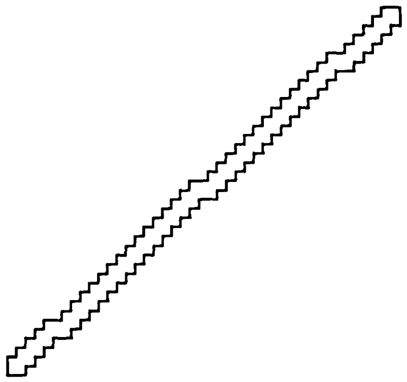

Fig. 2b. A similar digitization based on a circle of diameter 2.14

Many problems of the type shown in Figure 1 could be avoided by carefully controlling the position of the trajectory and distorting it slightly if necessary to get all parts positioned correctly. However, there are still more problems: Sometimes even a straight trajectory can lead to multiple thick and thin spots as shown in Figure 2a. In this case only the digitization is shown, but if the brush envelope were repositioned before digitizing, the only effect would be to reposition the thin spots. The problem is that while integer brush diameters are good for horizontal and vertical strokes, the circle of diameter 2 is a little bit too small for lines of the slope shown in Figure 2.

1.3. Computing Sets of Pixels

Now let us consider the computational difficulty of finding an appropriate set of pixels to approximate a brush stroke. Assume that we do want the digitization of the exact envelope and consider how the real analytic pieces of the trajectory and brush boundary can be represented. These curves are usually described either with an explicit parameter as $\big(x(t), y(t)\big)$, or implicitly by $f(x,y)=0$, where $x(t)$, $y(t)$, and $f(x,y)$ are quadratic or cubic polynomials. Given a brush and trajectory composed of pieces of this form, we can always find equations $g_i(x,y)=0$ for the exact envelope, where each g_i is a polynomial in two variables. The minimum possible degree of g gives us some indication of how hard it is to plot g_i directly. If the brush and trajectory are both conic sections (implicit curves of degree 2), the polynomial g must have degree 8; in the common case where both brush and trajectory are parametric cubics, g has to be of degree 11 in general. [24]

Of course the fact that a polynomial equation for a piece of the envelope must have high degree does not prove that the envelope is very difficult to plot, but it does indicate that the direct approach is likely be very expensive. It takes $O(n^2)$ arithmetic operations to evaluate a general nth degree polynomial in x and y, and the calculations may have to be done to a very high precision.

In his original METAFONT system (now called METAFONT 79), Knuth avoided this problem by using discrete *pens* composed of pixels [13]. Knuth's plotting algorithm had the effect of approximating the trajectory with a sequence of integer spaced points, placing the discrete brush at each, and blackening all the pixels so covered. As implemented in METAFONT 79, this takes time quadratic in the resolution, but it can be done in linear time if the digital output is represented with run lengths and the pens are suitably restricted [26]. The new METAFONT system described in [15] uses a simplified version of the algorithm that we shall develop below, and this in turn can be viewed as a generalization of the algorithm used in METAFONT 79. In honor of METAFONT we shall refer to the brush shapes that the algorithm actually uses as "pens".

Another way to get around the computational difficulties is to approximate the brush with a polygon. The boundary of the envelope then consists of straight line segments and shifted pieces of the trajectory, all of which are relatively easy to digitize. As we shall see, if we choose the polygon carefully, we can get better results with the polygonal approximation than we could with the original brush, therefore having the best of both worlds.

1.4. Overview of Thesis

We begin our investigation in Chapter 2 by developing criteria for measuring the weight of digitized brush strokes. Then we consider infinite straight line trajectories and define a class of "good" brush widths. We define an integer offset condition on brush envelopes that generalizes the concept of "good width" to curved trajectories and allows us to obtain good bounds on the accuracy and uniformity of stroke weight when such envelopes are digitized.

In Chapter 3 we derive a class of polygonal brush shapes or pens that have good widths and produce envelopes that satisfy the integer offset property. Furthermore, we show that this class is unique in the sense that no other brush shapes can have good widths in all directions.

In Chapter 4, we develop algorithms for finding pens that approximate an arbitrary convex brush shape. First we show how to reduce such a brush shape $\mathcal{B}$ to a version $\mathcal{B}'$ that has the same width as a function of direction, but has 180° degree rotational symmetry. We then present a simple algorithm for approximating such a symmetrical brush with a polygonal pen, and we show that the results are nearly optimal. We give two generalizations of the pen finding algorithm to the asymmetrical case: The first version often performs well in practice, but has no good error bound; the second version has a good error bound but has practical disadvantages. A hybrid algorithm combines the advantages of both versions at the expense of increased running time. The hybrid algorithm is difficult to analyze, but the other algorithms run in time $O(d^{2/3})$ and produce polygons with $O(d^{2/3})$ vertices, where d is the brush diameter.

In Chapter 5, we consider a more direct approach to the problem of producing envelopes with integer offsets. The object is to simulate a *dynamic brush*, i.e., a line segment of varying length maintained perpendicular to the trajectory. This requires ways of merging different integer offsets into the same envelope. We give a method for doing this, and we find properties of the discretized trajectory that can help to locate points where integer offsets can be changed without producing unnecessary glitches. Finally, we give an algorithm for deciding which offsets should be used and exactly where it is best to change offsets.

In Chapter 6, we conclude with a discussion of the results obtained with polygonal pens and with the dynamic brush simulation algorithm. We also mention some ideas for smoothing digitized trajectories.

The appendix begins with a brief introduction to the theory of tracings and convolutions from [8] and [24]. We generalize pens to a special class of *monostrophic tracings* and show how convolution can be used to deal with asymmetric pens. We also give a generalized method for finding polygonal versions of digitized trajectories. This has many applications including data compression and curve fitting.

Chapter 2

Measuring Stroke Weight

The examples of Section 1.2 suggest that we should investigate the apparent width of digitized brush strokes. Since this is a subjective concept, it is not easy to give exact definitions, but we shall begin with cases where the concept is clear, and then develop a good general model that tells us what we need to know about stroke weights.

We first develop conditions that are necessary to obtain appropriate stroke weights for straight line trajectories. As we shall see in the next chapter, these conditions alone are very powerful.

2.1. Apparent Width and Infinite Straight Line Trajectories

The easiest case is that of an infinite straight line trajectory. Since the digitization of a region is independent of which boundary points are included, we shall assume that the brush is a closed region. For any such brush, the envelope of an infinite straight line trajectory can always be described by an equation of the form

$$|ax + by - c| \leq d. \tag{2.1.1}$$

We shall investigate the apparent width of the digitizations of such envelopes. We want the overall width to be independent of the placement on the raster; i.e., we want it to be independent of c. We also want the apparent width to be close to the width $w_1 = 2d/\sqrt{a^2+b^2}$ of the exact envelope.

Intuitively, the apparent width should be the number of pixels per unit of length along the line. We can find the average apparent width for a section of a digitized stroke by finding the number of pixels in that section and dividing by its length. The *total pixel weight* of the digitization of (2.1.1) between $bx - ay = e_1$ and $bx - ay = e_2$ is the area of the intersection of the digital region with the region between these lines. This is similar to counting the pixels whose centers lie between the lines except that it does a better job of counting fractional pixels.

The *average apparent width* of the digitization of (2.1.1) between two lines $bx - ay = e_1$ and $bx - ay = e_2$ is the corresponding total pixel weight divided by their separation $|e_1 - e_2| / \sqrt{a^2+b^2}$. The average apparent width for the entire infinite stroke is just the limit as $e_1 \to -\infty$ and $e_2 \to \infty$.

The behavior of the average apparent width of the digitization of the region (2.1.1) depends on whether the slope a/b is rational or irrational. If $b = 0$ or if a/b is rational we say that (a, b) is a *rational pair* or a *rational direction*; otherwise it is an *irrational pair*. If (a, b) is a rational pair then we can assume without loss of generality that they are a *reduced rational pair*, that is that a and b are relatively prime integers. Note that if $a = 0$ then $b = \pm 1$, and vice versa.

The following lemma will allow us to determine the behavior of the average apparent width. Note that the digitization of (2.1.1) may be empty when (a, b) is a reduced rational pair, even when $d > 0$.

Lemma 2.1.1. *If a and b are relatively prime integers, then there are pixel centers satisfying $ax + by = c$ if and only if $c + \gamma(a, b)$ is an integer, where*

$$\gamma(a,b) = \begin{cases} 0 & \text{if } a \text{ and } b \text{ are odd,} \\ \frac{1}{2} & \text{otherwise.} \end{cases}$$

Furthermore, if (x_0, y_0) is a pixel center such that $ax_0 + by_0 = c$ then the solution set is $\{\, (x_0, y_0) + n(b, -a) \mid n \in \mathbb{Z} \,\}$.

Proof. Since $x \equiv y \equiv \frac{1}{2}$ (modulo 1) for pixel centers (x, y), it follows that $c \equiv 0$ when a and b are both odd. Otherwise $c \equiv \frac{1}{2}$ since a and b cannot both be even when $\gcd(a, b) = 1$. Conversely, if $c \equiv \gamma(a, b)$, we know that $ax + by \equiv c$ for any pixel center (x, y). Since there exist integers m and n such that $am + bn = 1$, we can achieve $ax + by = c$ by adding the appropriate multiple of (m, n) to (x, y).

If $ax+by = ax'+by' = c$ then $(x, y)-(x', y')$ must be a real multiple of $(b, -a)$. The previous argument shows that this difference is in $\mathbb{Z}^2$, but since $(b, -a)$ is also reduced, all such multiples of $(b, -a)$ must be integer multiples. This shows that all solutions must be of the required form. It is clear that adding a multiple of $(b, -a)$ to a solution (x, y) does not change $ax + by$. ∎

Corollary 2.1.2. *If (a, b) is a reduced rational pair, then the average apparent width of the digitization of the region described by (2.1.1) is*

$$\begin{cases} \big|\lfloor c + d + \gamma(a,b)\rfloor - \lfloor c - d + \gamma(a,b)\rfloor\big| / \sqrt{a^2 + b^2} & \text{if } a < 0 \text{ or } a = 0 < b; \\ \big|\lceil c + d + \gamma(a,b)\rceil - \lceil c - d + \gamma(a,b)\rceil\big| / \sqrt{a^2 + b^2} & \text{if } a > 0 \text{ or } a = 0 > b. \end{cases}$$

Proof. If R is the region described by (2.1.1), then $B_L(R)$ will be the bounding line $ax+by = c+d$. Thus the numerator gives the number of lines in $\big(R \backslash B(R)\big) \cup B_L(R)$ that satisfy the conditions of Lemma 2.1.1, and the last part of the lemma shows that each line contributes $1/\sqrt{a^2 + b^2}$ to the average apparent width. ∎

Corollary 2.1.3. *If (a, b) is a reduced rational pair, then the average apparent width of the digitization of the region described by (2.1.1) is independent of c if and only if $2d$ is an integer.* ∎

When (a, b) is a reduced rational pair, Lemma 2.1.1 allows us to break the set of pixel centers into equivalence classes based on the value of c for which $ax+by = c$. We can also divide the set of lines perpendicular to (a, b) into equivalence classes by saying that $ax + by = c$ is in class

$$\begin{cases} \lfloor c + \gamma(a,b)\rfloor & \text{if } a < 0 \text{ or } a = 0 < b \\ \lceil c + \gamma(a,b)\rceil & \text{if } a > 0 \text{ or } a = 0 > b \end{cases} \tag{2.1.2}$$

with respect to (a, b). Corollary 2.1.2 tells us that the average apparent width of (2.1.1) is always an integer multiple of $1/\sqrt{a^2 + b^2}$ in this case. Since this basic unit of width is the width of the equivalence classes determined by (2.1.2), we say that widths given in terms of this are given in *(a, b) classes*.

2.2. Integer Offset Vectors and the Accuracy of Apparent Width

We have developed conditions on the envelope width that are necessary in order to ensure that the average apparent width is independent of the trajectory position, but this does not cover all the aesthetic criteria given in Section 1.2. We shall now take a closer look at the average apparent width, and determine additional conditions that are sufficient to guarantee uniform, accurate width. Rather than just taking the average over an entire infinite stroke, we look at averages over short sections of the stroke.

To illustrate the need for this refinement, consider what happens when the slope a/b is irrational in our previous analysis. As we shall see later, the average apparent width of the digitization is always equal to the width w_1 of the region described by (2.1.1) in this case. The situation is entirely different when (a, b) is perturbed slightly to yield a rational pair, even though a large portion of the digitization may remain unchanged.

When the average apparent width is measured over finite lengths, it is not constant, but depends on where the average is taken. When the trajectory slope is irrational the overall average approaches the ideal width w_1, but averages over finite intervals behave as they do for nearby rational slopes. The appropriate tool for limiting this unevenness in average apparent width is the *integer offset vector.*

For rational slope trajectories, the existence of integer offset vectors is a direct consequence of the discrete possibilities for the average apparent width of infinite brush strokes. If (a, b) is a rational pair and if the brush has a *good width* in the direction (a, b), that is if the width of (2.1.1) is independent of c as required by Corollary 2.1.3, then there is an integer *offset vector* (m, n) for (2.1.1) such that $am + bn = 2d$. In general, an offset vector of a brush envelope is any vector such that the boundary of the envelope contains two curves that are identical except displaced by the offset vector. We shall only be interested in offset vectors where the two curves in question describe the left and right sides of a brush stroke. (Corollary 2.2.5 will make this more precise.)

If there is an integer offset vector (m, n) that is roughly perpendicular to the stroke, we can show that the average apparent width must be nearly constant. Consider the examples in Figure 3 where $(a, b) = (-2, 3)$ and offset vectors are indicated by bold lines. When the average apparent width is $4/\sqrt{13}$ (Figure 3b) there is no integer offset vector roughly perpendicular to the stroke, and the stroke weight appears comparatively uneven; the results are much better when the widths are $3/\sqrt{13}$ (Figure 3a) or $5/\sqrt{13}$ (Figure 3c). The following lemma will enable us to express this phenomenon in terms of average apparent width.

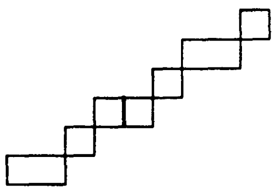
Fig. 3a. $2d = 3$

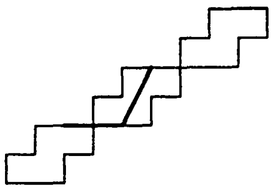
Fig. 3b. $2d = 4$

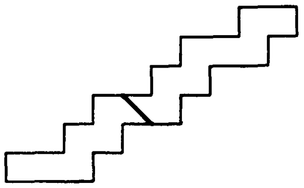
Fig. 3c. $2d = 5$

Lemma 2.2.1. *Let m and n be fixed integers not both 0; let C be a curve that intersects each line $nx - my = u$ exactly once; let R be the region bounded by C and the same curve shifted by (m, n); and let R' be the region $u_1 \leq nx - my \leq u_2$. Then the area of the intersection of R' with the digitization of R is the same as the area of $R \cap R'$.*

Proof. A point (x_0, y_0) is in the digitization of R if and only if $(x_0', y_0') \in \bar{R}$ where $\bar{R} = \big(R \setminus B(R)\big) \cup B_L(R)$ and (x_0', y_0') is the center of the pixel containing (x_0, y_0). Since the intersection of $\bar{R}$ with a line of the form $nx - my = u$ is a segment of length $\sqrt{m^2 + n^2}$ that contains exactly one of its endpoints, it follows that there is a unique integer i such that $(x_0' + im, y_0' + in) \in \bar{R}$. Hence for any point (x_0, y_0), there is a unique integer i such that $(x_0 + im, y_0 + in) \in \mathcal{D}(R)$ where $\mathcal{D}(R)$ is the digitization of R.

If ℓ is a line of the form $nx - my = u$, then the intersection $\ell \cap \mathcal{D}(R)$ is a finite number of line segments. Let f be the mapping from ℓ into $\ell \cap \bar{R}$ such that $f(x, y) = (x + im, y + in)$ where i is the unique integer such that $(x + im, y + in) \in \ell \cap \bar{R}$. If s is a segment of ℓ such that f is 1 to 1 on s, then the image $f(s)$ is a set of line segments whose total length is equal to the length of s. Since f is 1 to 1 on $\ell \cap \mathcal{D}(R)$, the total length of the segments in $\ell \cap \mathcal{D}(R)$ is equal to the length of the segment in the image $f\big(\ell \cap \mathcal{D}(R)\big) = \ell \cap \bar{R}$. Hence if dR' is the region $u \leq nx - my < u + du$, then the differential areas $R \cap dR'$ and $\mathcal{D}(R) \cap dR'$ are equal. Thus the areas $R \cap R'$ and $\mathcal{D}(R) \cap R'$ are also equal. ∎

If there is an integer offset that is exactly perpendicular to the stroke, then this lemma shows that the average apparent width of the digitization of (2.1.1) is always w_1. When the integer offset vector is not perpendicular, the average apparent width can deviate from the ideal value, but we can often show that this deviation cannot be very large. First, we shall need some tools for limiting the difference between digitizations and the regions that they are derived from.

Let the *maximum cover* of a region R be

$$\{\,(x, y) \mid -\tfrac{1}{2} \leq x - x' < \tfrac{1}{2} \text{ and } -\tfrac{1}{2} < y - y' \leq \tfrac{1}{2} \text{ where } (x', y') \in R \cup B(R)\,\} \tag{2.2.1}$$

and let the *minimum cover* be

$$\{\,(x, y) \mid \text{if } -\tfrac{1}{2} \leq x - x' < \tfrac{1}{2} \text{ and } -\tfrac{1}{2} < y - y' \leq \tfrac{1}{2} \text{ then } (x', y') \in R \setminus B(R)\,\} \tag{2.2.2}$$

where $B(R)$ is the boundary of R. Note that the difference between the maximum and minimum covers of R is the envelope with respect to $-\frac{1}{2} \leq x < \frac{1}{2}$, $-\frac{1}{2} < y \leq \frac{1}{2}$ of the boundary of R. The definitions immediately imply the following containment relationship, if we let $x' = \lfloor x \rfloor + \frac{1}{2}$ and $y' = \lceil y \rceil - \frac{1}{2}$.

Lemma 2.2.2. *The minimum cover of a region R is contained in the digitization of R, which is contained in the maximum cover.*

Lemmas 2.2.1 and 2.2.2 allow us to bound the average apparent width when there is an integer offset vector. The purpose of the following theorem is to show that the problems illustrated in Figures 1 and 2 cannot occur when there is a good enough integer offset vector. Later we can use the extra generality of Lemma 2.2.1 to extend the theorem to curved trajectories of practical importance.

Theorem 2.2.3. *If (m,n) is an integer offset vector for (2.1.1), then the average apparent width of the corresponding digitization between two lines*

$$bx - ay = e_1 \quad \text{and} \quad bx - ay = e_2 \tag{2.2.3}$$

separated by a distance s differs from the true width w_1 by a factor of $1+\epsilon$, where $|\epsilon| \le \delta$,

$$\delta = \frac{(w_1+\bar{w})^2 - \max(0, w_1-\bar{w})^2}{4sw_1}\tan\theta = \frac{\bar{w}}{s}\left(1 + \frac{\max(0,\bar{w}-w_1)^2}{4w_1\bar{w}}\right)\tan\theta, \tag{2.2.4}$$

$\bar{w} = (|a|+|b|)/\sqrt{a^2+b^2}$, and θ is the angle between (a,b) and (m,n).

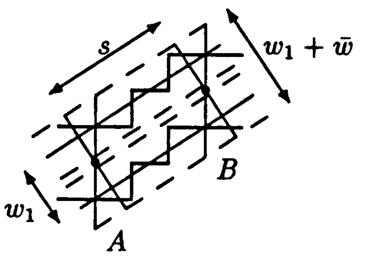

Fig. 4. The relationship between total pixel weight and the area from Lemma 2.2.1

Proof. Let A and B be the points where the bounding lines (2.2.3) intersect the center of (2.1.1) as shown with heavy dots in Figure 4, and construct lines through them parallel to the offset vector. Lemma 2.2.1 shows that the area of the digitization of (2.1.1) between these lines is sw_1. The difference between this and the total pixel weight between A and B is determined by the area of the intersection of the digitization of the region in question with the triangles formed by (2.2.3) and the offset lines just constructed through points A and B. We can use the minimum and maximum cover to bound the area of the digitization in the triangles above B and below A minus the area in the triangles above A and below B. Since $w_1 + \bar{w}$ is the width of the maximum cover of (2.1.1) and $\max(0, w_1 - \bar{w})$ is the width of the minimum cover, the area in question is at most $w_1 s\delta$ as required. ∎

Corollary 2.2.4. *If (a,b) is an irrational pair, then the average apparent width of the digitization of the infinite region described by (2.1.1) is w_1.*

Proof. The theorem shows that the average apparent width is equal to the ideal width $2d/\sqrt{a^2+b^2}$ for any d where there is an integer offset (m,n) such that $am + bn = 2d$. The set of such widths is dense because we can use continued fractions to find $(m,n) \in \mathbb{Z}^2$ for which $am + bn$ is arbitrarily small. Since the apparent width of the given digitization as greater than that of any digital region it contains and less than that of any that contains it, we can bound the apparent width arbitrarily close to w_1. ∎

Theorem 2.2.3 can be extended to curved strokes if we extend the definition of average apparent width. The concept of apparent width depends on being able to assign a direction to the digitized stroke that relates to the perceived overall direction rather than directions on the jagged boundaries of digital regions. We can easily associate such a direction with the digitization of an infinite straight line since it is consistent with a unique slope, but this no longer holds when we allow curved lines. One can imagine various schemes for fitting smooth curves to the boundary of a digital region, but this would not be appropriate for a definition. We therefore take advantage of the fact that we are dealing with the digitization of the envelope a brush with respect to a presumably smooth trajectory and make the definition depend on the envelope.

Let R be a region whose boundary $B(R)$ is composed of piecewise differentiable curves as described in the introduction. Two points A and B on the interiors of real analytic segments of $B(R)$ are *opposite points* of R if the segment between them is contained in R and is perpendicular to the angle bisector of the directions tangent to $B(R)$ at A and B.

Let A', B', C', and D' be four points on $B(R)$ such that the lines $A'B'$ and $C'D'$ are parallel, and the portion of $B(R)$ between these two lines contains unique non-intersecting curves connecting A' to C' and B' to D'. Thus these points define a region $A'B'D'C'$ contained in R. Any region of this form is a *simple subregion* of R.

Let (A, B) and (C, D) be two pairs of opposite points of R such that R has a simple subregion $A'B'D'C'$ for which the following hold: 1) The maximum cover of the region $ABDC$ must lie entirely between the parallel lines $A'B'$ and $C'D'$, and its intersection with R must be contained in $A'B'D'C'$. 2) The lines AB and CD must not intersect within the maximum cover of $A'B'D'C'$. 3) The points A and C must lie on the curve $A'C'$, and B and D must lie on the curve $B'D'$.

When the above conditions hold, the lines AB and CD divide the plane into at most four regions, one of which contains $ABDC$. We call this region R_{ABDC}. The *average apparent width* of R between A, B, C, and D is the area of the intersection of R_{ABDC} with the digitization of $A'B'D'C'$ divided by $\frac{1}{2}L(AC)+\frac{1}{2}L(BD)$, where L denotes arc length. Similarly, the *ideal average width* is the area of $ABDC$ divided by $\frac{1}{2}L(AC)+\frac{1}{2}L(BD)$. It can be shown that these definitions of average width are independent of which simple subregion is chosen, as long as it satisfies the above conditions.

The following corollary makes use of these definitions. The proof is similar to that of Theorem 2.2.3. Note that if the curvature is not too large, then the quantity $\bar{w}$ mentioned in the corollary is very close to

$$\max\left(\left(|a_1|+|b_1|\right)\Big/\sqrt{a_1^2+b_1^2}\,,\quad \left(|a_2|+|b_2|\right)\Big/\sqrt{a_2^2+b_2^2}\right) \qquad (2.2.5)$$

where (a_1, b_1) and (a_2, b_2) are vectors parallel to AB and CD respectively.

Corollary 2.2.5. *Let R be a region with piecewise real analytic boundary $B(R)$, containing eight points A, B, C, D, A', B', C', D' such that the definition of* *average apparent width applies. Assume further that the curves $A'C'$ and $B'D'$ of the simple subregion $A'B'D'C'$ are identical except shifted by an integer vector (m,n), and that they intersect no line parallel to (m,n) more than once. Then the average apparent width differs from the ideal average width by a factor that lies between $1-\delta$ and $1+\delta$ where δ satisfies (2.2.4); $w_1 = \min(d(AB), d(CD))$ and d denotes Euclidean distance; $s = \frac{1}{2}L(BC) + \frac{1}{2}L(AD)$; θ is the angle between the offset vector and either the line AB or the line CD, whichever is larger; and $\bar{w}$ is determined as follows: Construct lines $A''B''$ and $C''D''$ parallel to (m,n) and bisecting AB and CD, and make $\bar{w}$ just large enough so that all points between the lines $A'B'$ and $A''B''$ in the difference between the maximum cover of $A'B'C'D'$ and the minimum cover are within $\bar{w}/2$ of the lines through A and B perpendicular to AB. Furthermore, $\bar{w}$ must satisfy a similar property for points C and D.* ▮

With this result in mind, let us consider the practical benefits of integer offset vectors. We are concerned about variations in apparent width along strokes as in Figure 2 and between one part of a stroke and another as in Figure 1. These undesirable variations in average apparent width are most noticeable when the magnitude of the variations are significant fractions of the stroke weights and they occur over lengths at least comparable to the width of the stroke.

The trivial upper bound on the relative error of the average apparent width from the ideal is $\bar{w}/w$ where w is the minimum ideal width in the region in question. This is independent of the length over which the average is taken. The value of $\bar{w}$ always lies between 1 (for horizontal or vertical strokes) and $\sqrt{2}$ (for diagonal strokes). Straight strokes that are nearly horizontal or nearly vertical can come arbitrarily close to this bound for arbitrarily large s and w, and strokes at nearly 45° from the vertical can have similar variations of half this magnitude. When there is an integer offset vector and $w_1 > \bar{w}$, we have shown that the relative error is at most $(\bar{w}/s)\tan\theta$ where s is the length over which the average is taken and θ is the maximum angle by which the offset vector differs from perpendicular to the stroke. The ratio of this to the trivial upper bound is

$$(w/s)\tan\theta. \tag{2.2.6}$$

When this is less than 1, it gives an estimate of the benefit of having an integer offset vector.

The accuracy of this estimate depends on how closely the trivial upper bound and the bound of Corollary 2.2.5 can be achieved. We have already remarked that the trivial bound can be closely approached for special stroke directions. The bound of Corollary 2.2.5 can be closely approached when θ is small enough so that $w\tan\theta$ is on the order of one pixel. We can then expect to find width variations on the order of $\bar{w}$, and by adjusting s it should be possible to make the difference in pixel weight between the triangular regions considered in Theorem 2.2.3 and Corollary 2.2.5 close to the bounds obtained there.

The average apparent width can give more precise estimates of the absolute accuracy of the stroke weight in particular cases. Variations in stroke weight such as those in Figure 2 are due to variations in average apparent width as one moves along the stroke. The *s-unevenness* of a digitized region R is $(w_{\mathbf{max}} - w_{\mathbf{min}})/(w_{\mathbf{max}} + w_{\mathbf{min}})$ where $w_{\mathbf{max}}$ and $w_{\mathbf{min}}$ are the maximum and minimum w such that there is a subregion of the maximum cover of R of length s over which the average apparent width is w. By the length s, we mean the average arc length from the definition of average apparent width.

It is not hard to see that the s-unevenness of a stroke with an integer offset vector is at most the maximum δ given by (2.2.4), but this is often a significant overestimate. For instance Figure 2b has the integer offset vector $(-1, 2)$, and Theorem 2.2.3 shows that the 4.92-unevenness is at most .085 while in fact it is .034. Even the upper bound is much better than the unevenness of .213 in Figure 2a.

Chapter 3

Polygonal Pens

Having seen the advantages of integer offset vectors, it is natural to ask what conditions on the brush shape are necessary in order to ensure that the envelope of any sufficiently smooth trajectory will have integer offset vectors of the type required by Corollary 2.2.5. One condition that is definitely necessary is that the brush must have a good width in every rational direction. We therefore begin by considering the consequences of requiring good widths in rational directions. We then prove a characterization theorem for the class of brush shapes that satisfy this requirement. Brush shapes that satisfy the characterization theorem are called *pens.*

After proving the characterization theorem, we go on to investigate important properties of pens. The most important such property is that pen envelopes have integer offset vectors. This makes pens a very important class of brush shapes: Any brush shape that is not a pen performs badly even for straight lines of rational slope, while pen envelopes benefit from Theorem 2.2.3 and Corollary 2.2.5, thus sharply limiting possibilities for the effects described in Section 1.2. Furthermore as mentioned in Section 1.3, pen envelopes are particularly easy to digitize.

3.1. The Polygonal Pen Theorem

Assume that a convex brush $\mathcal{B}$ has good widths in all rational directions. We shall determine some of the properties of $\mathcal{B}$ by using a continuity argument based on the fact that the rationals are a dense set and $\mathcal{B}$ has a piecewise real analytic boundary. The following lemma encapsulates what we need to know about real analytic functions:

Lemma 3.1.1. *If $\big(x(t), y(t)\big)$ is a real analytic curve that intersects a line $ax+by = c$ at $t = t_1$, and if the slope $y'(t)/x'(t)$ is monotonic on some interval $t_0 < t < t_1$ where $x'(t) \neq 0$, then as the slope approaches $y'(t_1)/x'(t_1)$, the intersection of the tangent line with $ax + by = c$ approaches $\big(x(t_1), y(t_1)\big)$.*

Proof. The relation $s = y'(t)/x'(t)$ can be inverted to yield the analytic function $t = f(s)$. Thus the desired limit is

$$\lim_{t\to t_1}\left((x,y) + \frac{c-(ax+by)}{ax'+by'}(x',y')\right). \tag{3.1.1}$$

Since the slope is not constant, some sufficiently high order derivative of $ax + by$ must be nonzero at $t = t_1$. Hence by l'Hospital's rule, the second term of (3.1.1) approaches $(0,0)$. ∎

Our argument will be based on the concept of *points of support*, as a function of direction. The points of support of $\mathcal{B}$ in some direction (a, b) are where $\mathcal{B}$ intersects its supporting lines parallel to (a, b). Except where the boundary $B(\mathcal{B})$ contains line segments, there are at most two points of support in each direction $(\cos\theta, \sin\theta)$, and we can distinguish them as "left" and "right". Thus we obtain two functions of θ, each of which are continuous except where θ gives the direction of some line segment contained in $B(\mathcal{B})$.

Lemma 3.1.2. *If a convex brush $\mathcal{B}$ has a good width in every rational direction, then the difference between the left and right supporting points in any rational direction (a, b) not parallel to a line segment of $B(\mathcal{B})$ is an integer vector in $\mathbb{Z}^2$.*

Proof. Assume without loss of generality that (a, b) is a reduced rational pair. Then $\gcd(a, b) = 1$ so there exist integers m and n such that $ma + nb = 1$. Now let $(c, d) = \pm(-n, m)$, choosing the sign so that $ac + bd \geq 0$. The directions $(a_k, b_k) = (c, d) + k(a, b)$ are all between (a, b) and (c, d) when $k > 0$, and they approach (a, b) as k approaches ∞.

Let the supporting lines parallel to (a, b) be $bx - ay = u$ and $bx - ay = u + l$; similarly let the supporting lines parallel to (a_k, b_k) be $b_k x - a_k y = u_k$ and $b_k x - a_k y = u_k + l_k$. The differences l and l_k must be integers because $\mathcal{B}$ has good widths. Furthermore these supporting lines intersect at

$$A^{-1}\begin{pmatrix} u \\ u_k \end{pmatrix} \quad \text{and} \quad A^{-1}\begin{pmatrix} u + l \\ u_k + l_k \end{pmatrix} \qquad \text{where} \quad A = \pm\begin{pmatrix} b & -a \\ b_k & -a_k \end{pmatrix},$$

and since $\det A = ab_k - ba_k = \pm 1$, the difference between the two intersection points is always an integer vector in $\mathbb{Z}^2$. It follows from Lemma 3.1.1 that as k increases, the intersection points approach the points of support in direction (a, b), hence the difference between these is also an integer vector. ∎

It is now relatively easy to characterize the set of brush shapes that have good widths in all rational directions. Since the boundary of $\mathcal{B}$ is assumed to be piecewise real analytic, it contains a finite number of line segments. Between any two adjacent segment directions, the points of support are continuous functions of the direction angle, so their difference is also a continuous function. But since Lemma 3.1.2 shows that the difference is integer valued, it must be constant. We have proved the following theorem. (Two vertices of a convex polygon are called *opposing vertices* if they are the unique points of support in some direction.)

Theorem 3.1.3. *A brush $\mathcal{B}$ has an integer offset in every direction if and only if the convex hull of $\mathcal{B}$ is a finite polygon such that the difference between any pair of opposing vertices is in $\mathbb{Z}^2$.* ∎

We shall refer to convex brush shapes that satisfy Theorem 3.1.3 as *pens*. Such polygons have edges of rational slope. Furthermore if P_1P_2 and P_3P_4 are opposite edges, then the vector sum $(P_2 - P_1) + (P_4 - P_3)$ is in $\mathbb{Z}^2$. An alternative characterization of pens can be found in the next section.

3.2. Pens and Integer Offset Vectors

Our characterization of pens has been based on the simple desire for good widths in rational directions. The purpose of this section is to show that pen envelopes with respect to curved trajectories have integer offset vectors, and thus Corollary 2.2.5 can be used to show that digitized pen envelopes have good accuracy and uniformity of apparent weight. In other words, the brush shapes that produce width independent of trajectory position for straight line trajectories also produce consistent stroke weight for curved trajectories and control variations in apparent width along a single pen stroke.

Let us first consider ordinary polygonal pens. Figure 5 shows a such a pen and a set of curves that describe the boundary of the envelope with respect to the indicated trajectory. The polygonal pen is shown superimposed on an integer grid to demonstrate the integer difference vectors between opposite vertices. Dashed lines connect portions of the envelope boundary curves that are identical except shifted by such an integer vector. Each pair of such identical curves is delineated by a pair of parallel dashed lines, and together such pairs of curves make up most of the envelope.

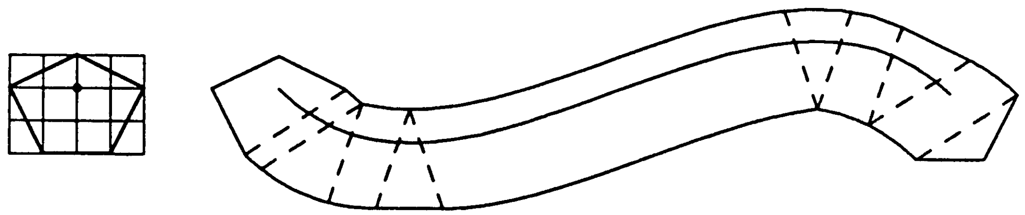

Fig. 5. A polygonal pen and its envelope with respect to a trajectory, showing regions where integer offset vectors apply.

The rule for determining what pairs of integer offset curves will be contained in the envelope boundary is best understood via the concept of convolutions discussed in the appendix, but the basic idea is as follows: First, divide the trajectory into pieces at the points where it achieves slopes parallel to pen edges. If the trajectory contains line segments parallel to pen edges, then any point on such a segment is a suitable break point. Once the trajectory has been broken into pieces, each can be associated with a pair of opposite supporting points on the pen boundary so that the chosen points are supporting points in any direction parallel to any part of the trajectory segment. As the pen moves along the trajectory, the support points describe shifted portions of the trajectory. It is shown in the appendix that all curved portions of the envelope boundary lie on such shifted trajectory segments. Since the difference between opposite vertices must be an integer vector, the relative displacements between the two corresponding segments must be integral.

The exact conditions for this are considered later, but for "reasonably well behaved" pens and trajectories, most of the envelope boundary is composed of pairs of curves that are identical except for an integer offset. In the example of Figure 5, about 70% of the boundary is composed of such curves, and most of the rest of the boundary belongs to the polygonal pen images at either end of the stroke.

3.3. Equivalence Classes of Brush Shapes

In order to get a better understanding of pens, we now examine classes of pens having identical width as a function of angle. The great importance of stroke width is enough to justify our interest in such classes of pens, but the primary application for our analysis of width as a function of angle is to simplify the task of pen construction. This simplification comes from the fact that the equivalence classes that we shall be dealing with are represented by pens that have 180° rotational symmetry about the origin. If we are interested only in width as a function of angle then we can restrict our attention to pens of this form. In fact the appendix shows that by generalizing the notion of "trajectory," we can make the results truly independent of pen shape as long as the width as a function of angle is preserved.

We seek to transform an arbitrary pen $\mathcal{P}$ into a canonical version $\mathcal{P}'$ that is symmetrical about the origin, where the transformation is to be width preserving, i.e., the width of $\mathcal{P}$ in any direction (a,b) is the same as the width of $\mathcal{P}'$ in that direction. Actually, this transformation can be extended to brushes that are not pens, and the additional generality will be useful later so we shall deal with arbitrary convex brushes from now on. Figure 6 shows a convex brush $\mathcal{B}$ and the corresponding symmetrical version $\mathcal{F}(\mathcal{B})$, where $\mathcal{F}$ is the transformation function that we shall develop.

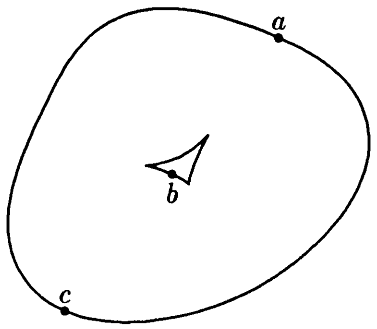

Fig. 6a. A brush and its locus of offset midpoints.

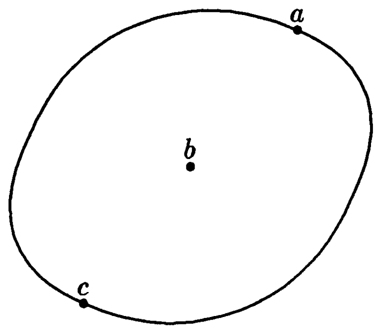

Fig. 6b. The corresponding symmetrical brush.

Consider the brush $\mathcal{B}$ to be processed and let a and c be the points of support in the direction $(\cos\theta, \sin\theta)$ as shown in Figure 6a. Now let b be the midpoint of the segment connecting a and c. As θ scans a complete circle, the vector $a-b$ scans the boundary of a symmetrical brush $\mathcal{B}'$. If the boundary of $\mathcal{B}$ contains line segments, then there will be some directions for which unique points a, b, and c cannot be assigned. Instead, we have sets of triples of support points a and c and midpoints b. The resulting vectors $a-b$ form line segments that we can add to $\mathcal{F}(\mathcal{B})$.

More formally, $\mathcal{F}(\mathcal{B})$ is the set of points z for which there exist points $a = (x_a, y_a)$ and $c = (x_c, y_c)$ on the boundary $B(\mathcal{B})$ and a direction $(u,v) \neq (0,0)$ in

$\mathbb{R}^2$ where $\mathcal{B}$ is contained in the strip

$$\{(x,y) \mid ux_a + vy_a \leq ux + vy \leq ux_c + vy_c\}$$

and $z = \alpha(a-c)$ for some α in the interval $[0, \frac{1}{2}]$.

It is not hard to show that the boundary of $\mathcal{F}(\mathcal{B})$ can be obtained by restricting $\alpha = \frac{1}{2}$ in the above definition, and that this boundary is piecewise real analytic. Thus $\mathcal{F}(\mathcal{B})$ satisfies our requirements for brush shapes. The following lemma gives an alternative characterization of $\mathcal{F}(\mathcal{B})$.

Lemma 3.3.1. *If $\mathcal{B}$ is a convex brush then $\mathcal{F}(\mathcal{B})$ is the set of all $(a-c)/2$ for $a, c \in \mathcal{B}$.*

Proof. If $z \in \mathcal{F}(\mathcal{B})$ then $z = \alpha(a-c)$ where $0 \leq \alpha \leq \frac{1}{2}$ and $\{a, c\} \subseteq \mathcal{B}$. Thus $z = (a-c')/2$ where $c' = a + 2\alpha(c-a)$. Since the point c' is between a and c, it follows by convexity that $c' \in \mathcal{B}$.

Conversely let $z = (a-c)/2$ where $\{a, c\} \subseteq \mathcal{B}$. If $z = (0,0)$ then trivially $z \in \mathcal{F}(\mathcal{B})$. Otherwise, consider all possible pairs of points (a', c') in $\mathcal{B}$ such that $a' - c' = \beta z$ for some real number β, and fix a' and c' so as to maximize β. Clearly $\beta \geq 2$, and thus $z = \alpha(a'-c')$ where $\alpha = 1/\beta \in (0, \frac{1}{2}]$. It only remains to be shown that $\mathcal{B}$ has a pair of parallel supporting lines containing the points a' and c'.

Consider the convex sets

$$S_{a'} = \{(u,v) \mid a' + \epsilon(u,v) \in \mathcal{B} \setminus B(\mathcal{B}) \text{ for some } \epsilon > 0\}$$

and

$$S_{c'} = \{(u,v) \mid c' + \epsilon(u,v) \in \mathcal{B} \setminus B(\mathcal{B}) \text{ for some } \epsilon > 0\}.$$

We shall use the notation $p + S$ for the set of all points $p + q$ where $p \in \mathbb{R}^2$, $S \subseteq \mathbb{R}^2$, and $q \in S$. If some point z lies in $\mathcal{B} \setminus B(\mathcal{B})$ then $z - a' \in S_{a'}$ and $z - c' \in S_{c'}$; i.e., the interior of $\mathcal{B}$ is contained in $(a' + S_{a'}) \cap (c' + S_{c'})$. We can also assume that $S_{a'}$ and $S_{c'}$ are disjoint because otherwise if $(u,v) \in S_{a'} \cap S_{c'}$ then $a' + \epsilon_1(u,v) \in \mathcal{B} \setminus B(\mathcal{B})$ and $c' + \epsilon_2(u,v) \in \mathcal{B} \setminus B(\mathcal{B})$ for some $\epsilon_1, \epsilon_2 > 0$. If we choose $\epsilon < \min(\epsilon_1, \epsilon_2)$ then the two points $a'' = a' + \epsilon(u,v)$ and $c'' = c' + \epsilon(u,v)$ in the interior of $\mathcal{B}$ satisfy $a'' - c'' = a' - c' = \beta z$, contradicting the maximality of β.

It is well known that for any two disjoint convex sets there must be a separating line between them so that the two sets lie on opposite sides of the line. Let ℓ be such a line separating $S_{a'}$ and $S_{c'}$. Since the origin is on the boundaries of both $S_{a'}$ and $S_{c'}$, we can assume that $(0,0) \in \ell$. Thus $a' + \ell$ and $c' + \ell$ are the required supporting lines. ▮

Theorem 3.3.2. *If $\mathcal{B}$ is a convex brush then $\mathcal{B}$ and $\mathcal{F}(\mathcal{B})$ have the same width as a function of angle.*

Proof. Let $(u,v) \in \mathbb{R}^2$ be a nonzero direction vector; and let $a = (x_a, y_a)$ and $c = (x_c, y_c)$ be two supporting points of $\mathcal{B}$ so that

$$\mathcal{B} \subset \{(x,y) \mid l_a \leq ux + vy \leq l_c\}.$$

where $l_a = ux_a + vy_a$ and $l_c = ux_c + vy_c$. If we can show that $\mathcal{F}(\mathcal{B})$ has supporting lines $ux + vy = (l_a - l_c)/2$ and $ux + vu = (l_c - l_a)/2$, then it follows that both $\mathcal{B}$ and $\mathcal{F}(\mathcal{B})$ have width $|l_a - l_c| / \sqrt{u^2 + v^2}$ in the (u, v) direction.

To show that $\mathcal{F}(\mathcal{B})$ has the desired supporting lines, we first use Lemma 3.3.1 to show that $\mathcal{F}(\mathcal{B})$ contains points $\pm(a - c)/2$ on the lines $ux + vy = \pm(l_a - l_c)/2$. Lemma 3.3.1 shows that an arbitrary point $z \in \mathcal{F}(\mathcal{B})$ may be written as $(a' - c')/2$, where $\{a', c'\} \subseteq \mathcal{B}$. If $a' = (x_{a'}, y_{a'})$ and $c' = (x_{c'}, y_{c'})$ then $ux_{a'} + vy_{c'}$ and $ux_{c'} + vy_{c'}$ are both between l_a and l_c. Thus the dot product of $(a' - c')/2$ with (u, v) has absolute value less than $|l_a - l_c| / 2$, and therefore z is between the lines $ux + vy = \pm(l_a - l_c)/2$. This shows that $ux + vy = \pm(l_a - l_c)/2$ are the required supporting lines. ▮

Chapter 4

Algorithms for Generating Polygonal Pens

Now that we have seen the advantages of polygonal pens, it is natural to ask how one goes about approximating an arbitrary convex brush $\mathcal{B}$ with a suitable pen $\mathcal{P}$. If the brush is not convex, it is necessary to find its convex hull and use some other algorithm to remove the unwanted parts of the resulting pen. The theory that we have developed does not apply to non-convex shapes, so there is no reason to force them to be polygons except for ease of computation. From now on, we shall assume that all brush shapes are convex.

Before we go any further, we need a way of measuring the quality of the approximation. Let the *error of $\mathcal{P}$ in approximating $\mathcal{B}$* be

$$E(\mathcal{P},\mathcal{B}) = \max\Bigl(\max_{z\in\mathcal{P}} d(z,\mathcal{B}),\ \max_{z\in\mathcal{B}} d(z,\mathcal{P})\Bigr)$$

where $d(z,S)$ denotes the distance between z and the closest member of S.

Actually, the process of generating pens is more than just approximation; another way of looking at it is that the pen $\mathcal{P}$ is a version of $\mathcal{B}$ with subpixel corrections to ensure accurate stroke weight. The approximation process has the effect of determining the width of the envelope as a function of the trajectory direction. We can choose the width to be as accurate as possible for simple rational directions where the possibilities are widely separated. In other directions, the choice is forced but we can still keep the unevenness of digitized envelopes under control.

We first consider the problem of generating a convex polygonal pen to approximate a brush that is symmetrical about the origin. This special case is very important in practice, because the brush shapes tend to have such symmetry. Even when the brush $\mathcal{B}$ is asymmetrical, we can reduce to the symmetrical case by replacing $\mathcal{B}$ with $\mathcal{F}(\mathcal{B})$. Section A.3 in the appendix discusses methods for modifying a pen created to approximate $\mathcal{F}(\mathcal{B})$ so that it approximates $\mathcal{B}$.

4.1. Generating symmetrical pens

When $\mathcal{B}$ is symmetrical about the origin, we naturally want to approximate it with a pen $\mathcal{P}$ that is also symmetrical. The following lemma shows that we are justified in restricting our attention to symmetrical $\mathcal{P}$ because they always yield better approximations to $\mathcal{B}$.

Lemma 4.1.1. *If $\mathcal{B}$ is a symmetrical brush, then $E(\mathcal{F}(\mathcal{P}),\mathcal{B}) \le E(\mathcal{P},\mathcal{B})$ for any pen $\mathcal{P}$.*

Proof. The first step is to show that for any point $p = (x_p, y_p) \in \mathcal{F}(\mathcal{P})$, there exists a point $q \in \mathcal{P}$ such that $d(q,\mathcal{B}) \ge d(p,\mathcal{B})$, where d is the distance function used to define E. It is well known that if a point p is a distance δ away from a convex set $\mathcal{B}$, then $\mathcal{B}$ has a supporting line that separates p from $\mathcal{B}$ and is at a distance of δ from p. Let $\delta = d(p, \mathcal{B})$; and let $ux + vy = l$ be such a supporting line for $\mathcal{B}$, where (u, v) has unit length, $l \geq 0$, and $ux_p + vy_p = l + \delta$. By symmetry, $ux + vy = -l$ is also a supporting line for $\mathcal{B}$, and thus $d\big((x, y), \mathcal{B}\big) \geq |ux + vy| - l$ for any point (x, y).

Lemma 3.3.1 allows us to assume that $p = (a - c)/2$ where $\{a, c\} \subseteq \mathcal{P}$. Hence

$$(ux_a + vy_a) - (ux_c + vy_c) = 2(l + \delta),$$

where $a = (x_a, y_a)$ and $c = (x_c, y_c)$; and therefore

$$\max\left(|ux_a + vy_a|\,,\ |ux_c + vy_c|\right) \geq l + \delta$$

so that either $d(a, \mathcal{B}) \geq \delta$ or $d(c, \mathcal{B}) \geq \delta$.

It remains to be shown that $d\big(p, \mathcal{F}(\mathcal{P})\big) \leq \max\big(d(p, \mathcal{P}), d(-p, \mathcal{P})\big)$ for any point $p = (x_p, y_p) \in \mathcal{B}$. As before we can assume that $\mathcal{F}(\mathcal{P})$ is contained in the strip $|ux + vy| \leq l$, where $ux_p + vy_p = l + \delta$ and $\delta = d\big(p, \mathcal{F}(\mathcal{P})\big)$. Theorem 3.3.2 implies that $\mathcal{P}$ is contained in a strip $l_0 + l \leq ux + vy \leq l_0 - l$ for some real number l_0. The dot product of $\pm p$ with (u, v) is $\pm(l + \delta)$. If $l_0 \leq 0$ then $l + \delta \geq (l_0 + l) + \delta$ otherwise $-(l + \delta) < (l - l_0) - \delta$. Thus $\max\big(d(p, \mathcal{P}), d(-p, \mathcal{P})\big) \geq \delta$ as required. ∎

Lemma 4.1.1 allows us to restrict our attention to symmetrical $\mathcal{P}$, and Theorem 3.1.3 immediately leads to the conclusion that the vertices of $\mathcal{P}$ lie in the set $\frac{1}{2}\mathbb{Z}^2$ of half integer pairs. Let us see how this observation relates to the ideas about stroke width from Chapter 2.

Recall that when (u, v) is a reduced rational pair, a pen $\mathcal{P}$ has a good width in direction (u, v) only when the width is a multiple of $1/\sqrt{u^2 + v^2}$. Therefore it is most important to select the optimum width in simple rational directions where the quantization is large.

The best way to scan simple rational directions is to generalize the concept of a Stern-Peirce tree ([14], exercise 4.5.3–40) to a "Stern-Peirce wreath" as follows: Start with four nodes $(1, 0)$, $(0, 1)$, $(-1, 0)$, and $(0, -1)$ arranged in a circle in that order, and successively add a new node $(u{+}u', v{+}v')$ between each pair of consecutive nodes (u, v) and (u', v') until some stopping criterion is satisfied.

It is not difficult to prove that consecutive directions (u, v) and (u', v') on the wreath always satisfy

$$uv' - vu' = 1 \quad \text{and} \quad uu' + vv' \geq 0. \tag{4.1.1}$$

Thus we see that the vectors $(v, -u)$ and $(v', -u')$ each span exactly one class with respect to $(u'', v'') = (u{+}u', v{+}v')$; i.e., adding $(v, -u)$ or $(v', -u')$ to a point (x, y) adds 1 to the class number $u''x + v''y$.

The basic idea of the algorithm is to use the Stern-Peirce wreath to generate simple rational directions, and to choose edges in those directions as close as possible to the edge of $\mathcal{B}$. Suppose $\mathcal{B}$ has width w and height h. We start with a $\lfloor w{+}\frac{1}{2}\rfloor$ by $\lfloor h{+}\frac{1}{2}\rfloor$ rectangle and add new edges by slicing off corners. Because of the symmetry we need to represent edges only when $v \geq 0$ as we go counterclockwise around the polygon.

We can associate a *class number* with each edge in the following manner: An edge in direction (u, v) has class number c if the current version of $\mathcal{P}$ is c units wide relative to $(v, -u)$ classes. Another way of looking at it is that the edge lies on the line $vx - uy = c/2$. The beauty of this definition is that we can compute class numbers very easily as we generate directions with the Stern-Peirce wreath. If the edge in direction (u, v) has class number c and if the edge in the adjacent direction (u', v') has class number c', then their intersection point lies on the line $v''x - u''y = (c + c')/2$ where $(u'', v'') = (u+u', v+v')$. In other words, the class number of the new edge is the sum of the class numbers of the edges incident on the corner it cuts off, minus the number of $(v'', -u'')$ classes between the new edge and the corner.

Consider a closed path $X_{\mathcal{B}}(t) = \bigl(x_{\mathcal{B}}(t), y_{\mathcal{B}}(t)\bigr)$ defined on $[a, b]$ such that $X_{\mathcal{B}}$ goes counter-clockwise once around the boundary of the convex brush $\mathcal{B}$ as t increases from a to b. Let $\mathcal{B}(u, v)$ be a function that returns a point in the range of $X_{\mathcal{B}}$ at which $(x'_{\mathcal{B}}, y'_{\mathcal{B}})$ is a postive multiple of the direction (u, v). The exact nature of this function depends on what kind of shape $\mathcal{B}$ has and how that shape is represented, but the computation is usually not difficult. This function is in a sense the core of the following algorithm. The algorithm also occasionally needs to find points where the boundary of $\mathcal{B}$ intersects a given line ℓ. In counterclockwise order the two intersections will be called $I_1(\mathcal{B}, \ell)$ and $I_2(\mathcal{B}, \ell)$.

The algorithm operates on a data structure that consists of a sequence of *edge nodes* and *vertex nodes*, which strictly alternate. If p is a vertex node then we shall refer to the nodes for the edges incident on p as p_l and p_r. For each vertex p we store its coordinates $z(p) = \bigl(x(p), y(p)\bigr)$; for each edge e, we store a reduced rational pair $w(e) = \bigl(u(e), v(e)\bigr)$ giving the direction and two "lengths" $ll(e)$ and $rl(e)$, whose meaning will be explained later. If e is preceded by a vertex node p and followed by a vertex node q, then $z(q) - z(p) = \bigl(ll(e) + rl(e)\bigr) \cdot w(e)$.

In a real implementation it may be desirable to store the edge information in the two adjacent vertex nodes instead of in separate nodes, or perhaps a similar trick could be used to eliminate the vertex nodes. In addition, it is possible to save space by deleting the auxiliary information when advancing p in Step 3 below, since at this point there is no more processing to be done on that vertex.

Any edge e in the data structure determines a directed line segment $\ell(e)$ as follows: If p is a vertex node adjacent to e then $\ell(e)$ is the set of points $z(p) + tw(e)$ for real numbers t, and the direction associated with $\ell(e)$ is $w(e)$. Let $R(e)$ be the set of points to the right of $\ell(e)$, and $L(e)$ be the set of points to the left; i.e., $(x, y) \in R(e)$ if $v(e)x - u(e)y > v(e)x(p) - u(e)y(p)$ and $(x, y) \in L(e)$ if $v(e)x - u(e)y < v(e)x(p) - u(e)y(p)$. For convenience, we shall let $\bar{R}(e) = R(e) \cup \ell(e)$ and $\bar{L}(e) = L(e) \cup \ell(e)$.

To simplify the following presentation, we introduce functions $r(x) = \frac{1}{2}\lfloor 2x + \frac{1}{2}\rfloor$ and $r_d(x) = \frac{1}{2}\lceil 2x - \frac{1}{2}\rceil$ that round to the nearest multiple of $\frac{1}{2}$. (The only difference between these functions is that r_d rounds downward in ambiguous cases while r rounds upward.) We shall also use the notation $\bigl\langle (x_1, y_1), (x_2, y_2) \bigr\rangle$ for the dot product $x_1x_2 + y_1y_2$.

Algorithm 1 (Symmetric brush to symmetric pen).

1) Let $(x_1, y_1) = -\mathcal{B}(1,0) = \mathcal{B}(-1,0)$ and $(x_2, y_2) = \mathcal{B}(0,1)$ and initialize the data structure to $(e_1, v_1, e_2, v_2, e_3)$ where

$$\begin{aligned} z(v_1) &= \big(r(x_2), -r(y_1)\big) & z(v_2) &= \big(r(x_2), r(y_1)\big) \\ w(e_1) &= -w(e_3) = (1,0) & w(e_2) &= (0,1) \\ ll(e_2) &= r(y_2) + r(y_1) & rl(e_1) &= r(x_2) - r(-x_1) \\ ll(e_3) &= r(x_2) + r(-x_1) & rl(e_2) &= r(y_1) - r(y_2). \end{aligned}$$

Finally set $p \leftarrow v_1$ and go on to Step 2.

2) Let $(\bar{u}, \bar{v}) = w(p_l) + w(p_r)$ and $\bar{z} = \mathcal{B}(\bar{u}, \bar{v})$. If either $\mathcal{B}\big(u(p_l), v(p_l)\big)$ or $\mathcal{B}\big(u(p_r), v(p_r)\big)$ belongs to $\bar{R}(p_l) \cap \bar{R}(p_r)$, then set $\delta \leftarrow 0$ and go on to Step 3. If $\bar{z} \in R(p_r) \cap L(p_l)$, then set $\bar{z} \leftarrow I_1(\mathcal{B}, p_r)$; otherwise if $\bar{z} \in R(p_l) \cap L(p_r)$, then set $\bar{z} \leftarrow I_2(\mathcal{B}, p_l)$. The number of $(-\bar{v}, \bar{u})$ classes to cut off at vertex p is

$$\delta = \min\big(rl(p_l),\ ll(p_r),\ r_d(\langle(-\bar{v}, \bar{u}),\ \bar{z} - z(p)\rangle)\big).$$

3) If $\delta > 0$ then go on to Step 4. Otherwise advance p to the next vertex and go to Step 2, but halt if there are no more vertices to advance to.

4) Insert a new vertex q and a new edge q_l between p and p_r. (Thus the edge that was p's right neighbor becomes q's right neighbor, and we now call it q_r.) Now update the data structure as follows and go on to Step 5.

$$\begin{aligned} z(q) &\leftarrow z(p) + \delta \cdot \big(u(q_r), v(q_r)\big) & ll(q_r) &\leftarrow ll(q_r) - \delta \\ z(p) &\leftarrow z(p) - \delta \cdot \big(u(p_l), v(p_l)\big) & rl(p_l) &\leftarrow rl(p_l) - \delta \\ w(q_l) &\leftarrow (\bar{u}, \bar{v}) \end{aligned}$$

5) If $rl(p_l) = 0$ then set $ll(q_l) \leftarrow 0$; otherwise if $ll(q_r) = 0$ then set $ll(q_l) \leftarrow \delta$; otherwise set

$$ll(q_l) \leftarrow r\left(\frac{\langle(\bar{u}, \bar{v}),\ \bar{z} - z(p)\rangle}{\bar{u}^2 + \bar{v}^2}\right). \qquad (4.1.2)$$

Now set $rl(q_l) \leftarrow \delta - ll(q_l)$ and go back to Step 2. ▮

When this algorithm halts it has computed only half of a polygon. If p_1, p_2, ..., p_k are the vertices in the data structure, then the vertices of the pen polygon are

$$\big(z(p_1), z(p_2), \ldots, z(p_k), -z(p_1), -z(p_2), \ldots, -z(p_k)\big).$$

Another interesting property of the algorithm is that it deals with *null edges*; i.e., edges e such that $ll(e) = rl(e) = 0$. This means that it may be necessary to remove some repeated vertices from the above list.

Figure 7 shows an example of the results of Algorithm 1. The computed pen is shown as a bold polygon, superimposed on a $\frac{1}{2}$ unit grid along with the outline of the brush shape to be approximated. This is the same brush shape as shown in Figure 6b. It has $(x_1, y_1) = (.48, 3.79)$ and $(x_2, y_2) = (4.25, 1.01)$.

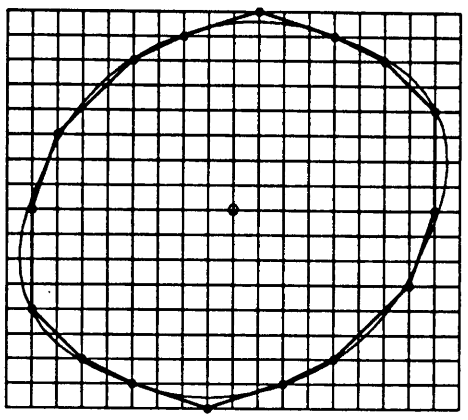

Fig. 7. A polygonal pen generated by Algorithm 1.

Table 1. A trace of Algorithm 1 applied to the example of Figure 7.

$w(p_l)$	$rl(p_l)$	$z(p)$	$ll(p_r)$	$w(p_r)$	$(\bar{u},\bar{v})$	δ	$ll(q_l)$	$rl(q_l)$
(1,0)	4.5	(4.0,−4.0)	5.0	(0,1)	(1,1)	3.0	1.5	1.5
(1,0)	1.5	(1.0,−4.0)	1.5	(1,1)	(2,1)	1.0	.5	.5
(1,0)	.5	(0.0,−4.0)	.5	(2,1)	(3,1)	.5	.5	0.0
(1,0)	0.0	(−.5,−4.0)	.5	(3,1)	(4,1)	0.0		
(3,1)	0.0	(1.0,−3.5)	0.0	(2,1)	(5,2)	0.0		
(2,1)	.5	(2.0,−3.0)	.5	(1,1)	(3,2)	0.0		
(1,1)	1.5	(4.0,−1.0)	2.0	(0,1)	(1,2)	.5	0.0	.5
(1,1)	1.0	(3.5,−1.5)	0.0	(1,2)	(2,3)	0.0		
(1,2)	.5	(4.0, −.5)	1.5	(0,1)	(1,3)	.5	.5	0.0
(1,2)	0.0	(3.5,−1.5)	.5	(1,3)	(2,5)	0.0		
(1,3)	0.0	(4.0, 0.0)	1.0	(0,1)	(1,4)	0.0		
(0,1)	3.0	(4.0, 4.0)	3.5	(−1,0)	(−1,1)	2.0	.5	1.5
(0,1)	1.0	(4.0, 2.0)	.5	(−1,1)	(−1,2)	0.0		
(−1,1)	1.5	(2.0, 4.0)	1.5	(−1,0)	(−2,1)	1.0	0.0	1.0
(−1,1)	.5	(3.0, 3.0)	0.0	(−2,1)	(−3,2)	0.0		
(−2,1)	1.0	(1.0, 4.0)	.5	(−1,0)	(−3,1)	.5	0.0	.5
(−2,1)	.5	(2.0, 3.5)	0.0	(−3,1)	(−5,2)	0.0		
(−3,1)	.5	(.5, 4.0)	0.0	(−1,0)	(−4,1)	0.0		

Table 1 shows how Algorithm 1 works on this example. The first seven columns show the state at the end of successive invocations of Step 2 of the algorithm, and the following two lemmas explain the meaning of the point $\bar{z}$ that is used in that step. The first lemma gives a basic property of convex sets, and the second lemma uses this to show how $\bar{z}$ is placed. Note that $\bar{z}$ is a point of support of a convex set S in a direction $(\bar{u},\bar{v})$ if and only if $\bar{z}$ lies on a directed supporting line for S whose direction is $(\bar{u},\bar{v})$.

Lemma 4.1.2. *Let $S \subseteq \mathbb{R}^2$ be a convex set; let H be an open halfplane with boundary $B(H)$; let g be a linear function from $\mathbb{R}^2$ to $\mathbb{R}$, not constant on $B(H)$; and let A be a point on $S \cap B(H)$ where g is maximized. If there exists a point $C \in S \cap H$ such that $g(C) \geq g(A)$, then the maximum of g on $S \setminus H$ occurs at A.*

If C can be chosen so that $g(C) > g(A)$ then the maximum of g on S occurs only at points in $S \cap H$.

Proof. Let f be a linear function from $\mathbb{R}^2$ or $\mathbb{R}$ such that $H = \{ P \mid f(P) > 0 \}$, and suppose that there exists a point $P \in S$ such that $f(P) \leq 0$ and $g(P) > g(A)$. If there is a point $C \in S$ such that $f(C) > 0$ and $g(C) \geq g(A)$, then there is a point Q on the line segment joining P to C such that $f(Q) = 0$. The linearity of g implies that $g(Q) > g(A)$, and the convexity of S implies that $Q \in S$. Since this contradicts our choice of A, the maximum of g on $S \cap \{ P \mid f(P) \leq 0 \} = S \setminus H$ must occur at A.

If $g(C) > g(A)$ and the maximum of g on S occurs at D, then $g(D) \geq g(C) > g(A)$. Thus by the result just proved, $D \notin S \setminus H$, hence $D \in S \cap H$. ∎

Lemma 4.1.3. *Let $L_p = \bar{L}(p_l) \cap \bar{L}(p_r)$ at the end of Step 2 of Algorithm 1. Then either $\bar{z}$ is a point of support of $\mathcal{B} \cap L_p$ in the direction $(\bar{u}, \bar{v})$, or $\mathcal{B}$ contains a point z_0 on the boundary $B(R_p)$ where $R_p = \bar{R}(p_l) \cap \bar{R}(p_r)$. Furthermore, if $\mathcal{B}$ intersects R_p then $\delta \leq 0$ at the end of Step 2.*

Proof. If $\mathcal{B}(\bar{u}, \bar{v}) = z(p)$, then either we immediately set $\delta \leftarrow 0$ and go on to Step 3, or we set $\bar{z} = z(p)$ and then compute $\delta = 0$. Letting $z_0 = z(p)$ satisfies the lemma.

If $\mathcal{B}(\bar{u}, \bar{v}) \in L_p \setminus \{z(p)\}$ then $\mathcal{B}$ does not intersect R_p. Since $\bar{z} \in L_p$, the rest of Step 2 does not change $\bar{z}$ and $\bar{z}$ is the required supporting point.

Otherwise we can assume without loss of generality that $\mathcal{B}(\bar{u}, \bar{v}) \in R(p_r)$. If either $\mathcal{B}\big(u(p_l), v(p_l)\big)$ or $\mathcal{B}\big(u(p_r), v(p_r)\big)$ belongs to R_p or $\mathcal{B}(\bar{u}, \bar{v}) \in R_p$, then $\mathcal{B} \cap R_p$ contains some point z_1. Since $(0,0) \in \mathcal{B} \setminus R_p$, the convexity of $\mathcal{B}$ allows us to construct a point z_0 between $(0,0)$ and z_1 such that $z_0 \in \mathcal{B} \cap B(R_p)$.

We are left with the case where $\mathcal{B}(\bar{u}, \bar{v}) \in R(p_r) \cap L(p_l)$ and the algorithm does not immediately skip on to Step 3. Let $H = R(p_r)$, and let $g(P) = \big\langle P, (\bar{v}, -\bar{u}) \big\rangle$ so that maximizing $g(P)$ on a convex set locates a point of support in the direction $(\bar{u}, \bar{v})$. Thus if we let $S = \mathcal{B}$, $\bar{z} = I_1(\mathcal{B}, p_r)$ satisfies the requirements for the point A in Lemma 4.1.2, and $\mathcal{B}(\bar{u}, \bar{v})$ satisfies the requirements for C. Lemma 4.1.2 shows that the maximum of g on $\mathcal{B} \setminus R(p_r)$ must occur at $\bar{z}$. Hence $\bar{z}$ is a point of support for this set in the direction $(\bar{u}, \bar{v})$. If $\bar{z} \in L_p$ then $\bar{z}$ clearly is also a point of support for $\mathcal{B} \cap L_p$; otherwise $\bar{z}$ is the point z_0 required by the lemma.

We must now show that δ is not set positive when $\mathcal{B}(\bar{u}, \bar{v}) \in R(p_r)$ and $\mathcal{B} \cap R_p$ is nonempty. Since $\big\langle (-\bar{v}, \bar{u}), \bar{z} - z(p) \big\rangle \leq 0$ for $\bar{z} \in R_p$, it is sufficient to show that either $\mathcal{B}\big(u(p_l), v(p_l)\big) \in R_p$ or $I_1(\mathcal{B}, p_r) \in R_p$. If $\mathcal{B}$ intersects both R_p and $L(p_r) \cap \bar{R}(p_l)$, then it must contain some point on the dividing line $\ell(p_r) \cap \bar{R}(p_l)$, and thus $I_1(\mathcal{B}, p_r) \in \ell(p_r) \cap \bar{R}(p_l)$ as required. If $\mathcal{B} \cap \bar{R}(p_l)$ does not intersect $L(p_r)$, then $\mathcal{B} \cap \bar{R}(p_l) \subseteq R_p$. Hence $\mathcal{B}\big(u(p_l), v(p_l)\big) \in R_p$ as required. ∎

When $\delta > 0$ in Step 2 so that a new edge can be inserted, the last two columns of Table 1 show the "lengths" computed for it in Step 5. Notice that $rl(q_l) + ll(q_l) = \delta$ whenever these lengths are given. The meaning of this is that δ is the total "length" of the new edge in the sense that the displacement between opposite ends of the new edge is $\delta \cdot (\bar{u}, \bar{v})$. The two length fields give similar displacements between the ends of the new edge and a special point $\zeta(q_l)$ on q_l:

$$\zeta(q_l) - z(p) = ll(q_l)\cdot(\bar{u},\bar{v}) \quad \text{and} \quad z(q) - \zeta(q_l) = rl(q_l)\cdot(\bar{u},\bar{v}).$$

The following lemma shows that for any edge e, the point $\zeta(e)$ is actually between the endpoints of e.

Lemma 4.1.4. *For all edges e during the entire course of Algorithm 1, $ll(e)$ and $rl(e)$ are nonnegative integer multiples of $\frac{1}{2}$.*

Proof. The lemma clearly holds after Step 1. When $ll(q_r)$ and $rl(p_l)$ are updated in Step 4, δ is an integer multiple of $\frac{1}{2}$ no greater than $\min\bigl(ll(p_l), rl(q_r)\bigr)$. The only other time the ll and rl fields are updated is in Step 5 where $ll(q_l)$ is set to 0 or δ unless $0 < \delta < \min\bigl(rl(p_l), ll(p_r)\bigr)$ at end of the previous invocation of Step 2. In this case $\bigl|\delta - \bigl\langle(-\bar{v},\bar{u}), \bar{z} - z(p)\bigr\rangle\bigr| \le \frac{1}{4}$ at the end of Step 2, and thus by (4.1.1), $\bigl\langle(-\bar{v},\bar{u}), \bar{z} - z(p)\bigr\rangle \le \frac{1}{4}$ in Step 5. Since Lemma 4.1.3 shows that $\bar{z} \in \bar{L}(p_l)\cap\bar{L}(p_r)$ at the end of Step 2, we have

$$A\bigl(\bar{z} - z(p)\bigr)^T = \begin{pmatrix} a \\ b \end{pmatrix} \quad \text{where} \quad A = \begin{pmatrix} -v(p_l) & u(p_l) \\ -\bar{v} & \bar{u} \end{pmatrix}, \quad a \ge 0, \quad b \le \tfrac{1}{4}.$$

Now (4.1.1) shows that $\det A = 1$ and $0 < u(p_l)\bar{u} + v(p_l)\bar{v} < \bar{u}^2 + \bar{v}^2$, hence

$$(\bar{u},\bar{v})\bigl(\bar{z} - z(p)\bigr)^T = (\bar{u},\bar{v})\begin{pmatrix} \bar{u} & -u(p_l) \\ \bar{v} & -v(p_l) \end{pmatrix}\begin{pmatrix} a \\ b \end{pmatrix} \ge -\frac{u(p_l)\bar{u} + v(p_l)\bar{v}}{4} > -\frac{\bar{u}^2 + \bar{v}^2}{4}$$

so that $ll(q_l) \ge 0$ as required.

A similar argument suffices for

$$rl(q_l) = -r\left(\frac{\langle(\bar{u},\bar{v}), \bar{z} - z(q)\rangle}{\bar{u}^2 + \bar{v}^2}\right),$$

except we use the $(-\bar{v},\bar{u})$ and $\bigl(-v(q_r), u(q_r)\bigr)$ components of $\bar{z} - z(q)$ and solve for the $(\bar{u},\bar{v})$ component. Thus $rl(q_l) = -r(x)$ where $x < \frac{1}{4}$. ∎

Corollary 4.1.5. *If Step 2 of Algorithm 1 sets $\bar{z}$ and δ such that $\bar{z} \in L_p = \bar{L}(p_l)\cap\bar{L}(p_r)$ and $\delta \le 0 < \min\bigl(rl(p_l), ll(p_r)\bigr)$, then $\bigl|\bigl\langle(\bar{u},\bar{v}), \bar{z} - z(p)\bigr\rangle\bigr| \le \frac{1}{4}(\bar{u}^2+\bar{v}^2)$ and $0 \le \bigl|\bigl\langle(-\bar{v},\bar{u}), \bar{z} - z(p)\bigr\rangle\bigr| \le \frac{1}{4}$.*

Proof. Let $w_{\text{rot}}(e)$ denote $\bigl(-v(e), u(e)\bigr)$ for edge nodes e. Since $\bar{z} \in L_p$, we have $\bigl\langle w_{\text{rot}}(p_l), \bar{z} - z(p)\bigr\rangle \ge 0$ and $\bigl\langle w_{\text{rot}}(p_r), \bar{z} - z(p)\bigr\rangle \ge 0$. Thus $\delta_0 \ge 0$, where $\delta_0 = \bigl\langle(-\bar{v},\bar{u}), \bar{z} - z(p)\bigr\rangle$. The requirement that $|\delta_0| \le \frac{1}{4}$ immediately follows from $r_d(\delta_0) = \delta \le 0$.

In the proof of Lemma 4.1.4, we showed the following: if $\bigl\langle(-\bar{v},\bar{u}), \bar{z}-z(p)\bigr\rangle \le \frac{1}{4}$ and $\bigl\langle w_{\text{rot}}(p_l), \bar{z}-z(p)\bigr\rangle \ge 0$, then $\bigl\langle(\bar{u},\bar{v}), \bar{z}-z(p)\bigr\rangle > -(\bar{u}^2+\bar{v}^2)/4$; and if $\bigl\langle(-\bar{v},\bar{u}), \bar{z} - z(q)\bigr\rangle \le \frac{1}{4}$ and $\bigl\langle w_{\text{rot}}(q_r), \bar{z} - z(q)\bigr\rangle \ge 0$, then $\bigl\langle(\bar{u},\bar{v}), \bar{z} - z(q)\bigr\rangle < (\bar{u}^2 + \bar{v}^2)/4$. Letting $q = p$ completes the proof of the corollary. ∎

When the algorithm creates a new edge and implicitly places $\zeta(q_l)$, this point is retained on the polygon throughout the rest of the computation. For this reason, we shall refer to $\pm\zeta(q_l)$ as *retention points.* The purpose of retention points is to guarantee that when new edges are added to the polygon, the width in direction $(-\bar v, \bar u)$ is never allowed to decrease.

In general there is a retention point $\zeta(e_i)$ for each edge e_i in the data structure, but some of these points may coincide. Whenever a new retention point is placed on the $(\bar u, \bar v)$ edge in Step 5, it is placed as close as possible to $\bar z$ subject to the constraint that both components must be integer multiples of $\frac{1}{2}$. The following lemma states some important consequences of this placement strategy.

Lemma 4.1.6. *At the end of Step 5 in Algorithm 1, $\langle(-\bar v,\bar u),\zeta(q_l)-\bar z\rangle < \frac14$. Furthermore if $\zeta(q_l) \notin \{\zeta(p_l),\zeta(q_r)\}$, then $|\langle(\bar u,\bar v),\zeta(q_l)-\bar z\rangle| \le (\bar u^2+\bar v^2)/4$ and $|\langle(-\bar v,\bar u),\zeta(q_l)-\bar z\rangle| \le \frac14$.*

Proof. If Step 2 of Algorithm 1 does not set $\bar z$, or if at the end of Step 2, $\bar z \notin L_p$, where L_p is as defined in Lemma 4.1.3, then Lemma 4.1.3 shows that $\delta \le 0$ and thus Step 2 is immediately repeated. Otherwise if we let $\bar\delta = r_d(\langle(-\bar v,\bar u),\, \bar z - z(p)\rangle)$, then $\delta \le \bar\delta$ at the end of Step 2, and either $\delta \in \{rl(p_l), ll(p_r)\}$ or $\delta = \bar\delta$.

It follows from (4.1.1) that Step 4 causes both endpoints of the new edge to lie on the line $\langle(-\bar v,\bar u),(x,y)-P\rangle = \delta$, where P is the vertex $z(p)$ from Step 2. Hence

$$\langle(-\bar v,\bar u),\,\zeta(q_l)-P\rangle = \delta \le \bar\delta < \langle(-\bar v,\bar u),\,\bar z - P\rangle + \tfrac14$$

and $\langle(-\bar v,\bar u),\zeta(q_l)-\bar z\rangle < \frac14$ as required.

Now let $A=\zeta(p_l)$ and $B=\zeta(p_r)$ at the end of Step 2, and let $C=\zeta(q_l)$ be the retention point on the newly added edge at the end of the following invocation of Step 5. If $\delta = rl(p_l)$ after Step 2 then $rl(p_l)=0$ in Step 5, and thus $C=A$; otherwise if $\delta = ll(p_r)$ after Step 2, then $C=B$ after Step 5. Either way $\zeta(q_l) \in \{\zeta(p_l),\zeta(q_r)\}$ at the end of Step 5.

In the remaining case $\delta=\bar\delta$, and (4.1.2) is used to set $ll(q_l)$. Letting P be the vertex $z(p)$ from Step 2,

$$\langle(-\bar v,\bar u),\,\zeta(q_l)-P\rangle = \delta = \bar\delta \ge \langle(-\bar v,\bar u),\bar z - P\rangle - \tfrac14$$

and $\frac14 > \langle(-\bar v,\bar u),\zeta(q_l)-\bar z\rangle \ge -\frac14$. The lemma also requires $|\langle(\bar u,\bar v),\zeta(q_l)-\bar z\rangle| \le (\bar u^2+\bar v^2)/4$, but this follows immediately from (4.1.2). ∎

We still need three more lemmas dealing with properties of Algorithm 1. The first of these gives conditions under which more than one point on an edge must be retained on the computed pen.

Lemma 4.1.7. *Let v_l and e be consecutive nodes in the data structure at some point in the execution of Algorithm 1, such that $z_l = z(v_l)$ and z_r are the endpoints of the edge e and $z_r - z_l = (ll(e)+rl(e))\cdot w(e)$. If the intersection of $\mathcal{B}$ with the line segment z_lz_r is a segment AB, then the pen $\mathcal{P}$ computed from $\mathcal{B}$ by Algorithm 1 contains points $A' = A + aw(e)$ and $B' = B + bw(e)$, where $|a| \le \frac14$ and $|b| \le \frac14$.*

Proof. Let $l = ll(e)+rl(e)$ and choose a_0 so that $A = z_l + a_0w(e)$. Clearly $0 \le a_0 \le l$. Now let $a' = r(a_0)$, let $A' = z_l + a'w(e)$, and let $a = a' - a_0$ so that

$|a| \le \frac{1}{4}$. Since $0 \le r(a_0) \le l$ when $0 \le a_0 \le l$, we know that A is on the edge denoted by e.

In order to show that $A' \in \mathcal{P}$, it suffices to prove that for $c \ge 0$, if $p_l = e$ in Step 2 and $A' - z(p) = -cw(p_l)$, or if $p_r = e$ and $A' - z(p) = cw(p_r)$, then $c \ge \delta$. In both cases, either $\delta \le 0$ or $A \in \mathcal{B} \cap L_p$ and Lemma 4.1.3 shows that $C(\bar{z}) \le C(A)$, where $C(z) = \langle(-\bar{v}, \bar{u}), z\rangle$ for all z. Thus

$$\begin{aligned}\delta \le r_d\bigl(C(\bar{z} - z(p))\bigr) &\le r_d\bigl(C(A - z(p))\bigr) < C\bigl(A - z(p)\bigr) + \tfrac{1}{4}\\ &= C\bigl(A + aw(e) - z(p)\bigr) + \tfrac{1}{4} - aC\bigl(w(e)\bigr) = c + \tfrac{1}{4} - aC\bigl(w(e)\bigr) \le c + \tfrac{1}{2}.\end{aligned}$$

We know that δ is an integer multiple of $\frac{1}{2}$; if we can show that c is also, then we have $c \ge \delta$ as required. This is true initially since the difference between A' and either endpoint of e is a half integer multiple of $w(e)$. The rest of the algorithm preserves this invariant because the endpoints of e are only updated in Step 4 where $2\delta \in \mathbb{Z}$.

Since we have used no properties of A other than that it lies on the segment AB, the same argument also proves the existence of the required point $B' \in \mathcal{P}$. ∎

Lemma 4.1.8. *Suppose there is a point $(x, y) \in \mathcal{B}$ such that $|x| > x(v_2)$ and $|y| > y(v_2)$, where $(x(v_2), y(v_2))$ is as computed in Step 1 of Algorithm 1. Then $d((x, y), \mathcal{P}) < \sqrt{2}/4$, where d denotes Euclidean distance and $\mathcal{P}$ is the pen computed from $\mathcal{B}$ by Algorithm 1.*

Proof. Since $|x| \le x_2 < x(v_1) + \frac{1}{4} = x(v_2) + \frac{1}{4}$ and $|y| \le y_1 < y(v_2) + \frac{1}{4} = \frac{1}{4} - y(v_1)$ in Step 1, it follows that (x, y) is at a distance of less than $\sqrt{2}/4$ from some point $\pm z(v_i)$, where $i = 1$ or 2. Thus we need only show that $z(v_i) \in \mathcal{P}$.

We can always choose i and $(x_0, y_0) = \pm(x, y)$ so that either $i = 1$, $x_0 > x(v_1)$, and $y_0 < y(v_1)$, or $i = 2$, $x_0 > x(v_2)$, and $y_0 > y(v_2)$. If $i = 1$ then $(x_0, y_0) \in \mathcal{B} \cap R_p$ when $(\bar{u}, \bar{v}) = (1, 1)$. Hence Lemma 4.1.3 shows that p passes v_1, and thus $z(v_1) \in \mathcal{P}$. If $i = 2$ then a similar argument shows that $z(v_2) \in \mathcal{P}$. ∎

Lemma 4.1.9. *Consider the state of Algorithm 1 at the beginning of Step 2; let z_q be the vertex following $z_p = z(p)$; and let AB be the intersection of $\mathcal{B}$ and the segment $z_p z_q$, where $A = (x_A, y_A)$, $B = (x_B, y_B)$, and $B - A$ is a positive multiple of $w(p_r)$. Let $\mathcal{P}$ be the pen computed from $\mathcal{B}$ by Algorithm 1; and let $H_p = \{(x, y) \mid C_p(x, y)\}$ and $H_q = \{(x, y) \mid C_q(x, y)\}$, where*

$$C_p(x,y) = \begin{cases} y < y_A & \text{if } u(p_r) \ge 0;\\ x > x_A & \text{if } u(p_r) < 0;\end{cases} \qquad C_q(x,y) = \begin{cases} x > x_B & \text{if } u(p_r) > 0;\\ x < x_B & \text{if } w(p_r) = (-1, 0);\\ y > y_B & \text{otherwise.}\end{cases}$$

If $\mathcal{B} \cap R(p_r)$ intersects H_p, then $z_p \in \mathcal{P}$; if $\mathcal{B} \cap R(p_r)$ intersects H_q, then $z_q \in \mathcal{P}$.

Proof. Let $H = R(p_r)$, and let g be such that $H_p = \{P \mid g(P) > 0\}$; e.g., we may take $g(x, y) = y_A - y$ if $u(p_r) \ge 0$. Let $(u_0, v_0) = (1, 0)$ if $u(p_r) \ge 0$, and $(u_0, v_0) = (0, 1)$ if $u(p_r) < 0$, so that the maximum of g on $\mathcal{B}$ occurs at $\mathcal{B}(u_0, v_0)$. Applying Lemma 4.1.2 with $S = \mathcal{B}$ shows that if $\mathcal{B} \cap R(p_r)$ intersects H_p, then $\mathcal{B}(u_0, v_0) \in R(p_r)$.

If $\mathcal{B}(u_0, v_0) \notin L(p_l)$ then $\mathcal{B} \cap R_p$ is nonempty, and thus Lemma 4.1.3 shows that $\delta \leq 0$ at the end of the current invocation of Step 2. This leaves $z_p \in \mathcal{P}$ as required.

It remains to be shown that $z_p \in \mathcal{P}$ when $\mathcal{B} \cap R(p_r)$ intersects H_p and $\mathcal{B}(u_0, v_0) \in R(p_r) \cap L(p_l)$. The direction (u_0, v_0) has been chosen so that the dot products of $w(p_l)$ and $w(p_r)$ with (u_0, v_0) and $(-v_0, u_0)$ are all nonnegative. Thus $\mathcal{B}(u_0, v_0) \in R(p_r) \cap L(p_l) \subseteq (Q + z_p)$, where

$$Q = \{\, (x, y) \mid u_0 x + v_0 y \geq 0 \text{ and } -v_0 x + u_0 y \geq 0 \,\}.$$

Let e_i be the edge such that $w(e_i) = (u_0, v_0)$. Step 1 creates a retention point $\zeta(e_i)$ that is obtained from $\mathcal{B}(u_0, v_0)$ by rounding its coordinates to half integers. Since z_p has half integer coordinates, such rounding maps $Q + z_p$ into a subset of itself. Thus $\zeta(e_i) \in Q + z_p$. The difference $z_p - \zeta(e_i)$ is a sum of positive multiples of $w(e)$ for edges e between e_i and p_r. Since $w(e) \in Q$ for all such e, it follows that $z(p) - \zeta(e_i) \in Q$. Hence $\zeta(e_i) = z_p$ and $z_p \in \mathcal{P}$.

To show that $z_q \in \mathcal{P}$ when $\mathcal{B} \cap R(p_r)$ intersects H_q, use B in place of A, modify g so that $H_q = \{\, P \mid g(P) > 0 \,\}$, and redefine (u_0, v_0) as follows:

$$(u_0, v_0) = \begin{cases} (0,1) & \text{if } u(p_r) > 0; \\ (0,-1) & \text{if } w(p_r) = (-1,0); \\ (-1,0) & \text{otherwise.} \end{cases}$$

An argument similar to the above allows us to assume that $\mathcal{B}(u_0, v_0) \in R(p_r) \cap L(q_r)$, where q_r is the edge following p_r in the data structure. (If there is no such edge, then $w(p_r) = (-1, 0)$ and we can replace $L(q_r)$ with $-L(e)$ where e is the first edge such that $u(e) > 0$.) Replacing z_p with z_q in the above argument shows that $z_q = \zeta(e_i)$, where $w(e_i) = (u_0, v_0)$. ∎

We are almost ready to use the above lemmas to get a bound on the error $E(\mathcal{P}, \mathcal{B})$, where $\mathcal{P}$ is the pen computed from $\mathcal{B}$ by Algorithm 1. The proof will be based on the fact that if a point P is at a distance d from a convex region S, then there is a separating line for P and S that is at a distance of at least d from P. Specifically, there is a directed line ℓ such that no point of S is to the right of ℓ, and P is *d to the right of ℓ*; i.e., P is to the right of ℓ and the distance between P and ℓ is at least d. The line ℓ is a called a *directed d separating line* for P and S. For any point P and convex region S, there is a directed d separating line with direction (u, v) if and only if P is d to the right of ℓ, where ℓ is the supporting line ℓ of S with direction (u, v). Thus the distance from a convex set S to a point P is the maximum d such that P is d to the right of ℓ for some directed supporting line ℓ of S.

Lemma 4.1.10. *Let ℓ_1 and ℓ_2 be directed supporting lines for the convex hull of a set of three distinct, non-collinear points $\{P_1, P_2, Q\}$ such that ℓ_i contains P_i and has direction $w_i = (u_i, v_i)$ for $i = 1, 2$. Suppose that there is a directed separating line for Q and the segment P_1P_2 with some direction $w_3 = (u_3, v_3)$ such that $u_1 v_3 > u_3 v_1$ and $u_3 v_2 > v_3 u_2$. The direction of any other directed separating* *line for Q and the segment P_1P_2 must be a nonnegative linear combination of either w_1 and w_3 or of w_2 and w_3.*

Proof. Let $(a_1, a_2, a_3) \neq (0,0,0)$ be such that $a_1w_1 + a_2w_2 + a_3w_3 = (0,0)$ and $a_1 \geq 0$. Taking the dot product with $(-v_3, u_3)$ yields $a_1\alpha + a_2\beta = 0$ where $\alpha < 0$ and $\beta > 0$. Thus a_1 and a_2 are both positive. (They cannot both be zero because $w_3 \neq (0,0)$.)

If $c_{ji} = \langle P_j - Q, (-v_i, u_i)\rangle$ for $i = 1,2,3$ and $j = 1,2$, then the given information can be summarized as follows:

$$\begin{array}{lll} c_{11} \leq 0, & c_{12} \geq c_{22}, & c_{13} > 0, \\ c_{21} \geq c_{11}, & c_{22} \leq 0, & c_{23} > 0. \end{array} \tag{4.1.3}$$

Let $w_4 = (u_4, v_4) = b_1w_1 + b_3w_3$ be the direction of a separating line for Q and $\{P_1, P_2\}$ so that

$$b_1c_{j1} + b_3c_{j3} = \langle P_j - Q, (-v_4, u_4)\rangle > 0 \quad \text{for } j = 1,2.$$

If $b_1 \geq 0$ then $b_1c_{11} \leq 0$, hence $b_3c_{13} > 0$ and therefore $b_3 > 0$. Similarly if $w_4 = b_2'w_2 + b_3'w_3$ and $b_2' \geq 0$, then $b_2'c_{22} \leq 0$ and therefore $b_3'c_{23} > 0$ and $b_3' > 0$. Substituting $w_1 = -(a_2/a_1)w_2 - (a_3/a_1)w_3$ shows that $b_2' > 0$ when $b_1 < 0$. Thus either $b_1, b_3 \geq 0$ or $b_2', b_3' \geq 0$. ∎

Lemma 4.1.11. *Let (u,v) and $(\bar{u},\bar{v})$ be reduced rational directions such that $u\bar{v} - v\bar{u} = 1$, $\bar{u}, v \geq 0$, $u, \bar{v} > 0$, and either $(u,v) = (1,0)$ or ($\bar{u} \geq u$ and $\bar{v} \geq v$). If $(x,y) \in \mathbb{R}^2$ satisfies $-y \leq \bar{v}/4$ and $\bar{v}x - \bar{u}y \leq \frac{1}{4}$, and if (u',v') is a nonnegative linear combination of (u,v) and $(\bar{u},\bar{v})$, then*

$$\frac{v'x - u'y}{\sqrt{u'^2 + v'^2}} \leq \frac{\sqrt{2}}{4}.$$

Proof. Assume without loss of generality that $(u',v') = (1-\alpha)\cdot(\bar{u},\bar{v}) + \alpha\cdot(u,v)$, where $0 \leq \alpha \leq 1$. Thus $v'x - u'y = (1-\alpha)(\bar{v}x - \bar{u}y) + \alpha(vx - uy)$. Since

$$\bar{v}(vx - uy) = v(\bar{v}x - \bar{u}y) + y(v\bar{u} - u\bar{v}) = v(\bar{v}x - \bar{u}y) - y \leq v/4 + \bar{v}/4,$$

we have $vx - uy \leq \frac{1}{4}(1 + v/\bar{v})$, and thus

$$v'x - u'y = (\bar{v}x - \bar{u}y)(1-\alpha) + (vx - uy)\alpha \leq \tfrac{1}{4}(1-\alpha) + \tfrac{1}{4}(1 + v/\bar{v})\alpha = \tfrac{1}{4}(1 + \alpha v/\bar{v}).$$

It remains to be shown that $1 + \alpha v/\bar{v} \leq \sqrt{2}\sqrt{u'^2 + v'^2}$.

First consider the case where $(u,v) = (1,0)$ so that $1 + \alpha v/\bar{v} = 1$. We then need $\sqrt{u'^2 + v'^2} \geq 1/\sqrt{2}$, but this follows directly from the fact that $u' + v' = (1-\alpha)(\bar{u} + \bar{v}) + \alpha(u + v) \geq 1$.

Otherwise $u' = u + (1-\alpha)(\bar{u} - u) \geq u$ and $v' = v + (1-\alpha)(\bar{v} - v) \geq v$, so that $2 \leq u^2 + v^2 \leq u'^2 + v'^2$. Hence

$$1 + \alpha v/\bar{v} \leq 2 \leq \sqrt{2}\sqrt{u^2 + v^2} \leq \sqrt{2}\sqrt{u'^2 + v'^2}$$

as required. ∎

Theorem 4.1.12. *If $\mathcal{P}$ is a pen computed by Algorithm 1 to approximate a symmetrical convex brush $\mathcal{B}$, then $E(\mathcal{P},\mathcal{B}) \le \frac{3}{8}$.*

Proof. We have seen that Algorithm 1 maintains a list of vertices that can be extended to define a symmetrical convex polygon. At any stage in the computation, the vertices in the data structure and their negatives (in order) constitute the vertices of this polygon. The computation defines a sequence of such polygons $\mathcal{P}_1, \mathcal{P}_2, \ldots, \mathcal{P}_k$ for which $\mathcal{P}_{i+1} \subseteq \mathcal{P}_i$ for $i < k$, and $\mathcal{P}_k$ is the computed pen $\mathcal{P}$. Any such polygon $\mathcal{P}_i$ has directed edges $\ell(e)$ and $-\ell(e)$ for each edge e in the data structure at the point in computation corresponding to $\mathcal{P}_i$. The direction of $-\ell(e)$ is the negative of that of $\ell(e)$.

Now suppose that there is some point $z \in \mathcal{B}$ such that $d(z,\mathcal{P}) > \sqrt{2}/4$, and consider the first i such that $z \notin \mathcal{P}_i$. Either there is a unique directed edge of $\mathcal{P}_i$ that z is to the right of, or $i = 1$ and Lemma 4.1.8 shows that $d(z,\mathcal{P}) < \sqrt{2}/4$, contradicting the assumption.

Since z is to the right of $\ell(e)$ if and only if $-z$ is to the right of $-\ell(e)$ for any edge e, we may assume that the unique directed edge is $\ell(e)$ for some edge node e, where $z \in R(e)$ and e is not the first node in the data structure. Let A, B, A', and B' be the points on $\ell(e)$ defined in Lemma 4.1.7, and let ℓ^- and ℓ^+ be the directed edges of $\mathcal{P}_i$ adjacent to $\ell(e)$; i.e., their directions should be as close as possible to $w(e)$. We shall use (u^-, v^-) for the direction of ℓ^-, and similarly (u^+, v^+) for the direction of ℓ^+.

Letting P_1 and P_2 be the retention points on ℓ^- and ℓ^+, and letting $Q = z$, Lemma 4.1.10 shows that the direction of any separating line for z and $\mathcal{P}$ must be a nonnegative linear combination of (u^-, v^-) and $w(e)$, or of $w(e)$ and (u^+, v^+).

If $i = 1$ then $w(e) = (0,1)$ or $w(e) = (-1,0)$, and there is clearly no $\frac{1}{4}$ separating line for z and $\ell(e)$ with direction $w(e)$. If $i > 1$ then Lemma 4.1.3 shows that z is to the left of the $w(e)$ directed line through the point $\bar{z}$ from the last invocation of Step 2 before edge e is created. Lemma 4.1.6 shows that $\ell(e)$ is a distance of less than $1/\bigl(4\sqrt{u(e)^2 + v(e)^2}\,\bigr)$ to the left of the directed line through $\bar{z}$. Hence there can be no $1/\bigl(4\sqrt{u(e)^2 + v(e)^2}\,\bigr)$ separating line with direction $w(e)$.

We shall now find a point $(x_0, y_0) \in \ell(e) \cap \mathcal{P}$ such that the following holds for $(x,y) = z$: If $u(e) \ge 0$ then $y \ge y_0 - v(e)/4$; if $u(e) < 0$ then $x \le x_0 - u(e)/4$. If $u(e) \ge 0$ then Lemma 4.1.11 applies to the vector $z - (x_0, y_0)$ and the directions (u^-, v^-) and $w(e)$; if $u(e) < 0$ then it is necessary to rotate these three vectors $90°$ clockwise before applying the lemma. Either way, the lemma allows us to conclude that there is no $\sqrt{2}/4$ separating line for z and $\{(x_0, y_0)\} \subset \mathcal{P}$ with a direction that is a nonnegative linear combination of (u^-, v^-) and $w(e)$.

Lemma 4.1.9 determines a point (x_0, y_0) that satisfies the above requirements: If $z \in H_p$ then z_p can serve as (x_0, y_0). Otherwise we can let $(x_0, y_0) = A'$, because $z \notin H_p$ and Lemma 4.1.7 shows that $A' - A = a w(e)$, where $|a| \le \frac{1}{4}$.

A similar argument shows that there can be no $\sqrt{2}/4$ separating line for z and $\mathcal{P}$ with a direction that is a nonnegative linear combination of $w(e)$ and (u^+, v^+): If $z \in H_q$ in Lemma 4.1.9 then let $(x_0, y_0) = z_q$, otherwise let $(x_0, y_0) = B'$. Lemma 4.1.11 applies as before except it is necessary to do a more complicated transformation: If $u(e) > 0$ then the lemma applies to (u,v), $(\bar{u},\bar{v})$, and (x,y)

where $(u^+, v^+) = (v, u)$, $w(e) = (\bar{v}, \bar{u})$, and $z - (x_0, y_0) = (-y, -x)$; if $u(e) \le 0$ then the lemma applies when $(u^+, v^+) = (-u, v) = (-1, 0)$, $w(e) = (-\bar{u}, \bar{v})$, and $z - (x_0, y_0) = (x, -y)$.

We must also show that $\max_{z \in \mathcal{P}} d(z, \mathcal{B}) \le \frac{3}{8}$, or equivalently that $\mathcal{P} \subseteq \mathcal{B}'$, where if $\mathcal{B}'$ is the convex set formed by taking the envelope of $\mathcal{B}$ with respect to the circle $x^2 + y^2 \le \frac{9}{64}$. We shall show by induction that $\zeta(e) \in \mathcal{B}'$ for all edges e, and that $z(p) \in \mathcal{B}'$ whenever p advances from a vertex in Step 3. Thus all vertices of $\mathcal{P}$ belong to $\mathcal{B}'$, and hence so does their convex hull. Since $\mathcal{P}$ is a convex polygon, this completes the proof.

Step 1 initializes the retention points to $\big(r_d(x_1), r(y_1)\big)$, $\big(r(x_2), r_d(y_2)\big)$, and $\big(r(-x_1), r(-y_1)\big)$, all of which are within $\sqrt{2}/4$ of $\mathcal{B}$ and thus belong to $\mathcal{B}'$.

Consider the algorithm in Step 3 when p passes a vertex and $z(p) \notin \mathcal{B}$ or when a new retention point is about to be created. Let z be that vertex or retention point; let $\bar{\alpha}$ and α be the lengths of $(\bar{u}, \bar{v})$ and $w(p_l)$; and let θ be the angle between $w(p_l)$ and $(\bar{u}, \bar{v})$ as shown in Figure 8. Lemma 4.1.6 shows that when z is a retention point, the point $\bar{z}$ from Step 2 will differ from z by at most $\bar{\alpha}/4$ in its $(\bar{u}, \bar{v})$ component and $1/(4\bar{\alpha})$ in its $(-\bar{v}, \bar{u})$ component. Corollary 4.1.5 shows that this also holds when z is a passed vertex and $\bar{z} \in L_p$. When $\bar{z} \notin L_p$, Lemma 4.1.3 shows that $\mathcal{B}$ contains a point $z_0 \in B(R_p)$, where $B(R_p)$ is as defined in the lemma.

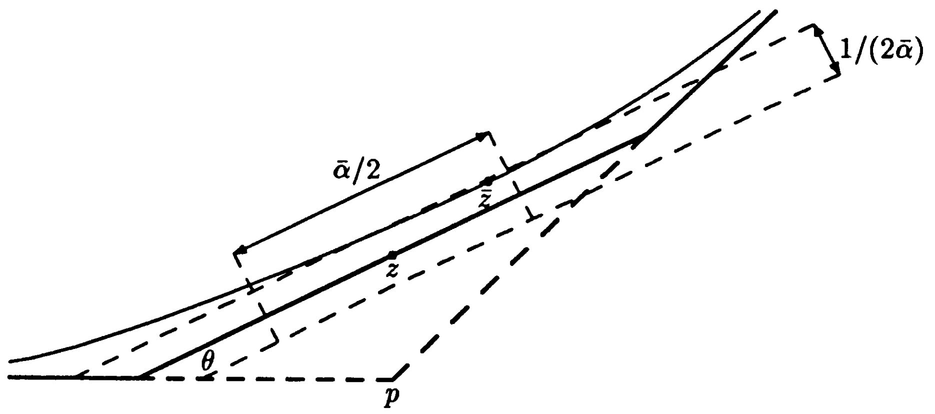

Fig. 8. A construction for showing why retention points lie close to $\mathcal{B}$.

Consider the case when the absolute values of the $(\bar{u}, \bar{v})$ and $(-\bar{v}, \bar{u})$ components of $z - \bar{z}$ are at most $\bar{\alpha}/4$ and $1/(4\bar{\alpha})$ respectively, and suppose there exists a separating line between z and $\mathcal{B}'$. The induction hypothesis implies that $\{\zeta(p_l), \zeta(p_r)\} \subseteq \mathcal{B}'$, so that applying Lemma 4.1.10 with $\ell_1 = \ell(p_l)$ and $\ell_2 = \ell(p_r)$ shows that any directed separating line for z and $\mathcal{B}'$ must have a direction that is between $w(p_l)$ and $w(p_r)$. Assume temporarily that its direction is between $w(p_l)$ and $(\bar{u}, \bar{v})$.

Since $\bar{z}$ belongs to an $\bar{\alpha}/2$ by $1/(2\bar{\alpha})$ rectangle centered on z, the component of $\bar{z} - z$ perpendicular to the separating line can be at most

$$f(\phi) = \frac{1}{4}\left(\frac{\cos\phi}{\bar{\alpha}} + \bar{\alpha}\sin\phi\right)$$

where ϕ is the angle between $(\bar{u}, \bar{v})$ and the separating line. Since $\bar{\alpha} > 1$, this function is monotone increasing for $0 < \phi < 45°$. Thus $f(\phi) < f(\theta)$, and (4.1.1) shows that

$$f(\theta) = \frac{1}{4}\left(\frac{\cos\theta}{\bar{\alpha}} + \frac{1}{\alpha}\right) = \frac{1}{4}\left(\frac{\sqrt{1-(\alpha\bar{\alpha})^{-2}}}{\bar{\alpha}} + \frac{1}{\alpha}\right).$$

This is at most $\frac{3}{8}$ because $\bar{\alpha} \geq \sqrt{2}$ and $\alpha \geq 1$: if $\alpha \geq \sqrt{2}$ then both terms are at most $\sqrt{2}/8$; if $\alpha = 1$ then the first term is at most $\frac{1}{8}$ and the second term is at most $\frac{1}{4}$. Thus $f(\phi) \leq \frac{3}{8}$ and the envelope of $\{\bar{z}\}$ with respect to $x^2 + y^2 \leq \frac{9}{64}$ must intersect the separating line, contradicting our assumption that $z \notin \mathcal{B}'$.

If there is a separating line between z and $\mathcal{B}'$ whose direction is between $(\bar{u}, \bar{v})$ and $w(p_r)$, then the above argument still applies if we use the length of $w(p_r)$ in place of α. Thus there cannot be such a separating line.

The remaining case is when $\mathcal{B}$ contains a point $z_0 \in B(R_p)$ and $z = z(p)$. Thus z is on one of the line segments $z_0\zeta(p_l)$ and $z_0\zeta(p_r)$, and the induction hypothesis implies that both endpoints of these line segments lie in $\mathcal{B}'$. By convexity, $z \in \mathcal{B}'$ as required. ∎

Algorithm 1 has been stripped down about as much as possible while still retaining the above theorem. It could be speeded up by avoiding null edges in the data structure, or by not computing $\mathcal{B}(\bar{u}, \bar{v})$ when $\mathcal{B}(u(p_l), v(p_l))$ or $\mathcal{B}(u(p_r), v(p_r))$ belongs to $\bar{R}(p_l) \cap \bar{R}(p_r)$ or when $\min(rl(p_l), ll(p_r)) = 0$. In addition, the quantities $\mathcal{B}(u(p_l), v(p_l))$, $\mathcal{B}(u(p_r), v(p_r))$, $I_2(\mathcal{B}, p_l)$, and $I_1(\mathcal{B}, p_r)$ could be saved in the data structure to avoid recomputing them.

It would also be possible to avoid finding $I_1(\mathcal{B}, p_r)$ and $I_2(\mathcal{B}, p_l)$ by always using $\bar{z} = \mathcal{B}(\bar{u}, \bar{v})$ even when this point is outside of the current pen polygon $\mathcal{P}_i$. This has not been done because this would allow the error $E(\mathcal{P}, \mathcal{B})$ to approach $\frac{1}{2}$ as shown in Figure 9. The algorithm works by slicing corners off of the polygon until it approximates $\mathcal{B}$, and the decision of how much to cut off should be based only the region $\mathcal{P} \setminus \mathcal{B}$ that we wish to remove. Thus $\bar{z}$ should be a supporting point for $\mathcal{B} \cap \mathcal{P}_i$ rather than a supporting point for $\mathcal{B}$.

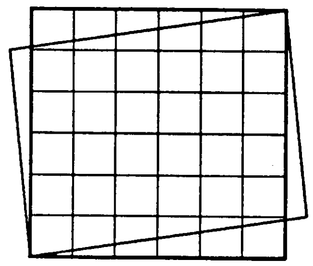

Fig. 9. A brush shape for which $E(\mathcal{P}, \mathcal{B}) \approx \frac{1}{2}$ if Step 2 of Algorithm 1 is simplified.

The most obvious complicating factor remaining in the algorithm is that it deals implicitly with retention points. Why not just have one "length" field for each edge, equal to the sum of the ll and rl fields? This would be equivalent to always placing the retention points at the far endpoints of p_l and p_r when considering vertex p.

Retention points are necessary because eliminating them can lead to arbitrarily large errors. Consider the infinite class of brush shapes $\mathcal{B}_1, \mathcal{B}_2, \ldots,$ where $\mathcal{B}_k$ is a hexagon whose vertices are

$$(-\tfrac{k}{2},-\tfrac{k}{2}),\quad (0,-\tfrac{k}{2}),\quad .49(0,-\tfrac{k}{2})+.51(\tfrac{k}{2},-\tfrac{1}{2}),$$
$$(\tfrac{k}{2},\tfrac{k}{2}),\quad (0,\tfrac{k}{2}),\quad \text{and}\quad .49(0,\tfrac{k}{2})+.51(-\tfrac{k}{2},\tfrac{1}{2}).$$

Figure 10 shows $\mathcal{B}_{10}$ and a pen polygon that would be computed from it if Algorithm 1 were modified to avoid retention points as suggested above.

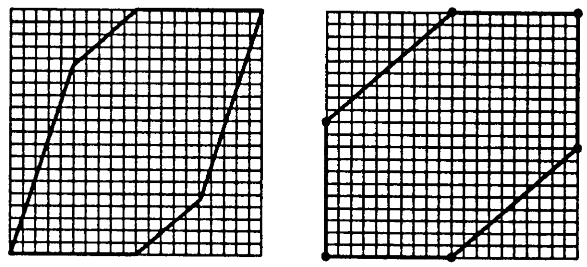

Fig. 10. A brush shape and a pen generated from it without using retention points. Each is superimposed on a half integer grid.

The first edge added in Step 4 would go between $(.5,-5)$ and $(5,-.5)$. Next the vertex at $(.5,-5)$ would be replaced by an edge of slope $\frac{1}{2}$ from $(0,-5)$ to $(1,-4.5)$. The vertex at $(1,-4.5)$ is then replaced by an edge of slope $\frac{2}{3}$ ending at $(1.5,-4)$. This process continues with each new vertex displaced by $(.5,.5)$ from the previous one until there is a line from $(0,-5)$ to $(5,-.5)$. Since this is followed in the data structure by an edge of length 0 in direction $(1,1)$, the vertex at $(5,-.5)$ is never removed from the polygon. The error between this polygon and $\mathcal{B}_{10}$ is about 1.67 and the general formula for the error in approximating $\mathcal{B}_k$ is $\lambda_1(k+1)/\sqrt{1+\lambda_2/k+\lambda_3/k^2}$ where $\lambda_1 \approx .156$ and $0<\lambda_2,\lambda_3<.6$.

We have seen that Theorem 4.1.12 can fail disastrously if Algorithm 1 is simplified too much, but does the theorem in any sense guarantee that the algorithm is optimal? Not quite, but the worst case bound of $\frac{3}{8}$ is nearly optimal. Consider the class of brushes $\mathcal{B}_1, \mathcal{B}_2, \ldots,$ where $\mathcal{B}_k$ is a line segment with endpoints $\pm z_k$, and $z_k = \big((2k+1)/4, (2k-1)/4\big)$. In the following theorem, we make use of this class of "difficult" brushes to show that any pen construction algorithm must have a worst case error bound of at least $\sqrt{2}/4$.

Theorem 4.1.13. *For any $\epsilon > 0$, there exists a convex brush $\mathcal{B}$, symmetrical about the origin, such that $E(\mathcal{P},\mathcal{B}) > \sqrt{2}/4-\epsilon$ for any convex pen $\mathcal{P}$.*

Proof. Let $\mathcal{B}_k$ be as defined above. Since $\lim_{k\to\infty} 1/\sqrt{8+2/k^2} = \sqrt{2}/4$, it suffices to show that for any convex pen $\mathcal{P}$, the error $E(\mathcal{P},\mathcal{B}_k) \geq 1/\sqrt{8+2/k^2}$. Furthermore, we can assume that $\mathcal{P}$ is symmetrical about the origin since by Lemma 4.1.1 $E(\mathcal{F}(\mathcal{P}),\mathcal{B}) \leq E(\mathcal{P},\mathcal{B})$ for any pen $\mathcal{P}$ and any symmetrical brush $\mathcal{B}$.

Now consider the supporting lines of $\mathcal{P}$ perpendicular to the vector $(1,1)$. They must be of the form $x+y=\pm n/2$ for some integer n. If $n \le 2k-1$ then $d(z_k,\mathcal{P}) \ge \sqrt{2}/4$; if $n \ge 2k+1$ then any point z on the supporting lines satisfies $d(z,\mathcal{B}_k) \ge \sqrt{2}/4$. If $n=2k$ then $\mathcal{P}$ contains a point of the form $\bigl((k+j)/2,(k-j)/2\bigr)$ and of these, $(k/2,k/2)$ is the closest to $\mathcal{B}_k$ and it is $1/\sqrt{8+2/k^2}$ away. ∎

4.2. Generating Asymmetrical Pens

The problem of generating a pen $\mathcal{P}$ that closely approximates an asymmetrical brush $\mathcal{B}$ appears to be significantly harder than the analogous problem for symmetrical brushes discussed in Section 4.1. We can still construct good practical algorithms analogous to Algorithm 1, but without the 180° rotational symmetry assumption. However, worst case bounds on the error $E(\mathcal{P},\mathcal{B})$ are not as good as that of Theorem 4.1.12. The algorithms discussed here have the advantage of good running time and small typical values for $E(\mathcal{P},\mathcal{B})$.

Our algorithm for finding a pen that approximates a symmetrical brush works by effectively taking intersections of good width strips of the form $|vx-uy| \le k/2$ for simple rational directions (u,v) and integers k. For asymmetrical brush shapes, we can simply generalize this to $a-k \le vx-uy \le a$. For any brush $\mathcal{B}$, it is always possible to find a real number a and an integer k so that the supporting lines of $\mathcal{B}$ in the direction (u,v) are within $1/\bigl(4\sqrt{u^2+v^2}\bigr)$ of those of the strip $a-k \le vx-uy \le a$.

In reality there are limitations on the positions of the new edges, so that it is not always possible to keep the errors less than $1/\bigl(4\sqrt{u^2+v^2}\bigr)$. This leads to an interesting problem: Given real numbers a, b, l, and m with $l,m \ge 0$, find α and β minimizing $\max\bigl(|\alpha-a|,|\beta-b|\bigr)$ subject to the constraints

$$0 \le \alpha \le l, \quad 0 \le \beta \le m, \quad \text{and} \quad \alpha+\beta \in \mathbb{Z}. \tag{4.2.1}$$

If there is more than one way to do this optimally then we want the one that minimizes $\min\bigl(|\alpha-a|,|\beta-b|\bigr)$. We shall call this the *pairwise rounding problem.*

Once we have decided what $\alpha+\beta$ should be, the problem is fairly easy. The pair (x,y) minimizing the ∞-norm distance $\max\bigl(|x-a|,|y-b|\bigr)$ with fixed $x+y$ is the one where $x-y=a-b$. In fact the ∞-norm distance is just

$$\tfrac{1}{2}\bigl(\bigl|(x+y)-(a+b)\bigr| + \bigl|(x-y)-(a-b)\bigr|\bigr)$$

so it increases linearly as we move away from the point of closest approach. The possibilities for $x-y$ form a closed interval and we need only select the value closest to $a-b$.

Choosing the best value for $\alpha+\beta$ is not quite so easy. Figure 11 shows what regions in (α,β) space have each possible optimum value of $\alpha+\beta$ in a typical example. The bold diagonal lines represent the possible choices for (α,β) and the rectangle shows the bounds of the region $0 \le a \le l$, $0 \le b \le m$. Thinner diagonal lines delimit regions of (α,β) space where particular values of $\alpha+\beta$ are optimal. These regions can have even more complicated shapes in special cases where l or m is small.

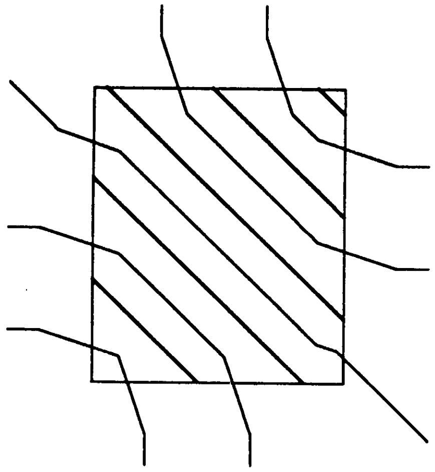
Fig. 11. Optimum choices for $\alpha + \beta$ when $l = 2.375$ and $m = 2.875$

In spite of the complexity it is not hard to find two candidates for $\alpha + \beta$, one of which must be optimal. Let a' be the point of the interval $[0, l]$ that is closest to a, and similarly let $b' \in [0, m]$ be as close to b as possible. Then the two candidates for $\alpha + \beta$ are $\lfloor a' + b' \rfloor$ and $\lceil a' + b' \rceil$. For any other pair (α, β) satisfying (4.2.1), there is a similar pair with one of the candidate sums and α strictly closer to a or β strictly closer to b. Thus, it suffices to just try the two candidate sums and see which one allows the ∞-norm to be minimized.

Let us incorporate our ideas into Algorithm 1 by extending the data structures to store both sides of the polygon being constructed. There are many ways to do this but the following method involves the least change to Algorithm 1: Each vertex node p will contain an additional pair of coordinates $z'(p) = \bigl(x'(p), y'(p)\bigr)$, and each edge e will contain two more "lengths" $ll'(e)$ and $rl'(e)$. Thus each vertex node represents two opposite vertices and each edge node represents two opposite edges. We shall still use $\mathcal{B}(u,v) = \bigl(\mathcal{B}_x(u,v), \mathcal{B}_y(u,v)\bigr)$ for the point where the brush boundary attains the direction (u, v).

Each edge e determines two directed lines: If p is a vertex node adjacent to e, then $\ell(e)$ is the line of direction $w(e)$ passing through $z(p)$, and $\ell'(e)$ is the line of direction $-w(e)$ passing through $z'(p)$. Given two such lines ℓ^- and ℓ^+, we can define a point $\bar{\mathcal{B}}(\ell^-, \ell^+, \bar{u}, \bar{v})$ analogous to the point $\bar{z}$ computed in Step 2 of Algorithm 1: Let $(u^\pm, v^\pm)$ be the direction of $\ell^\pm$; let $I_1(\mathcal{B}, \ell^\pm)$ and $I_2(\mathcal{B}, \ell^\pm)$ be the intersection points of the directed line $\ell^\pm$ with $\mathcal{B}$ ordered so that $I_2(\mathcal{B}, \ell^\pm) - I_1(\mathcal{B}, \ell^\pm)$ is a non-negative multiple of $(u^\pm, v^\pm)$; and let $L(\ell^\pm)$, $R(\ell^\pm)$, and $\bar{R}(\ell^\pm)$ be the halfplanes to the left and right of the directed line $\ell^\pm$ analogous to the halfplanes defined for edge nodes in Section 4.1. If $\mathcal{B}(u^-, v^-) \in \bar{R}(\ell^-) \cap \bar{R}(\ell^+)$ or $\mathcal{B}(u^+, v^+) \in \bar{R}(\ell^-) \cap \bar{R}(\ell^+)$ then $\bar{\mathcal{B}}(\ell^-, \ell^+, \bar{u}, \bar{v}) = \mathcal{B}(\bar{u}, \bar{v})$. If $\mathcal{B}(\bar{u}, \bar{v}) \in R(\ell^+) \cap L(\ell^-)$, then $\bar{\mathcal{B}}(\ell^-, \ell^+, \bar{u}, \bar{v}) = I_1(\mathcal{B}, \ell^+)$; if $\mathcal{B}(\bar{u}, \bar{v}) \in R(\ell^-) \cap L(\ell^+)$, then $\bar{\mathcal{B}}(\ell^-, \ell^+, \bar{u}, \bar{v}) = I_2(\mathcal{B}, \ell^-)$. Otherwise $\bar{\mathcal{B}}(\ell^-, \ell^+, \bar{u}, \bar{v}) = \mathcal{B}(\bar{u}, \bar{v})$.

Algorithm 2 (Asymmetrical brush to asymmetrical pen).

1) Initialize the data structure to $(e_1, v_1, e_2, v_2, e_3)$ where

$$\begin{aligned}
x(v_2)+x'(v_2)&=\mathcal{B}_x(0,1)+\mathcal{B}_x(0,-1), & x(v_1)&=x(v_2),\\
x(v_2)-x'(v_2)&=\lfloor \mathcal{B}_x(0,1)-\mathcal{B}_x(0,-1)+\tfrac12\rfloor, & x'(v_1)&=x'(v_2),\\
y(v_2)+y'(v_2)&=\mathcal{B}_y(-1,0)+\mathcal{B}_y(1,0), & y(v_1)&=y'(v_2),\\
y(v_2)-y'(v_2)&=\lfloor \mathcal{B}_y(-1,0)-\mathcal{B}_y(1,0)+\tfrac12\rfloor, & y'(v_1)&=y(v_2),\\
w(e_1)=-w(e_3)&=(1,0), & w(e_2)&=(0,1).
\end{aligned}$$

2) Use pairwise rounding to find $rl(e_1)\le x(v_1)-x'(v_1)$ near $x(v_1)-\mathcal{B}_x(1,0)$ and $rl'(e_1)\le x(v_1)-x'(v_1)$ near $\mathcal{B}_x(-1,0)-x'(v_1)$. Similarly find $rl(e_2)\le y(v_2)-y(v_1)$ near $y(v_2)-\mathcal{B}_y(0,1)$ and $rl'(e_2)\le y(v_2)-y(v_1)$ near $\mathcal{B}_y(0,-1)-y'(v_2)$. Now set the other "length" fields so that

$$\begin{gathered}
ll(e_2)+rl(e_2)=ll'(e_2)+rl'(e_2)=y(v_2)-y(v_1),\\
ll(e_3)=ll'(e_1),\qquad ll'(e_3)=ll(e_1).
\end{gathered}$$

Finally set $p\leftarrow v_1$ and go on to Step 3.

3) Let $\bar z=\bar{\mathcal{B}}\big(\ell(p_l),\ell(p_r),\bar u,\bar v\big)$ and $\bar z'=\bar{\mathcal{B}}\big(\ell'(p_l),\ell'(p_r),-\bar u,-\bar v\big)$, where $(\bar u,\bar v)=w(p_l)+w(p_r)$. Then use pairwise rounding to find

$$\delta\le\min\big(rl(p_l),ll(p_r)\big)\quad\text{and}\quad \delta'\le\min\big(rl'(p_l),ll'(p_r)\big)$$

near $\big\langle(-\bar v,\bar u),\,\bar z-z(p)\big\rangle$ and $\big\langle(\bar v,-\bar u),\,\bar z'-z'(p)\big\rangle$.

4) If $\delta>0$ or $\delta'>0$ then go on to Step 5. Otherwise advance p to the next vertex and go to Step 3, but halt if there are no more vertices to advance to.

5) Insert a new vertex q and a new edge q_l between p and p_r so that p_r becomes q_r; then set

$$\begin{aligned}
z(q)&\leftarrow z(p)+\delta\cdot w(q_r); & ll(q_r)&\leftarrow ll(q_r)-\delta;\\
z(p)&\leftarrow z(p)-\delta\cdot w(p_l); & rl(p_l)&\leftarrow rl(p_l)-\delta;\\
z'(q)&\leftarrow z'(p)-\delta'\cdot w(q_r); & ll'(q_r)&\leftarrow ll'(q_r)-\delta';\\
z'(p)&\leftarrow z'(p)+\delta'\cdot w(p_l); & rl'(p_l)&\leftarrow rl'(p_l)-\delta';\\
w(q_l)&\leftarrow(\bar u,\bar v).
\end{aligned}$$

6) If $rl(p_l)=rl'(p_l)=0$ then set $ll(q_l)\leftarrow ll'(q_l)\leftarrow 0$; if $ll(q_r)=ll'(q_r)=0$ then set $ll(q_l)\leftarrow\delta$ and $ll'(q_l)\leftarrow\delta'$; otherwise use pairwise rounding to find $ll(q_l)\le\delta$ and $ll'(q_l)\le\delta'$ near

$$\frac{\big\langle(\bar u,\bar v),\,\bar z-z(p)\big\rangle}{\bar u^2+\bar v^2}\quad\text{and}\quad\frac{\big\langle(-\bar u,-\bar v),\,\bar z'-z'(p)\big\rangle}{\bar u^2+\bar v^2}.$$

Finally set $rl(q_l)\leftarrow\delta-ll(q_l)$ and $rl'(q_l)\leftarrow\delta'-ll'(q_l)$ and go back to Step 3.

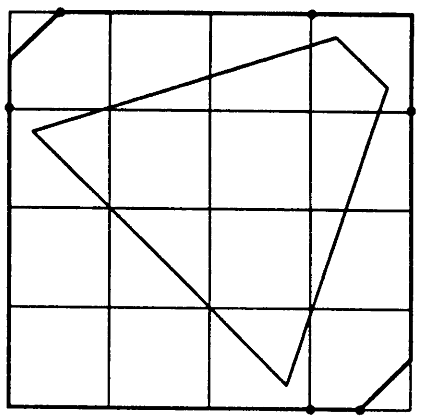
Fig. 12. A bad case for Algorithm 2

The above algorithm tends to perform well in practice, but its worst case error is quite large. Figure 12 shows a brush shape $\mathcal{B}$ that is approximated with an error greater than 1. The figure shows a unit grid and the partially constructed pen polygon in bold lines superimposed on the brush with retention points shown as black dots. In Step 1 the height and width of the initial square are rounded up from 3.52 to 4. Step 2 obtains $rl(e_1)$ and $rl'(e_1)$ by adjusting the ideal values of 1.23 and 3.25 to .99 and 3.01; $rl(e_2)$ and $rl'(e_2)$ are ideally .75 and 2.77 but are also adjusted to .99 and 3.01. This forces $\delta = \delta' = .5$ in the first execution of Step 3. Step 6 must then place the retention points for the new edges at opposite corners either as shown or with $ll(q_l) = ll'(q_l) = .5$. In any case, one of the retention points is about 1.1 units away from $\mathcal{B}$.

This example can be generalized to larger $\mathcal{B}$ in such a way that the error approaches 5/4. Thus the worst case error of Algorithm 2 is more than three times that of Algorithm 1. In spite of this, Algorithm 2 usually performs well in practice and we can modify it to obtain a worst case error bound of only twice that for the symmetrical case. The modified algorithm will be a hybrid that combines the practical advantages of Algorithm 2 with the good worst case performance of the following algorithm.

Let us replace the pairwise rounding operations in Steps 2, 3, and 6 of Algorithm 2 with independent rounding operations and call the result Algorithm 2b. That is, given a, b, l and m, let

$$\alpha = \lfloor \max(0, \min(a, l)) + \tfrac{1}{2} \rfloor \quad \text{and} \quad \beta = \lfloor \max(0, \min(b, m)) + \tfrac{1}{2} \rfloor .$$

This has the effect of doubling the expected error in placement of new edges and retention points when $0 \ll a \ll l$ and $0 \ll b \ll m$, but it also guarantees that the ll and rl entries will always be integers so that the limits l and m will always be integers.

The function $\bar{\mathcal{B}}$ has been carefully designed so that Step 3 of Algorithm 2b behaves like Step 2 of Algorithm 1. Step 2 of Algorithm 1 is almost exactly equivalent to setting $\bar{z} \leftarrow \bar{\mathcal{B}}(\ell(p_l), \ell(p_r), \bar{u}, \bar{v})$ and then setting δ as it as stated. The only difference is that Algorithm 1 sets $\delta \leftarrow 0$ when $\mathcal{B}(u(p_l), v(p_l))$ or $\mathcal{B}(u(p_r), v(p_r))$ belongs to $\bar{R}(p_l) \cap \bar{R}(p_r)$, while $r_d(\langle(-\bar{v}, \bar{u}), \bar{z} - z(p)\rangle)$ might be negative.

It is not too hard to see that Algorithm 2b is equivalent to the following procedure: Take the given brush $\mathcal{B}$ and shift it by a vector Δ so that when Algorithm 2 is applied to $\mathcal{B}+\Delta$, the initial rectangle will be centered on the origin. Now independently apply Algorithm 1 to the scaled version $\mathcal{B}' = \frac{1}{2}(\mathcal{B}+\Delta)$ and its reflection about the origin $-\mathcal{B}'$. Construct the final pen polygon by appending the negatives of the vertices found for $-\mathcal{B}'$ to those for $\mathcal{B}'$, doubling all of them and subtracting Δ. (The brush $\mathcal{B}'$ is not symmetrical about the origin, but Algorithm 1 effectively makes it symmetrical by just looking at the right side and assuming symmetry.) Theorem 4.1.12 allows us to conclude the following.

Theorem 4.2.1. *If $\mathcal{P}$ is a pen computed by Algorithm 2b to approximate a convex brush $\mathcal{B}$, then $E(\mathcal{P},\mathcal{B}) < \frac{3}{4}$.* ∎

While this shows that we can avoid disasters such as those of Figures 47 and 12, the bound does not seem to be very good. Algorithm 2b really can produce errors approaching $\frac{3}{4}$, but such large errors may not be necessary; there is no known lower bound on worst case error greater than the $\sqrt{2}/4$ obtained from Theorem 4.1.13.

In spite of the relatively poor error bound for Algorithm 2b, the results of Algorithm 2 can sometimes be improved substantially by replacing some of the pairwise rounding steps with independent rounding. When δ or δ' must be set far from the ideal values in Step 3, we can backtrack and force some of the previous δ or ll values to be integers. Typically this leads to quite reasonable errors, not errors approaching $\frac{3}{4}$. In addition, the trouble is usually detected quickly so that the amount of backtracking is not very large.

The most important new data structure required by the hybrid algorithm is a special record b that holds the information required to restore the main data structure to its previous state when it is necessary to back up. Ordinarily, b contains a vertex v_b and two adjacent edges e_{bl} and e_{br} together with pointer to two vertices v_{bl} and v_{br} in the main data structure. To restore the previous state, we throw way all edges and vertices between v_{bl} and v_{br} and replace them with e_{bl}, v_b, and e_{br} in that order.

The backup record is maintained so that vertices v_{bl} and v_{br} are as close together as possible while still surrounding the current vertex p and having all ll, ll', rl, and rl' fields of e_{bl} and e_{br} guaranteed to be integers. This means that the edges in directions $w(e_{bl})$ and $w(e_{br})$ and all the edges for directions that are ancestors of $w(e_{bl})$ or $w(e_{br})$ in the Stern-Peirce wreath were placed using independent rounding instead of pairwise rounding. If no lengths are guaranteed to be integers for any directions that are ancestors of $w(p_l)$ or $w(p_r)$, then b holds a flag *restart* to indicate that we must start over using independent rounding to place retention points on the initial rectangle. Another special flag *null* identifies the case when no backup is possible because all edges for directions $w(e_{bl})$ or $w(e_{br})$ and their ancestors are guaranteed to have integer lengths.

We also need a special flag *force_integers* to indicate whether pairwise rounding operations are currently being replaced by independent rounding. Here is how to change Algorithm 2 into the hybrid version, Algorithm 2c: set *force_integers* to *false* before returning to Step 3 from Steps 4 or 6; initialize *force_integers* to *false* and b to *restart* in Step 1; and at the end of Step 3 if $b \neq$ *null* then check for excessive error and backtrack if necessary. The backtracking operation consists of restoring the main data structures from the information in b, resetting b to *null* and *force_integers* to *true*, and repeating Step 3. When b is *null* and *force_integers* is false on entry to Step 5, then p, p_l, and p_r are saved in b. When p is advanced to v_{br} in Step 4, then b is reset to *null*. To restore the main data structures when $b =$ *restart*, we repeat Steps 1 and 2 with *force_integers* set to *true*.

The definition of "excessive error" in Step 3 is that

$$\max\left(\left|\langle(-\bar{v},\bar{u}),\,\bar{z}-z(p)\rangle-\delta\right|,\,\left|\langle(\bar{v},-\bar{u}),\,\bar{z}'-z'(p)\rangle-\delta'\right|\right)$$

is greater than $\epsilon\sqrt{\bar{u}^2+\bar{v}^2}$ where ϵ is a parameter that controls the amount of backtracking. It should be set large enough so that integer lengths are likely to lead to an improvement, but not large enough to allow unnecessarily large error. Increasing ϵ tends to reduce the amount of backtracking and thus speed up the algorithm when this is a factor. The best value for ϵ is probably slightly more than $\frac{3}{8}$ so that the results are equivalent to those of Algorithm 1 for symmetrical brush shapes. Algorithm 2c has never been known to give approximation error greater than ϵ when $\epsilon \geq \frac{3}{8}$.

Another possible refinement to Algorithm 2 is that we might want to use $ll(p_l) + rl(p_l)$ instead of $rl(p_l)$ and $ll'(p)_l + rl'(p_l)$ instead of $rl'(p_l)$ in Step 3 (except when p_l is the first edge in the data structure). That is, we can discard the retention point on the edge we have just passed if the previous part of the polygon retains more of that edge than necessary. Relaxing the restrictions on δ and δ' in Step 3 in this way can only reduce the error. The author has not been using this refinement so far because it complicates the analysis by introducing a dependence on scan order and it does not seem to reduce the worst case error.

4.3. Analysis of Pen Generating Algorithms

Let us look at the time and space requirements of Algorithm 1 and the various versions of Algorithm 2. The input is a brush shape given in terms of a spline description for the boundary; or if some limited class of shapes is being used, it is given in terms of a set of parameters. For instance, if our brush shapes are ellipses centered on the origin, we might be given the major and minor axes and the angle of rotation. Since the exact time and space requirements are not easily described in terms of natural properties of this input, we shall relate the requirements to properties of the pen polygon produced. We can then get reasonable time and space bounds by relating the pen polygon to the diameter of the brush that it must approximate.

Let $\mathcal{P}$ be a pen polygon computed by Algorithm 1 to approximate a symmetrical brush $\mathcal{B}$, and let a be the number of non-null edges in $\mathcal{P}$. (Recall that the data structures may contain edges of length zero, and we referred to them in Section 4.1 as "null edges.") Throughout this analysis we shall assume that the widths of $\mathcal{B}$ in directions $(1,0)$ and $(0,1)$ are $\geq \frac{1}{2}$ so that $\mathcal{P}$ is non-trivial.

Even though it is possible to eliminate null edges, we still need to talk about the number of them in the data structures of Algorithm 1, and it is desirable to relate this to the polygon $\mathcal{P}$. To do this, we use the Stern-Peirce wreath and consider the sequence w_0, w_1, ..., w_{a-1} of directions of edges in $\mathcal{P}$ where $w_j = (u_j, v_j)$. Because of the cyclic nature of the pen polygon, we shall consider the subscripts modulo a so that $w_j = w_{j+a}$ for any integer j. Given any two adjacent edge directions w_j and w_{j+1}, we can construct a unique sequence of directions for the null edges that must lie between them in the data structures after the completion of Algorithm 1.

We can reduce the general case to the case where v_j and v_{j+1} are both non-negative: If v_j and v_{j+1} are both ≤ 0, then take the negatives of the null edges between $-w_j$ and $-w_{j+1}$; if $v_j < 0$ and $v_{j+1} > 0$, then take the negatives of the null edges between $-w_j$ and $(-1,0)$ and append those between $(1,0)$ and w_{j+1} with exactly one copy of $(1,0)$ in between; if $v_j > 0$ and $v_{j+1} < 0$, then follow the previous instructions but negate the results.

We shall need to compare edge directions on the basis of "simplicity." In general for reduced rational pairs (u, v) and (u', v'), we say that (u, v) is *simpler* than (u', v') if and only if $|u| \leq |u|'$, $|v| \leq |v|'$, and $(u, v) \neq (u', v')$.

A basic property of the sequence of edge directions in the data structures of Algorithm 1 is that whenever a new edge replaces an existing vertex p, the directions of the edges p_l and p_r surrounding p are simpler than the direction of the new edge. Thus whenever two edges are adjacent in the data structure, either one is simpler than the other or they are both part of the initial rectangle. We can now prove the following lemma.

Lemma 4.3.1. *For any sequence of edges e_1, e_2, ..., e_k in the data structures of Algorithm 1 such that $ll(e_j) = rl(e_j) = 0$ for $1 \leq j \leq k$, there exists m such that $w(e_{j+1})$ is simpler than $w(e_j)$ when $j < m$ and $w(e_j)$ is simpler than $w(e_{j+1})$ when $j \geq m$.*

Proof. First strengthen the hypothesis by appending the surrounding non-null edges e_0 and e_{k+1} to the sequence, except do not append e_0 if it is the first edge in the data structure and do not append e_{k+1} if it is the last.

After Step 1 of Algorithm 1, the strengthened hypothesis is trivially satisfied because there are no null edges. Since no edges are ever deleted from the data structure and newly added edges are always non-null and cannot be adjacent to edges already of null, the only time that the hypothesis could possibly become false is when the length of an existing edge becomes 0. Furthermore this can happen only when δ is subtracted from $ll(q_r)$ or $rl(p_l)$ in Step 4. Either way there is a newly added edge q_l, and the null edge is its neighbor and has a simpler direction.

If the null edge had no null edge neighbors then the hypothesis clearly remains true. Otherwise the null edge was part of a sequence that already satisfied the hypothesis and we have merely appended q_l to one end of that sequence. Since the null edge is simpler than q_l, the hypothesis is still true. ∎

Lemma 4.3.1 gives important information about the directions of the sequence of null edges between two adjacent non-null edges e_j and e_{j+1}. Combining this with the fact that consecutive directions (u,v) and (u',v') in this sequence satisfy (4.1.1), we can deduce exactly what that sequence must be. This argument is based on well known facts due to Stern [28], but for clarity we restate them here as a lemma.

Lemma 4.3.2. *If (u,v) is a reduced rational pair then there is a unique reduced rational pair (u',v') that satisfies (4.1.1) and either is simpler than (u,v) or is one of the four basic directions $(0,\pm1)$ or $(\pm1,0)$.*

Proof. If (u,v) is one of the four basic directions mentioned in the lemma, then (u',v') must also be one of these. It is easy to check that $(u',v')=(-v,u)$ is the unique such direction that satisfies (4.1.1).

Otherwise since $\gcd(u,v)=1$ there are integers u' and v' such that $uv'-vu'=1$, and it follows immediately from this equation that $\gcd(u',v')=1$. If $uv''-vu''=1$ for some other reduced rational pair (u'',v'') then $u(v'-v'')=v(u'-u'')$ so that $v \mid v'-v''$ and $u \mid u'-u''$. In fact $uv''-vu''=1$ whenever $(u'',v'')=(u',v')+k(u,v)$ for any integer k.

The rest of the proof depends on the fact that we have already dealt with all cases where $u=0$ or $v=0$. If $uv>0$ then choose k so that $0<v''\le v$ or $v\le v''<0$; if $uv<0$ then force $0\le v''<v$ or $v<v''\le 0$. Either way vu'' is between 0 and uv inclusive, so u'' is between 0 and u. Thus (u'',v'') satisfies (4.1.1) and is simpler than (u,v). For any other value of k, either $uu''+vv''<0$ or (u'',v'') is not simpler than (u,v). ∎

In terms of the Stern-Peirce wreath, the pair (u',v') can be located as follows: if (u,v) is one of the four basic directions, then take the next such direction; if (u,v) is the sum of two basic directions, then take the one at its right; otherwise (u,v) belongs to an ordinary Stern-Peirce tree. If (u,v) is a left son then the simpler adjacent pair is the father of (u,v), otherwise it is the father's simpler adjacent pair.

Lemma 4.3.2 tells us how to find simpler adjacent pairs going counterclockwise around the wreath. Replacing 1 by -1 in (4.1.1) makes it applicable to adjacent pairs going around the wreath the other way. Lemma 4.3.2 can easily be adapted to use this modified version of (4.1.1) and find unique simpler clockwise adjacent pairs.

To construct the null edge directions between two consecutive non-null edge directions w_j and w_{j+1}, begin by constructing a sequence of successively simpler adjacent directions w_{j1}, w_{j2}, $\ldots$, starting from $w_j=w_{j0}$. Construct a similar clockwise adjacent sequence $w_{j+1,-1}$, $w_{j+1,-2}$, $\ldots$, starting from $w_{j+1}=w_{j+1,0}$. If none of the four basic directions lie between w_j and w_{j+1} then these sequences will meet at the lowest common ancestor of w_j and w_{j+1}, otherwise they will meet at the basic direction in between. Thus there will be nonnegative indices m and n such that the null edges have directions $w_{j,1}$, $w_{j,2}$, $\ldots$, $w_{j,m}=w_{j+1,-n}$, $w_{j+1,1-n}$, $\ldots$, $w_{j+1,1}$.

Let us construct the sequence of null edge directions that would lie between consecutive non-null edges of directions $(27,19)$ and $(11,25)$ shown in Figure 13. Starting from $(27,19)$, we obtain the sequence of successively simpler adjacent

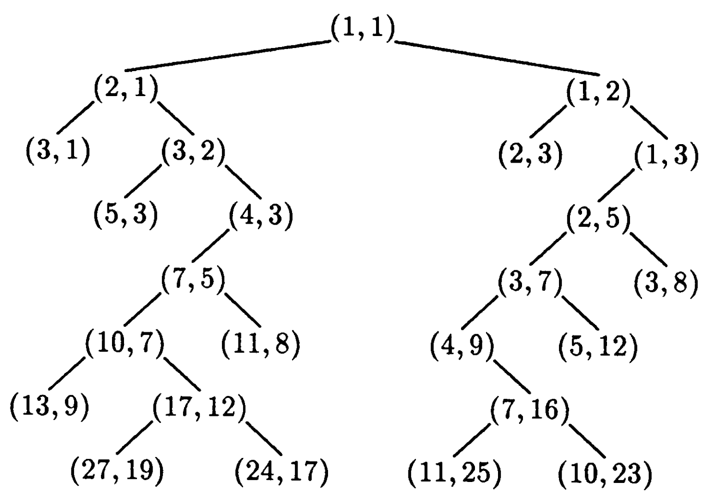

Fig. 13. Part of a Stern-Peirce tree.

directions $(17,12), (7,5), (4,3), (1,1), (0,1), \ldots$. Going backwards from $(11,25)$, we obtain $(4,9), (1,2), (1,1), (1,0), \ldots$ Thus the null edge directions are $(17,12)$, $(7,5), (4,3), (1,1), (1,2), (4,9)$.

Going back to the pen polygon $\mathcal{P}$ computed by Algorithm 1 from the symmetrical brush $\mathcal{B}$, recall that a is the number of non-null edges in $\mathcal{P}$. Let b be the total number of null edges whose directions are obtained by applying the above construction to each pair of adjacent edges in $\mathcal{P}$. Since the data structures actually contain only the right half of $\mathcal{P}$, the actual number of null edges stored will be about $b/2$. The total number of edges in the data structures on completion is $(a+b)/2+1$ counting two horizontal edges; the total number of vertices is just $(a+b)/2$. Since nothing is ever deleted from the data structures, this is the total space requirement.

The time requirement of Algorithm 1 is equally easy to analyze. Step 1 is executed once; Steps 4 and 5 are executed $(a+b)/2-2$ times each, once for each edge not inserted in Step 1; and Steps 2 and 3 are also executed once for each vertex of $\mathcal{P}$ so that the total is $a+b-2$ times.

Now consider modifying Step 4 so that it never allows any null edges into the data structure. This means that we delete q_r if $ll(q_r) = rl(q_r) = 0$, and we delete p_l if $ll(p_l) = rl(p_l) = 0$. When q_r is deleted, it is necessary to mark the vertex thus formed to indicate that the pointer p should skip over it. When the modified algorithm terminates, the data structures contain $a/2$ edges and $a/2+1$ vertices or vice versa. The maximum storage requirement can be larger than this in unusual circumstances, but the difference is never more than a factor of two.

The only change in the run time of Algorithm 1 due to null edge elimination is that Steps 2 and 3 are executed $(a+b+c)/2-2$ times instead of $a+b-2$ where $c \le a$ is the number of vertices of $\mathcal{P}$ that do not get marked and skipped over as described above. Thus the time saving due to null edge elimination is never very large, but the space saving may be significant if $b \gg a$.

These results can be extended to the asymmetric case if we let $\mathcal{P}'$ be a pen polygon computed by Algorithm 2 or 2b from a not necessarily symmetrical brush $\mathcal{B}'$. Its sequence of edge directions $w'_0, w'_1, \ldots, w'_{a'-1}$ can be filled out to another sequence $w_0, w_1, \ldots, w_{a-1}$ so that $w_{i+a/2} = -w_i$ for $0 \le i < a/2$. That is if for some i there is no j such that $w'_j = -w'_i$, then insert $-w'_i$ and repeat. We then use the sequence w_j to find null edge directions exactly as above and let b be the number of such directions obtained. Now $a \le 2a'$ and we still get $(a+b)/2+1$ edges and $(a+b)/2$ vertices in the data structures. Steps 1 and 2 are executed once; Steps 3 and 4 are executed $a+b-2$ times; and Steps 5 and 6 are executed $(a+b)/2-2$ times. We can eliminate all null edges not required by the restriction that the data structure contains pairs of opposite edges, and the effects on time and space requirements can still be expressed as above in terms of a quantity $c \le a$.

We now need upper bounds on a and b in terms of the size of $\mathcal{B}$ or $\mathcal{B}'$. We shall use the fact that the perimeter P of $\mathcal{P}$ or $\mathcal{P}'$ satisfies

$$P \ge \tfrac{1}{2} \sum_{0 \le j < a} \sqrt{u_j^2 + v_j^2} \tag{4.3.1}$$

where $(u_j, v_j) = w_j$ are the non-null edge directions. This is because $\ell + \ell'$ must be an integer multiple of $\sqrt{u_j^2 + v_j^2}$ when ℓ and ℓ' are the lengths of the edges with directions w_j and $-w_j$. The perimeter of $\mathcal{B}$ or $\mathcal{B}'$ can differ from P by at most a small additive constant.

We shall say the $\{\, w_j \mid 0 \le j < a \,\}$ is a *circular set of directions* whenever

$$\begin{aligned}\{\, (u,v) \in \mathbb{Z}^2 \mid \gcd(u,v) = 1 \text{ and } u^2+v^2 < r \,\} &\subseteq \{\, w_j \mid 0 \le j < a \,\}\\ &\subseteq \{\, (u,v) \in \mathbb{Z}^2 \mid \gcd(u,v) = 1 \text{ and } u^2+v^2 \le r \,\}.\end{aligned}$$

for some radius r. If we restrict the direction set to be circular, the sum in (4.3.1) depends only on the cardinality a, and for any given cardinality this sum is minimized when the direction set is circular. Thus we can define s_k to be half of the value of this sum for any particular cardinality k. It follows that $s_a \le P$ for any set of directions satisfying (4.3.1). Furthermore, this bound is the best possible since we can actually construct a pen polygon with k sides whose perimeter is s_k for any k.

In order to estimate s_k, we need some information about the distribution of pairs of relatively prime integers in $\mathbb{Z}^2$. Let I and J be intervals, and let $C(I,J)$ be the number of $(u,v) \in \mathbb{Z}^2$ such that $\gcd(u,v) = 1$, $u \in I$ and $v \in J$. Extending exercise 4.5.2–10 of [14] as Knuth explains on page 595 of that source shows that

$$C\big([a, a+c), [b, b+c)\big) = \frac{6}{\pi^2}c^2 + O(c \log c) \tag{4.3.2}$$

where the constant in the O is independent of a and b. (See also Mertens [18].)

If $f_\alpha(x) = \alpha\lfloor x/\alpha + \frac{1}{2}\rfloor$ is the function that rounds to the nearest multiple of α then $\sqrt{u^2+v^2} = \sqrt{f_\alpha(u)^2 + f_\alpha(v)^2} + O(\alpha)$ and there are $O(\alpha r)$ pairs $(u,v) \in \mathbb{Z}^2$ such that $f_\alpha(u)^2 + f_\alpha(v)^2 \le r^2 < u^2+v^2$. Thus

$$\frac{1}{r^3} \sum_{\substack{\gcd(u,v)=1\\ u^2+v^2 \le r^2}} \sqrt{u^2+v^2} = \left(\frac{1}{r^3} \sum_{\substack{\gcd(u,v)=1\\ f_{r/k}(u)^2 + f_{r/k}(v)^2 \le r^2}} \sqrt{f_{r/k}(u)^2 + f_{r/k}(v)^2}\right) + O(1/k)$$

$$= \left(\frac{1}{r^3}\sum_{i^2+j^2\le k^2}\frac{r}{k}\sqrt{i^2+j^2}\,C\big([a_i,a_i+r/k),\,[a_j,a_j+r/k)\big)\right)+O(1/k), \qquad (4.3.3)$$

where $a_i = (i-\frac{1}{2})r/k$ and the constant in the O is independent of r. If the above quantity is p_r then (4.3.2) shows that

$$p_r = \left(\frac{6}{\pi^2 k^3}\sum_{i^2+j^2\le k^2}\sqrt{i^2+j^2}\right)\big(1+O((k/r)\log(r/k))\big)+O(1/k).$$

A simple argument similar to that used to derive (4.3.3) shows that

$$\frac{2\pi r^3}{3} = \int_0^{2\pi}\int_0^r R^2\,dR\,d\theta = \left(\frac{r^2}{k^2}\sum_{i^2+j^2\le k^2}\frac{r}{k}\sqrt{i^2+j^2}\right)+O(r^3/k), \qquad (4.3.4)$$

hence

$$\frac{1}{k^3}\sum_{i^2+j^2\le k^2}\sqrt{i^2+j^2} = \frac{2\pi}{3}+O(1/k)$$

and $p_r = 4/\pi + O\big(1/k + (k/r)\log(r/k)\big)$. Setting $k = (r/\log r)^{1/2}$ yields $p_r = 4/\pi + O(r^{-1/2}\log^{1/2} r)$.

A similar but somewhat simpler argument shows that

$$q_r = \frac{1}{r^2}\sum_{\substack{\gcd(u,v)=1\\ u^2+v^2\le r}} 1 = \frac{6}{\pi}+O(r^{-1/2}\log^{1/2} r).$$

Since $s_a = \frac{1}{2}r^3 p_r$ when $a = r^2 q_r$, the inequality $s_a \le P$ becomes $\frac{1}{2}r^3 p_r \le P$ or $r^2 q_r \le q_r(2P/p_r)^{2/3}$, hence

$$\Theta(r^2) = a \le q_r(2P/p_r)^{2/3} = \frac{6}{\sqrt[3]{4\pi}}P^{2/3} + O(P^{2/3}r^{-1/2}\log^{1/2} r)$$

and $r = \Omega(P^{1/3})$ when $a > 6P^{2/3}/\sqrt[3]{4\pi}$. Thus

$$a \le \frac{6}{\sqrt[3]{4\pi}}P^{2/3} + O(P^{1/2}\log^{1/2} P). \qquad (4.3.5)$$

A more careful analysis reduces the error term in (4.3.3) and (4.3.4) to $O(1/k^2)$, thus giving an error bound of $O(P^{4/9}\log^{2/3} P)$ in (4.3.5). Actually $-.1 < k - 6s_k^{2/3}/\sqrt[3]{4\pi} < 3.44$ for $s_k < 600\,000$, so the error term is very small in practice.

It would be nice to have a bound on b similar to (4.3.5), but unfortunately b can be $\Omega(P)$ where P is the perimeter. Consider the class of brush shapes $\mathcal{B}_n$ where $\mathcal{B}_n$ is the parallelogram with vertices $(-n,0)$, $(-n,-\frac{1}{2})$, $(n,0)$, and $(n,\frac{1}{2})$. Of course the pens generated will be identical parallelograms. We will have $a = 4$ for all n, but $b = 8n$ and $P = 4n + 1 + O(1/n)$. The $8n$ edges of length 0 have directions $(\pm 1, 0)$ and $\pm(k,1)$, for $4n < k < 1$.

When Algorithm 1 processes $\mathcal{B}_n$, it first sets up the data structures corresponding to a $2n \times 1$ rectangle. Successive values of $(\bar{u}, \bar{v})$ in Step 2 are $(1,1)$, $(2,1)$, ..., $(4n,1)$. Each time $ll(p_r) = \delta = \frac{1}{2}$ so that the new edge effectively replaces the edge just added. This is like having one edge and moving it so that one endpoint stays fixed at $(n,0)$ and the other one moves to the left in steps of $\frac{1}{2}$.

Suppose that at the beginning of Step 4 of Algorithm 1, $ll(p_r) = \delta$ and $rl(p_r) = 0$. Instead of immediately inserting an edge of direction $(\bar{u}, \bar{v})$, we want to place the new edge in its "final position"; i.e., we want to skip all immediately following iterations of Steps 2–5 in which $\delta = ll(p_r)$.

Let $A = z(p) + \delta w(p_r)$; let ℓ be a $w(p_l)$-directed line through $A - \frac{1}{4}w(p_r)$; let $\bar{A} = A + \frac{1}{4}w(p_l)$; and let $B_k = z(p) - kll(p_r) \cdot w(p_l)$. Successive iterations of Steps 2–5 would add an edge B_1A and then replace it with B_2A, B_3A, etc. Now find a supporting line of $\mathcal{B} \cap R(p_l)$ that passes through A and a supporting line of $\bar{\mathcal{B}} = \mathcal{B} \cap \bar{L}(p_l) \cap \bar{R}(\ell)$ that passes through $\bar{A}$. The new edge goes from B_k to A, where k is as large as possible subject to the constraints that the line $B_{k-1}A$ must not intersect $\mathcal{B} \cap R(p_l)$ and the line through $\bar{A}$ parallel to B_kA must not intersect $\bar{\mathcal{B}}$. Figure 14 shows this situation with $w(p_l) = (1,0)$, $w(p_r) = (0,1)$, and $ll(p_r) = 1$. The dots at intervals of $\frac{1}{2}$ in the figure show where new edges are allowed to cut the existing horizontal and vertical edges.

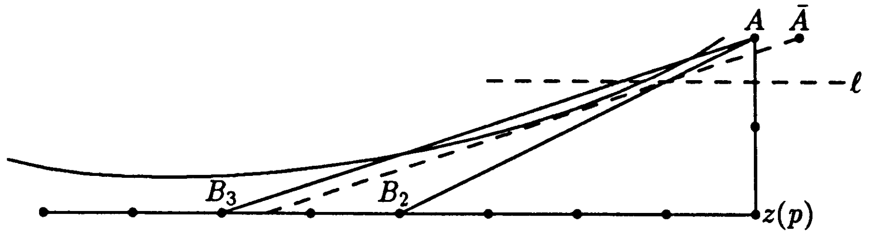

Fig. 14. How to avoid unnecessary iterations of algorithm 1 by finding a supporting line.

Let ℓ_k be the directed line through A and B_k with direction $w_k = kw(p_l) + w(p_r)$. Equation (4.1.1) shows that $\langle(\bar{v}, -\bar{u}), \bar{A} - A\rangle = \frac{1}{2}$ whenever $(\bar{u}, \bar{v}) = w_k$ and an edge parallel to ℓ_k is about to be added. Thus Lemma 4.1.3 shows that the $B_{k-1}A$ edge is replaced by an edge from B_k to A if and only if the line $\bar{\ell}_k$ through $\bar{A}$ with direction $A - B_k$ does not cut $\mathcal{B} \cap \bar{L}(p_l) \cap \bar{L}(\ell_{k-1})$, and $\mathcal{B}$ does not intersect $\bar{R}(p_l) \cap \bar{R}(\ell_{k-1})$. Since $\bar{\ell}_k \cap \bar{L}(\ell_{k-1}) = \bar{\ell}_k \cap \bar{R}(\ell)$, the intersection $\bar{\ell}_k \cap \mathcal{B} \cap \bar{L}(p_l) \cap \bar{L}(\ell_{k-1})$ is nonempty if and only if $\bar{\ell}_k \cap \bar{\mathcal{B}}$ is. Letting $\ell(p_l)$ be the line $\bar{L}(p_l) \cap \bar{R}(p_l)$, we have $\ell(p_l) \cap \bar{R}(\ell_{k-1}) \subset \ell(p_l) \cap \bar{R}(\bar{\ell}_k)$. Thus $\ell(p_l) \cap \bar{R}(\ell_{k-1}) \cap \mathcal{B}$ is empty when $\bar{R}(\bar{\ell}_k) \cap \mathcal{B}$ is. When $\bar{R}(\bar{\ell}_k)$ is empty, $\mathcal{B} \cap \bar{R}(p_l) \cap \bar{R}(\ell_{k-1})$ is empty if and only if $\mathcal{B} \cap \ell_{k-1} \cap \bar{R}(p_l)$ is. This shows that the final value of k for which an edge B_kA is added is the maximum k such that $\bar{\ell}_k \cap \bar{\mathcal{B}}$ and $\bar{R}(\ell_{k-1}) \cap (\bar{R}(p_l) \cap \mathcal{B})$ are both empty.

The above strategy is essentially a scheme for finding the maximum value of k such that

$$\langle(-\bar{v}, \bar{u}),\ \bar{\mathcal{B}}(\ell(p_l), \ell_k, \bar{u}, \bar{v}) - z_k\rangle > ll(p_r) - \tfrac{1}{4},$$

where $z_k = \ell(p_l) \cap \ell_k$ and $\bar{\mathcal{B}}$ is the function defined in Section 4.2. To extend this to Algorithm 2, let $\ell_k = \ell(p_r)$ and $\ell'_k = \ell'(p_r)$ after k iterations of Steps 3–6. We must find the maximum k such that

$$c_1 \cdot \langle(-\bar{v},\bar{u}), \mathcal{B}_1 - z_k\rangle + c_2 \cdot \langle(\bar{v},-\bar{u}), \mathcal{B}_2 - z'_k\rangle \geq c_3$$

where $\mathcal{B}_1 = \bar{\mathcal{B}}(\ell(p_l),\ell_k,\bar{u},\bar{v})$, $\mathcal{B}_2 = \bar{\mathcal{B}}(\ell'(p_l),\ell'_k,-\bar{u},-\bar{v})$, $z = \ell(p_l) \cap \ell_k$, $z' = \ell'(p_l) \cap \ell'_k$, and c_1, c_2, and c_3 are arbitrary constants. This would allow us to test against the thin lines shown in Figure 11.

The complete construction is rather complicated, but when $\bar{\mathcal{B}}(\ell(p_l),\ell_k,\bar{u},\bar{v}) = \mathcal{B}(\bar{u},\bar{v})$ and $\bar{\mathcal{B}}(\ell'(p_l),\ell'_k,-\bar{u},-\bar{v}) = \mathcal{B}(-\bar{u},-\bar{v})$, we apply a construction like that of Figure 14 to the curve

$$C = c_1\mathcal{B}(\sin\theta,\cos\theta) - c_2\mathcal{B}(-\sin\theta,-\cos\theta)$$

for some appropriate range of θ. If $C(u,v)$ denotes the point where C achieves the direction (u,v), then $C(u,v) = c_1\mathcal{B}(u,v) - c_2\mathcal{B}(-u,-v)$.

Actually, the above schemes are somewhat impractical because it is usually rather difficult to find supporting lines of a shape described by a spline description and it is often difficult to find convolutions. Perhaps a more realistic approach is to start some kind of binary search procedure if we repeatedly find $\delta = ll(p_r) + rl(p_r)$ in Algorithm 1 or if we repeatedly find $\delta = ll(p_r) + rl(p_r)$ and $\delta' = ll'(p_r) + rl'(p_r)$ in Algorithm 2. That is we can sometimes let $(\bar{u},\bar{v}) = kw(p_l) + w(p_r)$ where $k > 1$. If $\delta < ll(p_r) + rl(p_r)$ or $\delta' < ll'(p_r) + rl'(p_r)$ then we backtrack and reduce k.

Let us see how the above modification will allow us to replace the parameter b in the time and space bounds with a new parameter b' that is $O(P^{2/3})$ in the perimeter P. Recall that b was intended to describe the number of null edges in the data structures, but it also determines the number of iterations of the main loops of Algorithms 1 and 2. Consider the sequence of null edge directions that Lemmas 4.3.1 and 4.3.2 allow us to construct between the directions w_j and w_{j+1} of two edges that are adjacent in the computed pen polygon. Any subsequence w_{jk}, $w_{j,k+1}$, $\ldots$, w_{jl} all of which are left sons in their Stern-Peirce tree can now be thought of as a unit. A modification similar to that described above also allows the consecutive right sons to be thought of as a unit. Let such sequences of left sons or right sons be represented by their simplest element and let b' be the number of remaining null edges. Thus the remaining null edge directions between $(27,19)$ and $(11,25)$ from Figure 13 are $(17,12)$, $(4,3)$, $(1,1)$, and $(4,9)$.

Now consider the path leading to some edge direction w_j from the root of its Stern-Peirce tree. Let ω_0 be the root and let ω_1, ω_2, $\ldots$, ω_k be the left and right extrema along this path. That is we include all left sons whose successor is a right son and vice versa. If $w_j = (27,19)$ then $k = 3$, $\omega_0 = (1,1)$, $\omega_1 = (2,1)$, $\omega_2 = (4,3)$, and $\omega_3 = (10,7)$. Let $A(w_j)$ be the set of ω_i for even i and $B(w_j)$ be the remaining ω_i. The set of non-vertical, non-horizontal null edge directions between two consecutive directions w_j and w_{j+1} will be a subset of $A(w_j) \cup A(w_{j+1})$, $A(w_j) \cup B(w_{j+1})$, $B(w_j) \cup A(w_{j+1})$, or $B(w_j) \cup B(w_{j+1})$. By showing that

$$\sum_{(u,v)\in S} \sqrt{u^2+v^2} < \sqrt{u_j^2+v_j^2} \quad \text{for } S = A(w_j) \text{ and } S = B(w_j) \qquad (4.3.6)$$

and for any reduced rational pair $w_j = (u_j, v_j)$, we can conclude that the total length of all reduced rational pairs (u, v) representing null edge directions is less than $2P + 4$ where P is the pen perimeter. Thus we can substitute $2P$ for P in (4.3.5) and obtain the result that

$$b' \leq \frac{6}{\sqrt[3]{\pi}} P^{2/3} + O(P^{1/2} \log^{1/2} P). \tag{4.3.7}$$

A simple inductive argument suffices to prove (4.3.6). If $l \in \{0, 1\}$ we claim that $\sum_{0 \leq i \leq n} |\omega_{2i+l}| < |\omega_{2n+1+l}| - 1/|\omega_{2n+1+l}|$ where $|\omega_i|$ denotes the Euclidean length. This is clearly true for $n = 0$ since $1/|\omega_{l+1}| \leq 1/\sqrt{5}$, and the induction step follows immediately from the fact that $|\omega_{m-1}| - 1/|\omega_{m-1}| + |\omega_m| \leq |\omega_{m+1}| - 1/|\omega_{m+1}|$ for all $m \geq 1$. The following lemma completes the proof.

Lemma 4.3.3. *If (u_2, v_2) is a Stern-Peirce tree son of (u_1, v_1) then $\alpha - 1/\alpha + \beta \leq \gamma - 1/\gamma$ where $\alpha = \sqrt{u_1^2 + v_1^2}$, $\beta = \sqrt{u_2^2 + v_2^2}$, and $\gamma = \sqrt{(u_1 + u_2)^2 + (v_1 + v_2)^2}$.*

Proof. Let θ be the angle between (u_1, v_1) and (u_2, v_2). It follows from (4.1.1) that $\alpha\beta \sin\theta = u_1 v_2 - v_1 u_2 = \pm 1$. From the law of cosines we have

$$\begin{aligned} \gamma^2 = \alpha^2 + \beta^2 + 2\alpha\beta\cos\theta &= (\alpha + \beta)^2 - 4\alpha\beta\sin^2(\theta/2) \\ &> (\alpha + \beta)^2 - 4\alpha\beta\sin^2\theta = (\alpha + \beta)^2 - \frac{4}{\alpha\beta}. \end{aligned} \tag{4.3.8}$$

In order to show that $\gamma \geq \alpha + \beta - (1/\alpha - 1/\gamma)$, it is sufficient to have

$$\gamma^2 \geq (\alpha + \beta)^2 - 2\left(\frac{1}{\alpha} - \frac{1}{\gamma}\right)(\alpha + \beta) + \left(\frac{1}{\alpha} - \frac{1}{\gamma}\right)^2.$$

We shall show that

$$2\left(\frac{1}{\alpha} - \frac{1}{\gamma}\right)(\alpha + \beta) - \left(\frac{1}{\alpha} - \frac{1}{\gamma}\right)^2 \geq \frac{4}{\alpha\beta} \tag{4.3.9}$$

and add this to (4.3.8) to obtain the desired result.

Adding $\alpha^2 - 2\alpha\gamma$ to (4.3.8), we obtain

$$(\gamma - \alpha)^2 > 2\alpha^2 + 2\alpha(\beta - \gamma) + \beta^2 - \frac{4}{\alpha\beta} > \beta^2 - \frac{4}{\alpha\beta} \geq \beta^2 - 4/\sqrt{10}.$$

Since $\beta \geq \sqrt{5}$, it follows that $\gamma - \alpha > \sqrt{5 - 4/\sqrt{10}} > 1.9$. Now $4/\beta + 1/\alpha \leq 4/\sqrt{5} + 1/\sqrt{2} < 2.5$, hence $2(\gamma - \alpha) \geq 4/\beta + 1/\alpha$ and

$$\begin{aligned} 2\left(\frac{1}{\alpha} - \frac{1}{\gamma}\right)(\alpha + \beta) > 2\left(\frac{1}{\alpha} - \frac{1}{\gamma}\right)\gamma = 2\frac{\gamma - \alpha}{\alpha} \\ \geq \frac{1}{\alpha}\left(\frac{4}{\beta} + \frac{1}{\alpha}\right) = \frac{4}{\alpha\beta} + \frac{1}{\alpha^2} \geq \frac{4}{\alpha\beta} + \left(\frac{1}{\alpha} - \frac{1}{\gamma}\right)^2. \end{aligned}$$

Thus (4.3.9) holds as required. ∎

We now have a way to modify Algorithms 1 and 2 to limit the parameter b that determines the number of iterations not directly attributable to pen polygon edges. With this modification, the new parameter b' satisfies a bound $2^{2/3}$ times as large as the bound on the number a of actual pen polygon edges. Hence the execution counts and space bounds that depend on $a+b$ are all $O(P^{2/3})$ in the pen perimeter P. In fact this bound on $a+b$ is probably pessimistic because pen polygons that approach the bound on b tend to fall far short of the bound on a and vice versa.

On the other hand, our bound on a is fairly tight. One can easily construct k-vertex pen polygons whose perimeter is s_k, but it is natural to question how closely the results of Algorithms 1 and 2 can approach this bound when given a smooth brush shape $\mathcal{B}$. Figure 15a shows that in fact this bound is closely approached when $\mathcal{B}$ is a circle. The graph was obtained by taking a as a function of diameter and finding its least monotonic upper bound and its greatest monotonic lower bound; i.e., the bounds shown are as tight as possible subject to the condition that they must be monotonic.

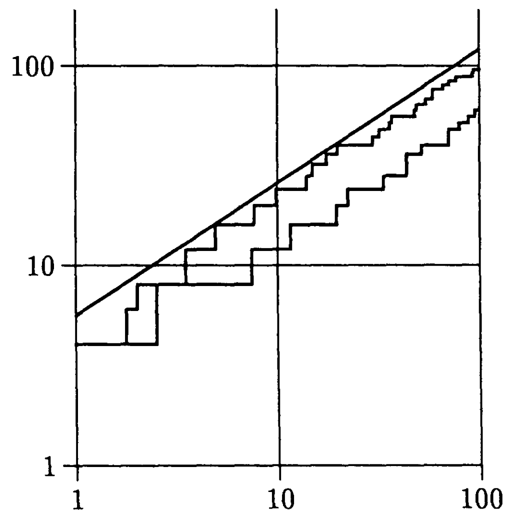

Fig. 15a. The range of a for circular pens as a function of diameter with the first term of the upper bound (4.3.5) for comparison.

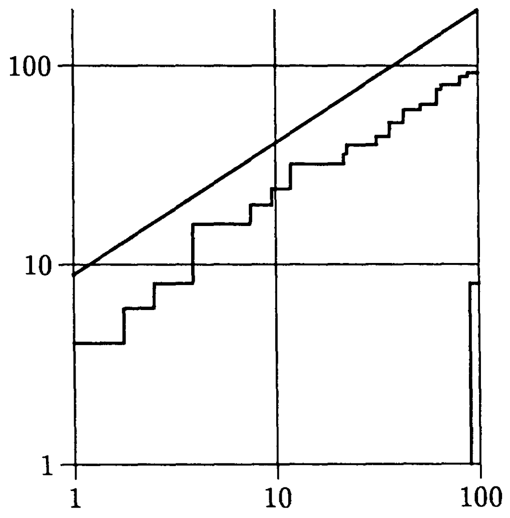

Fig. 15b. The range of b for circular pens as a function of diameter with the first term of (4.3.7) for comparison.

Figure 15b shows similar bounds on the number of missing edges b for pens generated by Algorithm 1 to approximate circles. Note that b actually remains well below our bound on b', and b can be 0 even when the diameter is as large as 90.5. Thus the modification to replace b with b' in the time bounds does not appear to be necessary for circles of reasonable size.

This modification is seldom likely to be worthwhile in practice because it significantly complicates the algorithm and it only pays off in unusual circumstances. On the other hand it probably is worthwhile to keep null edges out of the data structures because this significantly reduces the execution counts for Steps 2 and 3 of Algorithm 1 and Steps 3 and 4 of Algorithm 2 while reducing the total number of nodes in the data structures from $a+b+1$ to about $a+1$.

Since the space needed to represent the pen polygon is proportional to a which may be much smaller than the upper bound given by (4.3.5), we cannot claim that the running time is linear in the size of the input plus output even if we use the modification to reduce b. In fact the running time cannot be bounded by any function of the input plus output size. On the other hand the running time is sublinear in the physical size of the output and probably linear in its expected size over any reasonable probability distribution.

Conspicuously absent from any discussion of running time has been Algorithm 2c. In spite of favorable practical experience, it is very difficult to make any claims about the amount of backtracking in the algorithm as stated. If guaranteed performance is desired, it may be necessary to switch to Algorithm 2b at some point.

Chapter 5

Building Envelopes with Integer Offsets

We have designed pens to produce accurate and uniform stroke weight by producing envelopes with integer offset vectors. The purpose of this chapter is to investigate a more direct approach that can sometimes produce better results: Given a suitable brush-trajectory description, choose a desirable set of offset vectors and use these to construct a digitized version of the envelope. This problem can be thought of as a generalization of the problem of constructing pen envelopes.

First consider just how desirable the offset vectors obtained from a pen polygon are. Recall that the error bound (2.2.4) in Theorem 2.2.3, which is extended to curved strokes by Corollary 2.2.5, is proportional to the tangent of the angle θ between the offset vector and the normal to the trajectory direction. (We refer to θ as the *offset angle.*) As we have seen in Chapter 2, this bound is often achieved within a small factor when $w \tan\theta$ is small and it can be almost exactly met in certain circumstances for arbitrarily large widths w.

Since the offset vector for a pen and a particular trajectory direction is the difference between the points of support in that direction, and since the algorithms of Chapter 4 were designed to approximate closely the given brush shape $\mathcal{B}$, it follows that the value of $\tan\theta$ is largely determined by $\mathcal{B}$. In fact the use of retention points strictly limits the deviation between the actual offset angle and this ideal value. Thus, long narrow brush shapes result in pens that have a large value of $\tan\theta$ for some directions. On the other hand if $\mathcal{B}$ is a circle of diameter d, then the following lemma shows that the maximum of $\tan\theta$ over all directions is $\Theta(d^{-1/2})$.

Lemma 5.1. *Let $\mathcal{P}$ be a pen produced by Algorithm 1 from a circle of some diameter d centered on the origin, and let θ be the maximum offset angle of $\mathcal{P}$ over all possible trajectory directions. Then*

$$\frac{1}{2\sqrt{d+1}} \le \tan\theta \le \frac{\sqrt{(24+16\sqrt{2})d+1}}{4d-2\sqrt{2}}.$$

Proof. First consider the lower bound and let (u,v) be a reduced rational pair for the edge direction counterclockwise adjacent to $(1,0)$. The length of this edge must be at least $\frac{1}{2}\sqrt{u^2+v^2}$, so one of its endpoints must be at a distance of at least $\frac{1}{4}\sqrt{u^2+v^2}$ from the line $ux+vy=0$. Furthermore the edge is at a distance of at most $\frac{1}{2}d+\frac{3}{8}$ from the origin so

$$\tan\theta \ge \frac{\sqrt{u^2+v^2}}{2d+\frac{3}{2}}.$$

Since $\mathcal{P}$ has a vertex at which the exterior angle is at least $\arctan(v/u)$, it must have an offset angle of at least half this much. (A vertex that makes offset angles of θ_1 and θ_2 with respect to directions perpendicular to its adjacent sides has an exterior angle of $\theta_1 + \theta_2$; cf. Figure 16.) Thus $\tan\theta$ is at least

$$\max\left(\frac{\sqrt{u^2+v^2}}{2d+\frac{3}{2}}, \frac{v}{u+\sqrt{u^2+v^2}}\right). \tag{5.1}$$

To obtain a lower bound on (5.1), observe that each term is monotone increasing in v and the first term is monotone increasing in u while the second term is monotone decreasing. Hence we obtain a lower bound by substituting $(u,v) = (\sqrt{d},1)$ and observing that both terms are greater than $1/(2\sqrt{d+1})$.

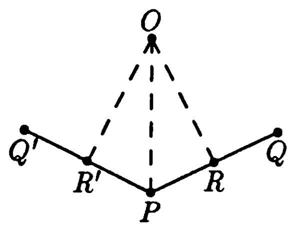

Fig. 16. A construction for obtaining bounds on $\tan\theta$.

Figure 16 illustrates the construction for the upper bound. Let PQ be an edge of $\mathcal{P}$ and let R be the point on this edge closest to the origin O. Similarly let $Q'P$ be the other edge incident on P and let R' be the point of closest approach as shown in the figure. When the offset vector is in the direction of P, the offset angle is at most POR or POR', whichever is greater. Since the length of OP is at most $\frac{1}{2}d + \frac{3}{8}$ and the lengths of OR and OR' are at least $\frac{1}{2}d - \sqrt{2}/4$, the lengths of PR and PR' are at most

$$\sqrt{\left(\frac{1}{2}d+\frac{3}{8}\right)^2 - \left(\frac{1}{2}d-\frac{\sqrt{2}}{4}\right)^2} = \frac{\sqrt{(24+16\sqrt{2})d+1}}{8}$$

and hence the result follows. ▮

5.1. Integer Offsets for Brush Strokes of Nonuniform Width

We have seen that a polygonal pen stroke has offset angles that are $\Theta(d^{-1/2})$ when the pen is chosen to approximate a circle of diameter d. That is, we can achieve such offset angles for brush strokes of uniform width d. What must be done in order to achieve such offset angle for nonuniform width strokes, and when is this desirable? The main advantage of integer offset vectors is that they produce envelopes of nearly uniform apparent weight; they are not likely to be beneficial when the width changes rapidly as a function of arc length along the trajectory.

Consider the envelope of a noncircular brush $\mathcal{B}$ with respect to a slowly curving trajectory. If the curvature of the trajectory is sufficiently low, then the width of the envelope at any point is essentially the width of the brush perpendicular to the direction of the trajectory at that point. For simplicity we shall assume that $\mathcal{B}$ is symmetrical about the origin; otherwise the techniques of Section A.3 can be used to reduce to this case. We wish to approximate the envelope with respect to $\mathcal{B}$ by a *dynamic pen envelope* as follows: Divide the trajectory into segments and use a unique integer offset for each segment in such a way as to achieve moderate offset angles. In Section 3.2, we saw how the envelope of a polygonal pen can be constructed in this manner.

A dynamic pen envelope may be specified as follows: Let the trajectory be given by the path $\big(x(t), y(t)\big)$ for $0 \le t \le a$ where $a > 0$, and let there be values $t_0, t_1, \ldots, t_n$ such that $t_0 = 0$, $t_n = a$, and $t_{i-1} < t_i$ for $1 \le i \le n$. Furthermore, let there be integer offset vectors (u_i, v_i) for $1 \le i \le n$, and let the envelope be described as follows: For $i = 1, 2, \ldots, n$, take the curves $\big(x(t)+\frac{1}{2}u_i, y(t)+\frac{1}{2}v_i\big)$ for $t_{i-1} \le t \le t_i$, followed by the curves $\big(x(-t)-\frac{1}{2}u_i, y(-t)-\frac{1}{2}v_i\big)$ for $-t_i \le t \le -t_{i-1}$. Successive curves must be connected in some way, e.g. by straight lines.

The *idealized width* of the above dynamic pen envelope with respect to the given parameterization is a function of t equal to $u_i \sin\phi - v_i \cos\phi$, where $(\cos\phi, \sin\phi)$ is a unit vector in the direction of $\big(x'(t), y'(t)\big)$ and i is chosen so that $t_{i-1} \le t \le t_i$. (The angle ϕ is called the *trajectory direction angle.*) The purpose of the requirement that the curvature of the trajectory "must not be too large" is to ensure that the idealized width of the dynamic pen envelope is a reasonable approximation to the width of the envelope with respect to $\mathcal{B}$. As the following lemma shows, it is not possible to ensure that the idealized width will always be continuous unless some of the offset angles are allowed to become large.

Lemma 5.1.1. *Consider a dynamic pen envelope as described above. If there are t_α and t_β such that $0 \le t_\alpha < t_\beta \le a$; if the trajectory direction angle and idealized width are continuous, piecewise real analytic functions $\phi(t)$ and $w(t)$ on $t_\alpha \le t \le t_\beta$; and if $\phi(t)$ is monotone increasing or monotone decreasing; then the maximum offset angle is at least*

$$\arctan\left|\frac{w(t_\beta) - w(t_\alpha)}{\big(\phi(t_\beta) - \phi(t_\alpha)\big) \cdot \max_{t_\alpha \le t \le t_\beta} w(t)}\right|.$$

Proof. Assume without loss of generality that $\phi(t)$ and $w(t)$ are real analytic on each interval $[t_{i-1}, t_i]$; i.e., subdivide each interval until ϕ and w are real analytic in each subinterval. Now let $\Delta w = w(t_\beta) - w(t_\alpha)$, $\Delta\phi = \phi(t_\beta) - \phi(t_\alpha)$, and $w_{\max} = \max_{t_\alpha \le t \le t_\beta} w(t)$. Whenever $1 \le i \le n$ and $t_{i-1} \le t \le t_i$, the idealized width $w(t)$ is $l_i \cos\big(\theta(t)\big)$ where $l_i = \sqrt{u_i^2 + v_i^2}$ and $\theta(t)$ is the offset angle $\big|\phi(t) - \arctan(-u_i/v_i)\big|$. Thus $\frac{dw}{d\phi} = \pm l_i \sin\big(\theta(t)\big)$ and by the mean value theorem, there must be some t on $t_{i-1} \le t \le t_i$ where $\frac{dw}{d\phi} = \big(w(t_i) - w(t_{i-1})\big)/\big(\phi(t_i) - \phi(t_{i-1})\big)$.

We shall now show that there must be some t on $t_\alpha \le t \le t_\beta$ where $\left|\frac{dw}{d\phi}\right| \ge |\Delta w/\Delta\phi|$. Otherwise there must be some bound $b < |\Delta w/\Delta\phi|$ such that

$$\left|\frac{w(t_i) - w(t_{i-1})}{\phi(t_i) - \phi(t_{i-1})}\right| \le b$$

for $1 \le i \le n$. Thus $|w(t_i) - w(t_{i-1})| \le b(\phi(t_i) - \phi(t_{i-1}))$ and

$$\Delta w \le \sum_{\alpha<i\le\beta} |w(t_i) - w(t_{i-1})| \le b \sum_{\alpha<i\le\beta} \phi(t_i) - \phi(t_{i-1}) \le b\Delta\phi.$$

Since this contradicts the assumption, we know that there exists a t between t_α and t_β such that $|\frac{dw}{d\phi}| \ge |\Delta w/\Delta\phi|$. Now $w_{\max} \ge l_i \cos\theta$, so $w_{\max} \tan\theta \ge l_i \sin\theta = |\frac{dw}{d\phi}| \ge |\Delta w/\Delta\phi|$ and the lemma follows. ∎

Let $\mathcal{B}$ be a noncircular brush with widths w_α and w_β in two directions ϕ_α and ϕ_β. Now select any suitably straight trajectory $\mathcal{T}$ whose terminal directions are perpendicular to ϕ_α and ϕ_β, and construct a dynamic pen envelope as described above. If this envelope can be considered an approximation to the envelope of $\mathcal{T}$ with respect to $\mathcal{B}$ then $w(a) - w(0) \approx w_\beta - w_\alpha$. Thus the maximum offset vector must be nearly as large as the bound given in the lemma, and $\mathcal{B}$ can be selected so as to make this can be arbitrarily large.

We have seen that no matter how straight the trajectory, the offset angle can approach 90° unless we allow the idealized width to be discontinuous. That is, there must be glitches where the offset vectors change as shown in Figure 17. The digitized envelope shown in the figure has three different integer offset vectors: $(1,-1)$, $(2,-2)$, and $(1,-3)$. The portion of the envelope where each applies is delimited by dashed lines. When the boundaries between such regions are carefully placed, the glitches there are not particularly obtrusive even though the change in width is significant. For instance, the trajectory used to generate the figure was a line of slope 1/6 and the three offsets have idealized widths of $7/\sqrt{37}$, $14/\sqrt{37}$, and $19/\sqrt{37}$.

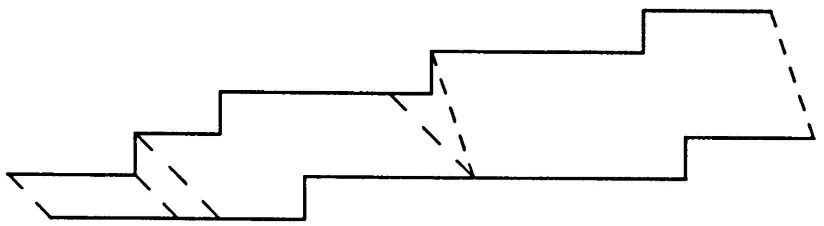

Fig. 17. A digitized envelope constructed from three integer offsets.

If we drop the requirement that the idealized width must be continuous, then it can be made to approximate any reasonable function. We need only break the trajectory into a sufficient number of segments so that the desired width function is nearly constant on each segment. Any piecewise real analytic function can be approximated in this manner.

We thus define the *dynamic brush envelope* of a trajectory T with respect to a width function W as follows: Let $T = (x(t), y(t))$ be a non-constant, C^1 continuous, piecewise real analytic function of a parameter t on a real interval $0 \le t \le a$, and let W be a continuous real analytic function on the same interval. Let the dynamic brush for T and W at time t be the unique line segment of length $W(t)$ perpendicular to T at time t with midpoint $(x(t), y(t))$. The dynamic brush envelope is the union of all such dynamic brushes for $0 \le t \le a$.

The concept of dynamic brush envelopes is not limited with regard to the curvature of the trajectory of the rate of change of the width function. The only problem is that the previously mentioned scheme of using the directional width of the brush $\mathcal{B}$ to define the width function is not the same as taking the ordinary envelope with respect to $\mathcal{B}$. Figure 18 illustrates this difference in an extreme case. Figure 18a shows a rectangular brush $\mathcal{B}$ and a portion of its envelope with respect to a parabolic trajectory T; Figure 18b shows a portion of the dynamic brush envelope of the same trajectory T with respect to the directional width of $\mathcal{B}$ perpendicular to T. The dashed lines give the location of the dynamic brush at various times.

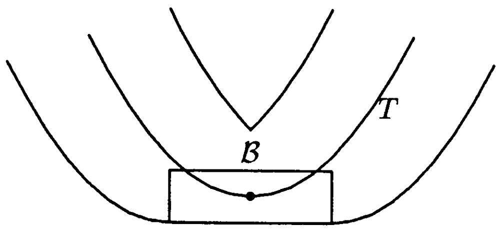

Fig. 18a. The envelope of a rectangular brush with respect to a parabolic trajectory

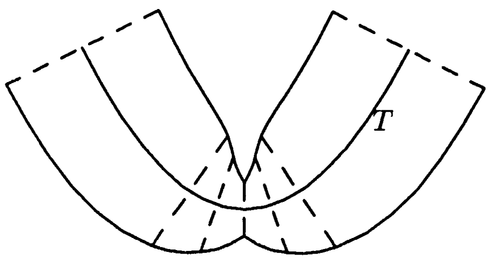

Fig. 18b. A dynamic brush envelope corresponding to Figure 18a.

As can be seen from Figure 18b, dynamic brush envelopes can have strange behavior when the width function changes rapidly. When the width function is more controlled, the flexibility of dynamic brush envelopes can be used to great advantage. Thus we shall consider dynamic brush envelopes with no rapid width variations and attempt to approximate them with dynamic pen envelopes that have small offset angles.

Just how rapid can these width variations be before integer offset vectors are no longer beneficial? Consider a straight line trajectory $x\cos\phi + y\sin\phi = c$ with width function $W(t)$. There will be a set of offset vectors (u, v) for which the offset angle $\arctan(v/u) - \phi$ is less than some function of the idealized width $u\cos\phi + v\sin\phi$, and we can just choose whichever offset minimizes the difference between $W(t)$ and the idealized width. The length of the portion of the trajectory for which any particular offset is used depends on the rate of change of $W(t)$ and the separation between adjacent idealized widths. We want these lengths to be great enough so that the integer offset vectors will have a beneficial effect on the average apparent weight.

All of this depends on what function of the idealized width is chosen. We probably want the idealized width to be continuous when $W(t)$ is constant, and as we shall see in Section A.3 of the appendix, this means that the offset angle must be $\Omega(w^{-1/2})$ in the idealized width w.

Another way of looking at it is that there is a tradeoff between the accuracy with which the idealized width approximates $W(t)$ and the bound of $(\bar{w}/s)\tan\theta$ on the relative difference between the average apparent weight and the ideal average weight. This allows for an absolute error in apparent weight on the order of $\tan\theta$

when the arc length s is on the order of w. Thus it seems reasonable to make the bound on $\tan\theta$ approximately equal to the best accuracy we can hope to attain for the idealized width. This accuracy is severely limited for simple rational slope trajectories, so we just want $\tan\theta$ as small as possible in such cases; increasing the bound on $\tan\theta$ only reduces the best possible error in idealized width.

Suppose we require the offset angle θ to satisfy $-c \le \tan\theta < c$ for some c, and we also require the idealized width to be in some interval $[w - \frac{1}{2}\Delta w, w + \frac{1}{2}\Delta w)$. This forces the integer offset vectors to lie in a trapezoidal region whose area is $2wc\Delta w$. The number of such offset vectors is just the area of the digitization of a shifted copy of the trapezoid, and Lemma 2.2.2 shows that this is between $(2wc - \sqrt{2})(\Delta w - \sqrt{2})$ and $(2wc + \sqrt{2})(\Delta w + \sqrt{2})$. Thus the average number of allowable integer offset vectors per unit change in idealized width is about $2wc$.

Even if the idealized widths are uniformly distributed, their error in approximating $W(t)$ will be about $1/(4wc)$ where w is the current value of $W(t)$ and c is the bound on $\tan\theta$ as a function of w. Thus our guess about the optimum point on the tradeoff is that $c \approx 1/(wc)$; i.e., $c \approx 1/\sqrt{w}$. Let us assume that

$$\tan\theta < \gamma/\sqrt{w} \tag{5.1.1}$$

for some constant γ.

Then the arc length required for $W(t)$ to change by $1/(4\gamma\sqrt{w})$ is approximately $1/(4\gamma\frac{dW}{ds}\sqrt{w})$. This is the minimum average arc length over which we expect each integer offset to apply. That is if $\frac{dW}{ds}$ and W remain fairly constant for long enough then we can expect the average arc length per offset to be at least this much. In order to gain full advantage from integer offsets this average arc length should probably be at least on the order of w so that the $(\bar{w}/s)\tan\theta$ bound on relative error in average apparent weight compares favorably with the difference between idealized weight and $W(t)$. This happens when $\frac{dW}{ds} = O(w^{-3/2})$.

On the other hand if the trajectory is nearly vertical or horizontal then each offset may apply over an arc length of $1/\left|\frac{dW}{ds}\right|$ and $\tan\theta$ might never exceed $1/w$. In this case we only need $(w/s)\tan\theta < 1$ or $\left|\frac{dW}{ds}\right| < 1$. A reasonable compromise is to require that

$$\frac{dW}{ds} < \frac{1}{4\gamma^2 w}, \tag{5.1.2}$$

so that the average arc length per offset will be at least $w\tan\theta$.

5.2. Digital Equivalence and Straightness

Before we can deal with the problem of how to smooth out the glitches where offsets change, we need a concept of smoothness that is applicable to the edges of digital regions. We shall take advantage of the fact that digitization is defined on paths so that the boundary of the digitization of an envelope E can be obtained by digitizing the boundary $B(E)$. The desired concept of smoothness can be based on the ability to find a nearly linear path with a matching digitization.

Our concept of smoothness will be based on the intuitive idea that the digital region representing a brush envelope should be describable as the region bounded by the digitization of a "smooth path." Given a digitized path $\mathcal{S}$, we wish to determine how wide a range of directions must be covered by a path $\mathcal{T}$ that is *digitally equivalent* to $\mathcal{S}$.

In general, two paths are digitally equivalent if they have the same digitization. This is essentially the same idea as was used in Section 2.1 to define equivalence classes of rational slope lines for use in measuring stroke weight.

We shall proceed by taking the given path and, if it is smooth enough, finding a digitally equivalent polygonal path for which the smoothness can be evaluated recursively. These paths will be restricted to have derivative vectors confined to certain subranges of the positive quadrant. For each $k \geq 1$ let $\mathcal{X}_k$ be the set of paths $\bigl(x(t), y(t)\bigr)$ whose direction vector $\bigl(x'(t), y'(t)\bigr)$ is always a nonnegative linear combination of $(k+1, 1)$ and $(k, 1)$; and let $\mathcal{Y}_k$ be a similar set of paths whose direction vectors are always nonnegative combinations of $(1, k)$ and $(1, k+1)$.

We now define a class of affine transformations T_k for the purpose of dealing with paths in $\mathcal{X}_k$:

$$T_k(x,y) = \begin{pmatrix} 1 & -k \\ -1 & k+1 \end{pmatrix} \begin{pmatrix} x \\ y - \frac{1}{2} \end{pmatrix}.$$

The transformation T_k maps lines of slope $1/k$ into vertical lines, and it maps lines of slope $1/(k+1)$ into horizontal lines. We also need to be able to do this for slope pairs $k+1$ and k for integers k, so we define $U_k(x,y) = S\bigl(T_k(y,x)\bigr)$ where $S(x,y) = (y,x)$. For future reference

$$T_k^{-1}(x,y) = \begin{pmatrix} k+1 & k \\ 1 & 1 \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix} + \begin{pmatrix} 0 \\ \frac{1}{2} \end{pmatrix}$$

and $U_k^{-1}(x,y) = S\bigl(T_k^{-1}(y,x)\bigr)$.

When restricted to the set $\mathbb{Z}^2 + (\frac{1}{2}, \frac{1}{2})$ of pixel centers, each transformation T_k is 1 to 1 and onto. Thus any two regions R and R' contain the same set of pixel centers if and only if $T_k(R)$ and $T_k(R')$ do. The following lemma is a consequence of this. The proof is deferred to the appendix because it depends on the precise statement of the correspondence between the digitizations of regions and boundary paths. (See Lemma A.4.2.)

Lemma 5.2.1. *If ℓ and ℓ' are infinite paths in $\mathcal{X}_k$ for some integer k, then ℓ and ℓ' are digitally equivalent if and only if $T_k(\ell)$ and $T_k(\ell')$ are; similarly for $\{\ell, \ell'\} \subset \mathcal{Y}_k$, ℓ and ℓ' are digitally equivalent if and only if $U_k(\ell)$ and $U_k(\ell')$ are.* ∎

The reason for the restriction to infinite paths in the above lemma is that there could otherwise be differences in the digitizations of the transformed paths near their endpoints. In other words, it may be necessary to shorten one of the paths somewhat in order to achieve digital equivalence.

Recall that the digitization of a path $X(t)$ passes through points obtained by rounding the coordinates of $X(t)$ to integers. That is we divide the plane into squares of the form

$$Q(m,n) = \{\, (x,y) \mid m - \tfrac{1}{2} < x \leq m + \tfrac{1}{2} \text{ and } n - \tfrac{1}{2} \leq y < n + \tfrac{1}{2} \,\}$$

and connect in order the center of each square through which $X(t)$ passes. Paths formed in this way by connecting points in $\mathbb{Z}^2$ by vertical and horizontal line segments are called *digital paths*. A digital path $X = (x, y)$ has the property that for any t, either $x(t) \in \mathbb{Z}$ or $y(t) \in \mathbb{Z}$.

If a path X passes through pixel centers, its digitization $\mathcal{D}(X)$ may have vertices (m, n) for which the range of X does not intersect $Q(m, n)$, but it is true that the range of X must be contained in the union of all $Q(m, n)$ over all vertices (m, n) of $\mathcal{D}(X)$. This union of $Q(m, n)$ is called the *digitization region* of $\mathcal{D}(X)$. Note that the digitization region is not a digital region because $Q(m, n)$ is centered on the point (m, n), not on a pixel center.

Figure 19 illustrates the point behind Lemma 5.2.1. It shows the digitization region R of a digital path $\mathcal{Q} = \mathcal{D}(\ell)$ and the image under T_1^{-1} of the digitization region R' of another digital path $\mathcal{Q}' = \mathcal{D}(T_1(\ell))$, where ℓ is the curve represented by the dotted line. Observe how the linear transformation distorts the squares comprising R' into parallelograms. The properties of the transformation T_1 ensure that these parallelograms are all contained in R so that if ℓ' is a path digitally equivalent to $\mathcal{Q}'$ then $T^{-1}(\ell')$ will be digitally equivalent to a subset of $\mathcal{Q}$. Furthermore $T_1^{-1}(R')$ fills up R so that, except for small regions near either end of the figure, no path in $\mathcal{X}_1$ can have any part of its range in common with $R \setminus T_1^{-1}(R')$.

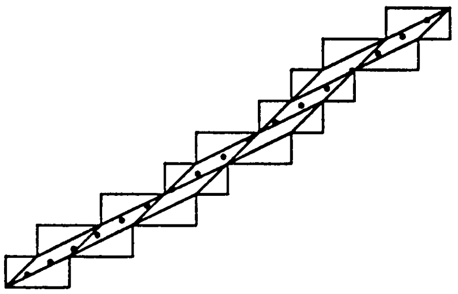

Fig. 19. A comparison of the digitization region of $\mathcal{D}(\ell)$ and T_1^{-1} of the digitization region of $\mathcal{D}(T_1(\ell))$.

If we have an infinite digital path that is the digitization of some path $\ell \in \mathcal{X}_k$, then Lemma 5.2.1 implies that $\mathcal{D}(T_k(\ell))$ is independent of which such ℓ is chosen. We need a way of taking a digital path and determining for what k it is equal to $\mathcal{D}(\ell)$ for some $\ell \in \mathcal{X}_k$, and if there is such a k, then what is $\mathcal{D}(T_k(\ell))$.

To simplify the notation in the following lemma, let a *nonnegative digital path* be a digital path $(x(t), y(t))$ such that $x'(t)$ and $y'(t)$ are always nonnegative. In addition, if $\mathcal{Q}$ is a nonnegative digital path define $L_x(\mathcal{Q})$ to be the set of all lengths of horizontal edges in $\mathcal{Q}$, and let $L_y(\mathcal{Q})$ be the set of vertical edge lengths.

Let the *polygonization* of a nonnegative digital path $\mathcal{Q}$ be a polygonal path built from line segments joining the midpoints of each pair of adjacent edges of $\mathcal{Q}$. For instance Figure 20 shows the polygonization of the path $\mathcal{D}(\ell)$ from Figure 19. Removing the first and last edges of the polygonization yields a polygonal path that connects the centers of the parallelograms that make up the transformed digitization region in Figure 19. The following lemma makes this relationship precise.

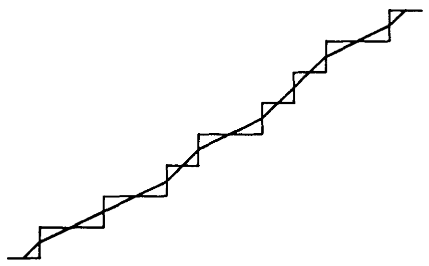

Fig. 20. The digital path $\mathcal{D}(\ell)$ from Figure 19 and its polygonization.

Lemma 5.2.2. *Let $\mathcal{Q}$ be an infinite nonnegative digital path. There exists a path $\ell \in \mathcal{X}_k$ such that $\mathcal{D}(\ell) = \mathcal{Q}$ if and only if $L_x(\mathcal{Q}) \subseteq \{k, k+1\}$ and $L_y(\mathcal{Q}) = \{1\}$, in which case $\mathcal{D}(T_k(\ell)) = T_k(\mathcal{Q}')$ where $\mathcal{Q}'$ is the polygonization of $\mathcal{Q}$. Similarly there exists $\ell \in \mathcal{Y}_k$ with $\mathcal{D}(\ell) = \mathcal{Q}$ when $L_y(\mathcal{Q}) \subseteq \{k, k+1\}$ and $L_x(\mathcal{Q}) = \{1\}$. Then $\mathcal{D}(U_k(\ell)) = U_k(\mathcal{Q}')$.*

Proof. Let (x_1, y_1) and (x_2, y_2) be two points in the range of ℓ. If $y_2 - y_1 = 1$ then $k \le x_2 - x_1 \le k+1$ when $\ell \in \mathcal{X}_k$, and $0 < x_2 - x_1 \le 1$ when $\ell \in \mathcal{Y}_k$. Applying this for integer y_1 and y_2 shows that $L_x(\mathcal{Q})$ must satisfy the required containment when $\mathcal{D}(\ell) = \mathcal{Q}$; a similar argument with $x_2 - x_1 = 1$ proves the containment for $L_y(\mathcal{Q})$.

Conversely if $L_x(\mathcal{Q})$ and $L_y(\mathcal{Q})$ do satisfy the containments then $\mathcal{Q}'$ has the correct digitization and is in $\mathcal{X}_k$ or $\mathcal{Y}_k$ as required.

We have just seen that the polygonization $\mathcal{Q}'$ is digitally equivalent to $\mathcal{Q}$ and ℓ when $\mathcal{Q}$ has the required edge lengths. Hence by Lemma 5.2.1, $T_k(\mathcal{Q}')$ or $U_k(\mathcal{Q}')$ is digitally equivalent to $T_k(\ell)$ or $U_k(\ell)$. All that remains is to show that whichever of $T_k(\mathcal{Q}')$ or $U_k(\mathcal{Q}')$ we are dealing with is a digital path. The transformations T_k and U_k have been carefully chosen to map edges parallel to those of $\mathcal{Q}'$ to vertical and horizontal segments. When the transformation is T_k, the vertices of $\mathcal{Q}'$ occur at the middle of unit length vertical edges and hence have integer x-coordinates and y-coordinates congruent to $\frac{1}{2}$ (modulo 1); when the transformation is U_k the unit edges are horizontal and it is the y-coordinates that are integers. Either way, the transformed vertices lie in $\mathbb{Z}^2$ as required. ∎

The paths that we will actually be dealing with are not infinite as required by the lemmas. We need to determine whether a finite digital path is a subset of a suitable infinite one. This is not difficult: Let $\mathcal{Q}$ be the given digital path and rotate it to obtain a nonnegative digital path $\mathcal{Q}_1$. If this is not possible, then $\mathcal{Q}$ has *straightness* 0. Now determine whether it is possible to extend the initial and final edges of $\mathcal{Q}_1$ so as to obtain a digital path $\mathcal{Q}_1'$ so that $L_x(\mathcal{Q}_1')$ and $L_y(\mathcal{Q}_1')$ are as required in Lemma 5.2.2. If at either endpoint there is more than one way to make such an extension, then that entire edge should be deleted. If the construction of $\mathcal{Q}_1'$ is successful then $\mathcal{Q}_1$ is the digitization of some path in $\mathcal{X}_k$ or $\mathcal{Y}_k$. Now let $\mathcal{Q}_2$ be the polygonization of $\mathcal{Q}_1'$ transformed by whichever of T_k or U_k is appropriate.

This process can be repeated indefinitely to produce $\mathcal{Q}_1$, $\mathcal{Q}_2$, $\mathcal{Q}_3$, ... until at some stage the edge lengths fail to satisfy the constraints. If $\mathcal{Q}_i$ is that last successfully produced path in the sequence then the straightness of $\mathcal{Q}$ is i. If at some point a path of length 0 is obtained, then the process will never terminate and Q is a subset of the digitization of an infinite straight line. In this case we say that the straightness is infinite. The problem of recognizing straight lines in this way is covered extensively by Rothstein and Weiman in [25].

Lemmas 5.2.1 and 5.2.2 guarantee that at each stage, any path that digitizes to Q_i can be extended without increasing its range of directions and then transformed by a linear transformation so as to obtain a path that digitizes to Q. Hence the above process determines how wide a variation in directions is necessary in order for a path to have Q as its digitization. It could be refined so as to determine the exact range of directions required rather than just determining how many times T_k or U_k can be applied and still keep the directions confined to one quadrant, but such refinement is not necessary for our purposes.

5.3. Smoothly Changing Offsets

In Section 5.1 we investigated the possibility of breaking the trajectory into sections and using a different integer offset for each section, but we still have not decided exactly what to do at the ends of such sections. We now investigate the possibilities in the light of the straightness criteria of the previous section.

Figure 21a illustrates the basic problem. It shows a portion of an envelope boundary with an integer offset of $(-2,3)$ on the left and $(0,4)$ on the right. The trajectory is the dashed line through the center of the figure, and the solid lines are copies of overlapping portions of the trajectory shifted by $\pm\frac{1}{2}$ times the integer offset vectors. These overlapping boundaries must be connected together somehow so that the digitizations of the composite boundaries thus formed will be as smooth as possible.

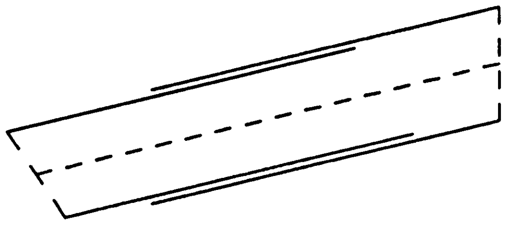
Fig. 21a. Undigitized envelope boundaries with different integer offsets.

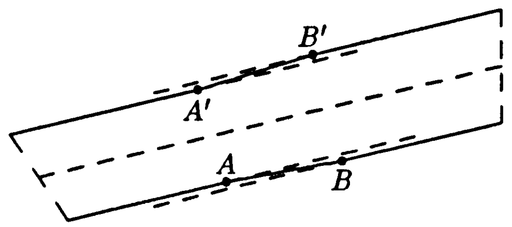

Fig. 21b. A possible approach to smoothing before digitization.

Figure 21b shows one possible approach to this problem: Find two points A and A' offset by $\pm\frac{1}{2}(-2,3)$ from the trajectory at the left side of the overlap, and connect them to two points B and B' offset by $\pm\frac{1}{2}(0,4)$ at the right side of the overlap. By increasing the distance between A and B and between A' and B', the undigitized boundaries can be made as smooth as desired at the expense of reducing the range over which the integer offset vectors apply.

It is of course the digitized version of the envelope boundary that should be as smooth as possible. Figure 22a shows the digitizations of the envelope boundaries of from Figure 21a. The original envelope boundaries are shown as dashed lines superimposed on their digitizations. Portions of the digitized boundaries for the two offsets coincide, but at intervals there are pixels that are inside of the $(0,4)$ envelope but not the $(-2,3)$ envelope. This row of "changeable pixels" could be continued indefinitely if the overlap region were long enough and the trajectory straight enough.

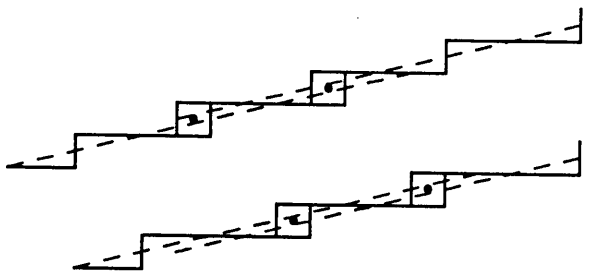
Fig. 22a. The digitization of Figure 21a.

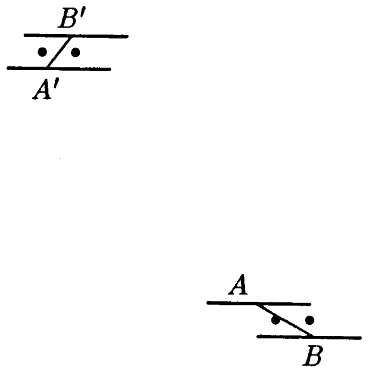

Fig. 22b. The polygonization of Figure 22a transformed by T_3. (Not to scale.)

There is no room to show points A, B, A', and B' in Figure 22a, but regardless of where they are placed there will be two x-values x_0 and x_0' that determine which changeable pixels will be part of the final digitized envelope: A changeable pixel centered at (x,y) is part of the envelope if and only if $x > x_0$ and (x,y) is on the lower edge of the envelope, or $x > x_0'$ and (x,y) is on the upper edge. Figure 22b shows this more clearly. Note that the extreme nature of the transformation being applied makes it necessary to show the two sides of the stroke closer together than they should be, but the neighborhood of each side is drawn to scale. Pixel centers within the transformed envelope correspond directly to pixel centers in the original envelope. The centers of the changeable pixels that appear in Figure 22a are shown as bold dots in Figure 22b. The bold horizontal lines in Figure 22b correspond to the dotted lines in Figure 22a, while the diagonal lines in Figure 22b correspond to the connecting lines in Figure 21b.

What we have seen so far can be summarized informally as follows: The straightness of the digitized envelope depends entirely on which changeable pixels within the overlap region are included. It is possible to achieve any desired degree of straightness adding long enough transition lines, but as Figure 23 illustrates, the cost can be very high.

Figure 23 shows what can happen if we insist on making an offset transition at a specific point C on the trajectory. If transition points A and B are too close together as shown in Figure 23a, then there is a nonmonotonicity in the digitized boundary directly below point C. This can be avoided by moving points A and B further apart as shown in Figure 23b, but then the width of the digitized envelope is very uneven. No integer offset vectors apply to the portion of the envelope between points A and B, and the result can be disastrous when these points are far apart.

This example shows that extremely long transition lines are very dangerous

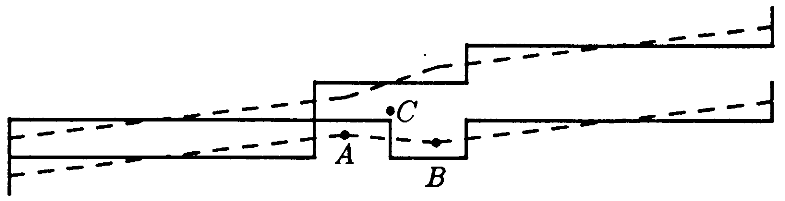

Fig. 23a. An unsuccessful attempt at smoothing before digitization.

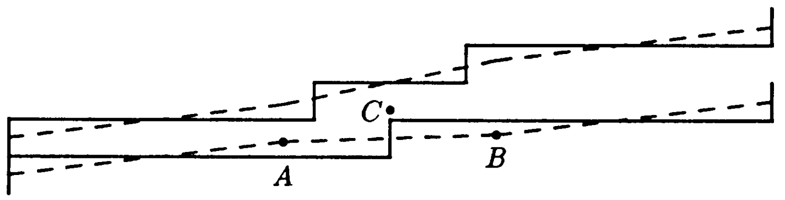

Fig. 23b. Large unevenness due to smoothing before digitization.

and sometimes we have no choice but to relocate the transition so as to increase the straightness. This involves locating the changeable pixels and deciding where the transition should be made so as to include the right ones. A well placed abrupt transition can produce smoother, straighter digitized envelope boundaries than a poorly placed gradual transition. The rest of this section will be devoted to controlling the placement of abrupt transitions.

To formalize the idea of a transition, let $X(t)$ be a path defined for $a \le t \le b$, and let (u_1, v_1) and (u_2, v_2) be two vectors in $\mathbb{Z}^2$ such that $y'u_1 - x'v_1$ and $y'u_2 - x'v_2$ are both positive or both negative for any direction (x', y') tangent to $X(t)$. We say that X, (u_1, v_1), and (u_2, v_2) form an *offset change problem*. The line segment whose endpoints are $\pm\frac{1}{2}(u_1, v_1)$ is called the *incoming pen*, and the line segment with endpoints $\pm\frac{1}{2}(u_2, v_2)$ is called the *outgoing pen*.

Let $[c_1, c_2]$ be a subinterval of $[a, b]$, and for $i = 1, 2$ let ℓ_i be the line parallel to (u_i, v_i) through $X(c_i)$. The offset change problem is said to be *simple* for the interval $[c_1, c_2]$ if the following hold: 1) The curves $X(t) \pm \frac{1}{2}(u_i, v_i)$ each cross each of the lines ℓ_j exactly once on $a \le t \le b$, for $1 \le i, j \le 2$. 2) No such curve may cross any line parallel to ℓ_1 or ℓ_2 more than once. 3) The lines ℓ_1 and ℓ_2 divide $\mathbb{R}^2$ into at most four regions, and one of these regions R contains segments of all four curves $X(t) \pm (u_i, v_i)$, where each such segment has one endpoint on ℓ_1 and one endpoint on ℓ_2. 4) None of the curves $X(t) \pm (u_i, v_i)$ include any points in common anywhere in R or on the boundary of R. The region R is called the *simplicity region* for $[c_1, c_2]$. Refer to Figure 24 for an illustration of an offset change problem that satisfies this definition.

Given an offset change problem as above, let E_1 be the envelope with respect to the incoming pen, of $X(t)$ for $a \le t \le b$. Similarly let E_2 be the envelope of the outgoing pen. (In Figure 24, $E_1 = A_1^+ B_1^+ B_1^- A_1^-$ and $E_2 = A_2^+ B_2^+ B_2^- A_2^-$.) If the offset change problem is simple for $[c_1, c_2]$ then Lemma 5.3.1 below shows that either $E_1 \cap R \subset E_2 \cap R$ or $E_2 \cap R \subset E_1 \cap R$, where R is the region described in the definition. In the former case we say that (u_2, v_2) is *effectively wider* on $[c_1, c_2]$, while in the latter case we say that (u_1, v_1) is effectively wider. Either way, the difference between $E_1 \cap R$ and $E_2 \cap R$ is called the *difference region* of the given

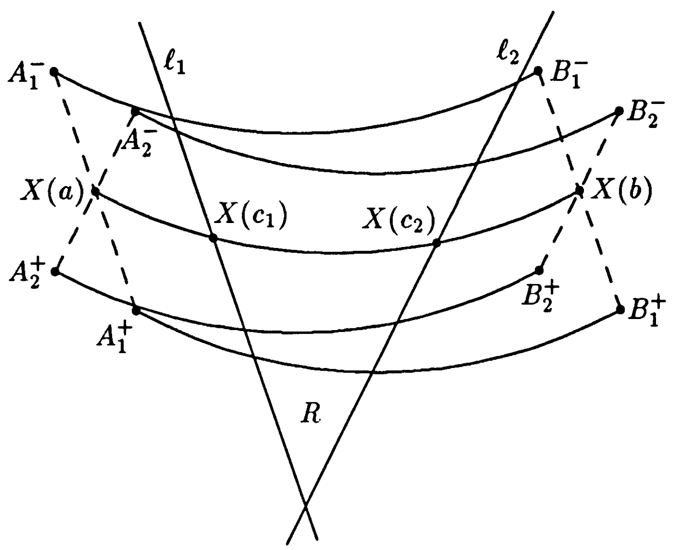

Fig. 24. A simple offset change problem where $(u_1, v_1) = (-1, 3)$ and $(u_2, v_2) = (1, 2)$.

offset change problem for the interval $[c_1, c_2]$. The difference region is composed of two disjoint subregions on either side of the trajectory. These are called the *left and right difference regions*; i.e., the left difference region lies to our left as we face in the trajectory direction $(x'(t), y'(t))$. In Figure 24 the left difference region lies between the curves $A_1^- B_1^-$ and $A_2^- B_2^-$, while the right difference region lies between $A_1^+ B_1^+$ and $A_2^+ B_2^+$.

Lemma 5.3.1. *Let X, (u_1, v_1), and (u_2, v_2) be an offset change problem that is simple for some subinterval $[c_1, c_2]$ of $[a, b]$, the domain of X. Either $E_1 \cap R \subset E_2 \cap R$ or $E_2 \cap R \subset E_1 \cap R$, where R is the simplicity region for $[c_1, c_2]$, and where E_i is the envelope of $\{ X(t) \mid a \le t \le b \}$ with respect to the pen $\{ \alpha \cdot (u_i, v_i) \mid -\frac{1}{2} \le \alpha \le \frac{1}{2} \}$.*

Proof. Choose directions w_1 parallel to ℓ_1 and w_2 parallel to ℓ_2 such that either w_1 and w_2 are positive multiples of each other, or R is the set of all points $I + \alpha_1 w_1 + \alpha_2 w_2$ where I is the intersection point of ℓ_1 and ℓ_2. Since E_1 and E_2 are invariant under negation of (u_1, v_1) and (u_2, v_2), we may assume that each (u_i, v_i) is a negative multiple of w_i. Thus if $z \in R$ and $\alpha_1, \alpha_2 \ge 0$ then $z + \alpha_1(u_1, v_1) + \alpha_2(u_2, v_2) \in R$.

Now let C_i^+ and D_i^+ be the points where $X(t) + \frac{1}{2}(u_i, v_i)$ intersects ℓ_1 and ℓ_2, respectively for $i = 1, 2$; let C_i^- and D_i^- be the similar intersection points for $X(t) - (u_i, v_i)$. The definition of simple offset change problems states that the curves $C_i^+ D_i^+$ and $C_j^+ D_j^+$ and the curves $C_i^- D_i^-$ and $C_j^- D_j^-$ are disjoint for $i \ne j$. Thus it suffices to show that either the segments $C_1^+ C_1^-$ and $D_1^+ D_1^-$ are contained in the segments $C_2^+ C_2^-$ and $D_2^+ D_2^-$, or vice versa.

We can assume without loss of generality that $C_2^+ - C_1^+$ and $D_2^+ - D_1^+$ are positive multiples of w_1 and w_2 respectively, and that $C_1^- - C_2^- = \alpha w_1$ and $D_1^- - D_2^- = \beta w_2$ where α and β are either both positive or both negative. Choose t so that $C_2^+ = X(t) + \frac{1}{2}(u_2, v_2)$, and let $P_1^- = X(t) - \frac{1}{2}(u_1, v_1)$ and $P_2^- = X(c_1) - \frac{1}{2}(u_2, v_2)$. Since $P_1^- - C_2^+ = P_2^- - C_1^+ = -\frac{1}{2}(u_1, v_1) - \frac{1}{2}(u_2, v_2) = \alpha_1 w_1 + \alpha_2 w_2$, where $\alpha_1, \alpha_2 < 0$, it follows that $\{P_1^-, P_2^-\} \subseteq R$ and thus P_i^- is on the curve $C_i^- D_i^-$ for $i = 1, 2$. Clearly $P_1^- - P_2^- = C_2^+ - C_1^+$ is a positive multiple of w_1, and thus P_1^- and P_2^- lie on some line ℓ parallel to ℓ_1. Any parallel line ℓ' between ℓ_1 and ℓ must intersect the curves $C_1^- P_1^-$ and $C_2^- P_2^-$ at least once, and the definition states that these curves are disjoint and each intersect ℓ' at most once. Thus if Q_1^- and Q_2^- are the intersection points of any such line ℓ' with $C_1^- P_1^-$ and $C_2^- P_2^-$, then $Q_1^- - Q_2^-$ is a positive multiple of w_1. In particular, $C_1^- - C_2^-$ has this property and therefore so does $D_1^- - D_2^-$. Thus $C_2^+ C_2^- \subseteq C_1^+ C_1^-$ and $D_2^+ D_2^- \subseteq D_1^+ D_1^-$ as required. (These containments are reversed if we begin by assuming that $C_2^+ - C_1^+$ and $D_2^+ - D_1^+$ are negative multiples of w_1 and w_2.) ∎

For any four values t_1, t_2, t_1', and t_2' in the interval $[a, b]$, the *linear solution* to the above offset change problem is a pair of paths $(\mathcal{T}, \mathcal{T}')$ defined as follows: The path $\mathcal{T}$ contains the curves $X(t) + \frac{1}{2}(u_1, v_1)$ for $a \le t < t_1$, $X(t) + \frac{1}{2}(u_2, v_2)$ for $t_2 \le t < b$, and a simple straight line connecting them. The path $\mathcal{T}'$ contains curves $X(t) - \frac{1}{2}(u_1, v_1)$ for $a \le t < t_1'$, $X(t) - \frac{1}{2}(u_2, v_2)$ for $t_2' \le t < b$, and a similar connecting line.

Given a linear solution to an offset change problem of the above form, the *change envelope* is the region bounded by the closed curve consisting of $\mathcal{T}$, $\mathcal{T}'$, and the line segments whose endpoints are $X(a) \pm \frac{1}{2}(u_1, v_1)$ and $X(b) \pm \frac{1}{2}(u_2, v_2)$. We are primarily interested in cases where this closed curve does not cross itself, but the region bounded by a self intersecting curve can be defined in terms of the concept of *winding number* introduced in the appendix.

A linear solution to an offset change problem is just the pairs of paths obtained by connecting points on appropriately shifted copies of the trajectory by straight lines as shown in Figure 21b. Points A, B, A', and B' in that figure correspond to t_1, t_2, t_1', and t_2'. We shall refer to the shifted copies of the trajectory as the *incoming and outgoing envelope boundaries.*

Linear solutions to offset change problems provide a formal description of the idea of connecting lines. As explained previously, it is desirable to keep the connecting lines short. A convenient way to do this is to simulate the effect of suddenly changing the pen at some point $X(t_0)$ on the trajectory, where $t_0 \in [a, b]$. Take the envelope of the incoming pen with respect to the portion of the trajectory where $a \le t \le t_0$, and use the outgoing pen for $t_0 \le t < b$. The brush shape be thought of as changing smoothly at $X(t_0)$ from the incoming pen to the outgoing pen and covering two triangular regions in the process. The result is shown in Figure 25.

We can construct such a linear solution for the case shown in the figure whenever the offset change problem is simple for some interval $[c_1, c_2]$ and the points $X(t_0) \pm \frac{1}{2}(u_1, v_1)$ and $X(t_0) \pm \frac{1}{2}(u_2, v_2)$ are all in the interior of the simplicity region for $[c_1, c_2]$: If (u_1, v_1) is effectively wider than (u_2, v_2) on $[c_1, c_2]$, then the segment whose endpoints are $X(t_0) \pm \frac{1}{2}(u_1, v_1)$ cuts each of the two curves $X(t) \pm \frac{1}{2}(u_2, v_2)$ exactly once. Let $X(t^+) + \frac{1}{2}(u_2, v_2)$ and $X(t^-) - \frac{1}{2}(u_i, v_i)$ be these two intersection points. If $t^+ > t^-$ then let $(t_1, t_2, t_1', t_2') = (t_0, t^+, t_0, t_0)$; otherwise let $(t_1, t_2, t_1', t_2') = (t_0, t_0, t_0, t^-)$. If (u_2, v_2) is effectively wider, then let $X(t^+) + \frac{1}{2}(u_1, v_1)$ and $X(t^-) - \frac{1}{2}(u_1, v_1)$ be the intersection points of the curves

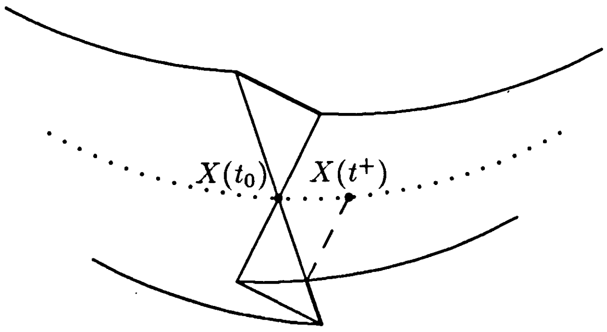

Fig. 25. How a linear solution to an offset change problem can simulate the action of a dynamic brush.

$X(t) \pm \frac{1}{2}(u_2, v_2)$ with the segment whose endpoints are $X(t) \pm \frac{1}{2}(u_2, v_2)$. If $t^+ < t^-$ then let $(t_1, t_2, t_1', t_2') = (t^+, t_0, t_0, t_0)$; otherwise let $(t_1, t_2, t_1', t_2') = (t_0, t_0, t^-, t_0)$.

This solution is called the *standard solution* for the offset change problem at time t_0, and its connecting lines are called the *standard connecting lines* for time t_0. In Figure 25 the standard connecting lines are shown in bold. Theorem 5.3.2 shows how the change envelope for the standard solution relates to the intuitive idea of the brush shape changing at some point $X(t_0)$ on the trajectory.

Theorem 5.3.2. *Let X, (u_1, v_1), and (u_2, v_2) be an offset change problem that is simple for some subinterval $[c_1, c_2]$ of $[a, b]$, the domain of X; let $(\mathcal{T}, \mathcal{T}')$ be the standard solution at some time $t_0 \in [c_1, c_2]$; let ℓ^+ be the line segment whose endpoints are $X(t_0) + \frac{1}{2}(u_i, v_i)$, for $i = 1, 2$; and let ℓ^- be the segment with endpoints $X(t_0) - \frac{1}{2}(u_i, v_i)$. If ℓ^+ and ℓ^- are contained in the difference region for $[c_1, c_2]$, then $\mathcal{T}$ and $\mathcal{T}'$ delimit a subset $S_0 \cup S_1 \cup S_2$ of the simplicity region R for $[c_1, c_2]$, where*

$$\begin{aligned} S_0 &= \{\, X(t_0) + \alpha(1-\beta)(u_1, v_1) + \alpha\beta(u_2, v_2) \mid -\tfrac{1}{2} \le \alpha \le \tfrac{1}{2} \text{ and } 0 \le \beta \le 1 \,\}, \\ S_1 &= \{\, X(t) + \alpha(u_1, v_1) \mid -\tfrac{1}{2} \le \alpha \le \tfrac{1}{2} \text{ and } c_1 \le t \le t_0 \,\}, \\ S_2 &= \{\, X(t) + \alpha(u_2, v_2) \mid -\tfrac{1}{2} \le \alpha \le \tfrac{1}{2} \text{ and } t_0 \le t \le c_2 \,\}. \end{aligned}$$

Proof. Assume that (u_1, v_1) is effectively wider than (u_2, v_2); otherwise we can swap (u_1, v_1) with (u_2, v_2), reverse the parameterization of X, and use the same proof. If t^+ and t^- are as in the definition of the standard solution, we can assume that $t^+ \ge t^-$; otherwise the same proof applies if we negate (u_1, v_1) and (u_2, v_2). These assumptions make the situation correspond to Figure 26.

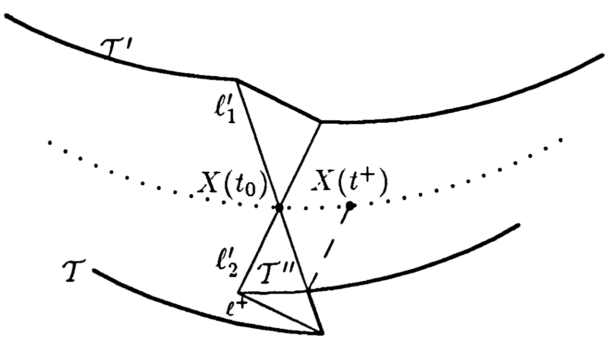

Fig. 26. A construction for Theorem 5.3.2.

We shall show that $B(S_0 \cup S_1 \cup S_2) \subseteq B(R) \cup \mathcal{T} \cup \mathcal{T}'$, where $B(S)$ denotes the boundary of S for any set $S \subseteq \mathbb{R}^2$. Except for a segment ℓ_1' whose endpoints are $X(t^+) + \frac{1}{2}(u_2, v_2)$ and $X(t_0) - \frac{1}{2}(u_1, v_1)$, all points in $B(S_1)$ are contained in $B(R) \cup \mathcal{T} \cup \mathcal{T}'$. The points in $B(S_2)$ not in $B(R) \cup \mathcal{T} \cup \mathcal{T}'$ consist of a segment ℓ_2' with endpoints $X(t_0) \pm \frac{1}{2}(u_2, v_2)$ and the curve $\mathcal{T}''$ given by $X(t) + \frac{1}{2}(u_2, v_2)$ for $t_0 \le t < t^+$. Since $B(S_0) \setminus \big(B(R) \cup \mathcal{T} \cup \mathcal{T}'\big) \subseteq \ell_1' \cup \ell_2' \cup \ell^+$, it is suffices to prove that $\ell_1' \cup \ell_2' \cup \ell^+ \cup \mathcal{T}'' \subseteq I(S_0 \cup S_1 \cup S_2) \cup \mathcal{T} \cup \mathcal{T}'$ where $I(S)$ denotes the interior of the set S.

The definition of simple offset change problems requires that for $i = 1, 2$ and $j = 1, 2$, any curve $\big(x(t), y(t)\big) = X(t) \pm \frac{1}{2}(u_i, v_i)$ intersects every line $v_j x(t) - u_j y(t) = c$ at most once for $\big(x(t), y(t)\big) \pm \frac{1}{2}(u_j, v_j) \in R$. Thus there are functions $f_j(x, y) = \pm(v_j x - u_j y)$ from $\mathbb{R}^2$ to $\mathbb{R}$ such that $f_j\big(x(t), y(t)\big)$ is monotonic increasing whenever $\big(x(t), y(t)\big) \in R$. The definition of offset change problems requires that $v_1 x' - u_1 y'$ and $v_2 x' - u_2 y'$ have the same sign whenever (x', y') is the tangent direction for one of the curves $X(t) \pm \frac{1}{2}(u_i, v_i)$. Thus either $f_1(x, y) = v_1 x - u_1 y$ and $f_2(x, y) = v_2 x - u_2 y$ or $f_1(x, y) = -v_1 x + u_1 y$ and $f_2(x, y) = -v_2 x + u_2 y$. Therefore $f_1(u_2, v_2) = -f_2(u_1, v_1)$.

If E_1 and E_2 are as defined in Lemma 5.3.1 then $z \in I(S_1) \cup \mathcal{T} \cup \mathcal{T}'$ when $f_1\big(X(c_1)\big) < f_1(z) < f_1\big(X(t_0)\big)$ and $z \in E_1$; similarly $z \in I(S_2) \cup \mathcal{T} \cup \mathcal{T}'$ when $f_2\big(X(t_0)\big) < f_2(z) < f_2\big(X(c_2)\big)$ and $z \in E_2$. Since $\mathcal{T}'' \subseteq R$, we know that $f_1(z) < f_1\big(X(t^+) + \frac{1}{2}(u_2, v_2)\big) = f_1\big(X(t_0)\big)$ when $z \in \mathcal{T}''$, and therefore $\mathcal{T}'' \subseteq I(S_1) \cup \mathcal{T} \cup \mathcal{T}'$. A similar argument applies to ℓ^+ and $\ell_2^+ = \{\, X(t_0) + \alpha(u_2, v_2) \mid 0 < \alpha \le \frac{1}{2} \,\}$ if we can show that $f_1(u_2, v_2) < 0$. In this case since $f_2(u_1, v_1) = -f_1(u_2, v_2)$, we can also conclude that $f_2(z) > f_2\big(X(t_0)\big)$ when $z \in \ell_1^+ = \{\, X(t_0) + \alpha(u_1, v_1) \mid 0 < \alpha \le \frac{1}{2} \,\}$. Thus $\ell_1^+ \cap E_2 \subseteq I(S_2) \cup \mathcal{T} \cup \mathcal{T}'$.

If $f_1(u_2, v_2) < 0$, we still need to show that for $i = 1, 2$, the segments $\ell_i^- = \ell_i' \setminus \ell_i^+$ do not intersect $B(S_0 \cup S_1 \cup S_2)$ except on $\mathcal{T} \cup \mathcal{T}'$. Let $z = X(t_0) - \alpha(u_1, v_1)$ be a point on ℓ_1^- where $0 \le \alpha < \frac{1}{2}$. If z' is a point in the immediate neighborhood of z such that $f_1(z') \le f_1\big(X(t_0)\big) = f_1(z)$, then $z' \in S_1$ because $z' = X(t_0 - \epsilon_1) + (\alpha + \epsilon_2)(u_1, v_1)$ for some small ϵ_1 and ϵ_2 where $\epsilon_1 > 0$. If $f_1(z') > f_1\big(X(t_0)\big)$ and $f_2(z') < f_2\big(X(t_0)\big)$, then $z' \in S_0$ because $z' = X(t_0) + \alpha'(1 - \beta')(u_1, v_1) + \alpha'\beta'(u_2, v_2)$ for some $\alpha' \approx \alpha$ and some small positive β'. We can avoid the case $f_2(z') \ge f_2\big(X(t_0)\big)$ unless $f_2(z) \ge f_2\big(X(t_0)\big)$ in which case the assumption $f_2(u_1, v_1) > 0$ implies that $\alpha = 0$ and thus $z = X(t_0)$. In that case $z' = X(t_0 + \epsilon_1) + \epsilon_2(u_2, v_2) \in S_2$ for some small ϵ_1 and ϵ_2, where $\epsilon_1 > 0$.

If $z = X(t_0) - \alpha(u_2, v_2)$ is a point on ℓ_2^- where $0 < \alpha < \frac{1}{2}$, then we can argue as above that any point z' in the neighborhood of z either lies in S_2 or lies in S_0, depending on how $f_2(z')$ compares to $f_2\big(X(t)\big)$.

The above argument shows that $\ell_1 \cup \ell_2 \cup \ell^+ \cup \mathcal{T}'' \subseteq I(S_0 \cup S_1 \cup S_2) \cup \mathcal{T} \cup \mathcal{T}'$ when $f_1(u_2, v_2) = -f_2(u_1, v_1) < 0$. Since $X(t) + \frac{1}{2}(u_2, v_2) \in R$ for t between t_0 and t^+, it follows that $f_1\big(X(t) + \frac{1}{2}(u_2, v_2)\big)$ is monotonic increasing for such t, hence so is $f_1\big(X(t)\big)$. Similarly $f_1\big(X(t)\big)$ is monotonic increasing for t between t^- and t_0 because $X(t) - \frac{1}{2}(u_2, v_2) \in R$ for such t. Thus $f_1(t^+) \ge f_1(t^-)$, and since $f_1\big(X(t^+) + \frac{1}{2}(u_2, v_2)\big) = f_1\big(X(t^-) - \frac{1}{2}(u_2, v_2)\big)$ it follows that $f_1(u_2, v_2) \le 0$.

If $f_1(u_2, v_2) = 0$ then $\mathcal{T}''$ is the empty set and ℓ_1, ℓ_2, ℓ^+ and ℓ^- all lie on some line segment ℓ contained in E_2. Since $\mathcal{T} \cup \mathcal{T}'$ contains $\ell \cap (E_1 \setminus I(E_2))$, we only need to show that $\ell \cap I(E_2) \subseteq I(S_1 \cup S_2 \cup S_3)$. Since the segment with endpoints $\pm\frac{1}{2}(u_1, v_1)$ contains the segment with endpoints $\pm\frac{1}{2}(u_2, v_2)$, we have $E_2 \cap R \subseteq S_1 \cup S_2$. Since the definition of the standard solution requires that $X(t_0) \pm \frac{1}{2}(u_2, v_2) \in I(R)$,

$$\ell \cap I(E_2) \subseteq I(R) \cap I(E_2) = I(R \cap E_2) \subseteq I(S_1 \cup S_2).$$

Thus $\ell_1 \cup \ell_2 \cup \ell^+ \cup \mathcal{T}'' \subseteq I(S_0 \cup S_1 \cup S_2) \cup \mathcal{T} \cup \mathcal{T}'$, and therefore $B(S_0 \cup S_1 \cup S_2) \subseteq B(R) \cup \mathcal{T} \cup \mathcal{T}'$. ∎

The standard solution is useful because it produces reasonably short connecting lines, and the concept can be extended to apply to offset change problems that are not simple: The regions S_0, S_1, and S_2 from Theorem 5.3.2 are well defined whenever $[c_1, c_2]$ is in the range of X. We can always just compute the digitizations of these regions. When the offset angles for the incoming and outgoing envelope boundaries are small, the standard connecting lines are nearly *opposite*; i.e., they both lie near the line perpendicular to the trajectory at $X(t_0)$. In the ideal case of truly opposite connecting lines, both connecting lines lie on the perpendicular through $X(t_0)$.

Restricting our attention to the standard solution removes three degrees of freedom, but this is seldom harmful because the digitization depends only on where the connecting lines fall relative to pixel centers in the left and right difference regions. When the connecting lines are contained in the difference regions as required by Theorem 5.3.2, each difference region is divided into two connected sets, one of which is contained in the change envelope E, and one of which is disjoint to E. The difference regions are usually so thin that the pixel centers inside of them have a linear ordering z_0, z_1, z_2, ..., such that whenever E is the change envelope for a linear solution with nearly opposite connecting lines, E contains exactly those z_i where $0 \le i \le k$ for some k. The choice of the time t_0 at which the standard solution is taken determines this value of k.

We shall now develop formal conditions under which the above idea of linear ordering of changeable pixels can be used to find a good solution to an offset change problem. Let $X = (x(t), y(t))$, (u_1, v_1), and (u_2, v_2) be an offset change problem that is simple for some interval $[c_1, c_2]$, and let R be the corresponding simplicity region. This offset change problem is *positive simple* for $[c_1, c_2]$ if the following hold: 1) It is true that $x'(t) \ge 0$ and $y'(t) \ge 0$ whenever $X(t) \pm \frac{1}{2}(u_i, v_i) \in R$ for $i = 1, 2$. 2) If S is either the left or the right difference region then the digitization $\mathcal{D}(S)$ contains no two pixels $P(m, n)$ and $P(m', n')$ such that $m + n = m' + n'$. 3) If such a digitization $\mathcal{D}(S)$ contains pixels $P(m, n)$ and $P(m+1, n+1)$, then $\mathcal{D}(S)$ contains neither $P(m+1, n)$ nor $P(m, n+1)$.

If the tangent directions do not lie in the first quadrant, then it may be possible to obtain a positive simple offset change problem by rotating the coordinate system. It may be necessary to use Theorem 1.1.1 in order to ensure that the digitization of the rotated change envelope will be a rotated version of the digitization of the original change envelope. If these techniques suffice to construct an equivalent positive simple offset change problem, we say that the original offset change problem is *positive simple under rotation.*

Let X, (u_1, v_1), and (u_2, v_2) be an offset change problem that is positive simple for some interval $[c_1, c_2]$; let Δ and Δ' be the right and left difference regions, respectively; let $(\mathcal{T}, \mathcal{T}')$ be the standard solution at some time $t_0 \in [c_1, c_2]$; let E be the corresponding change envelope; and let $\sigma = 1$ if (u_1, v_1) is effectively wider than (u_2, v_2) and $\sigma = -1$ otherwise. If k is an integer such that $\sigma(m+n) \leq \sigma k$ for all $P(m,n)$ in the digitization $\mathcal{D}(\Delta \cap E)$ and $\sigma(m+n) > \sigma k$ for all $P(m,n) \in \mathcal{D}(\Delta \setminus E)$, then k is a *standard right breakpoint* for X, (u_1, v_1), and (u_2, v_2) at time t_0 with respect to $[c_1, c_2]$. If similar conditions hold when Δ is replaced by Δ', then k is a *standard left breakpoint.* Note that if $t_0 \in [c_1, c_2] \subseteq [c_1', c_2']$ and $X(t_0) \pm \frac{1}{2}(u_i, v_i)$ are interior to the simplicity region for $[c_1, c_2]$, then a standard breakpoint at time t_0 with respect to $[c_1', c_2']$ is necessarily a standard breakpoint with respect to $[c_1, c_2]$, but not vice versa.

Conversely, given an offset change problem as above that is positive simple for some interval $[c_1, c_2]$, any pair of integers (k, k') determine a class of digitally equivalent solutions as follows: Let R, E_1, and E_2 be as defined in Lemma 5.3.1; let i be such that $E_i \cap R \subseteq E_{3-i} \cap R$; let Δ and Δ' respectively be the right and left difference regions; and let $\sigma = \pm 1$ as above. The class of linear solutions is those for which

$$\mathcal{D}(E) = \bigcup_{\substack{\sigma(m+n) \leq \sigma k \\ P(m,n) \subseteq \Delta}} P(m,n) \;\cup \bigcup_{\substack{\sigma(m+n) \leq \sigma k \\ P(m,n) \subseteq \Delta'}} P(m,n) \;\cup\; (E_i \cap R),$$

where $\mathcal{D}(E)$ denotes the digitization of the change envelope.

In general, there may or may not be standard left and right breakpoints at any particular time t_0, but the standard breakpoints usually do exist for offset change problems that appear in practice. If an offset change problem is positive simple with respect to an interval $[c_1, c_2]$, and if for all k there exist times t and t' such that k is the standard left breakpoint at t with respect to $[c_1, c_2]$ and k' is the corresponding standard right breakpoint, then the offset change problem is said to be *infinitely breakable* with respect to $[c_1, c_2]$.

Infinitely breakable offset change problems do not provide complete freedom in the selection of standard breakpoints because the standard left and right breakpoints are not independent of each other. As explained earlier, this is usually not a serious limitation because it is desirable to choose pairs (k, k') that determine classes of solutions with nearly opposite connecting lines. Since the rules that we shall give for finding good solutions to offset change problems are based on choosing breakpoints, our method will also be useful for finding nonstandard solutions if desired. The breakpoint selection method to be defined below depends on the following properties of positive simple offset change problems:

Lemma 5.3.3. *Let X, (u_1, v_1), and (u_2, v_2) be an offset change problem that is positive simple for some interval $[c_1, c_2]$; let ℓ_1 and ℓ_2 be the lines that bound the simplicity region R; and let S be either the left or the right difference region including its boundary. If the digitization $\mathcal{D}(S)$ contains pixels $P(m,n)$ and $P(m',n')$*

such that $m'+n' = m+n+1$ then either $m = m'$, or $n = n'$, or both pixel centers $(m+\frac{1}{2}, n+\frac{1}{2})$ and $(m'+\frac{1}{2}, n'+\frac{1}{2})$ are at a distance of ≤ 1 from both ℓ_1 and ℓ_2.

Proof. Theorem 1.1.1 allows us to assume that there are no pixel centers on the boundary of S. Thus $P(i,j) \subseteq \mathcal{D}(S)$ if and only if $(i+\frac{1}{2}, j+\frac{1}{2}) \in S$. Let (u,v) be a direction such that $vu_1 - uv_1$ and $vu_2 - uv_2$ are both positive or both negative and $u, v > 0$. Let f_1 and f_2 be linear functions such that $f_1(z) = 0$ if and only if z is on ℓ_1, $f_2(z) = 0$ if and only if z is on ℓ_2, and $R = \{ z \in \mathbb{R}^2 \mid f_1(z) \ge 0 \text{ and } f_2(z) \ge 0 \}$. Also let

$$S' = S \cup \{ z - \alpha(u,v) \mid z \in S \cap \ell_1 \text{ and } \alpha \ge 0 \} \\ \cup \{ z + \alpha(u,v) \mid z \in S \cap \ell_2 \text{ and } \alpha \ge 0 \}$$

so that $S = S' \cap R$ and $S' = \{ (x,y) \mid f^-(x) \le y \le f^+(x) \}$, where f^+ and f^- are monotone nondecreasing functions defined on $\mathbb{R}$.

If $m \ne m'$ and $n \ne n'$ then either $m' < m$ or $n' < n$. First consider the case when $m' < m$, and let $(m''+\frac{1}{2}, n''+\frac{1}{2})$ be a pixel center such that $m' \le m'' \le m$ and $n \le n'' \le n'$. (Note that $n' - n = 1 + m - m' \ge 2$.) Since S' contains the pixel centers $(m+\frac{1}{2}, n+\frac{1}{2})$ and $(m'+\frac{1}{2}, n'+\frac{1}{2})$, it follows that

$$f^-(m''+\tfrac{1}{2}) \le f^-(m+\tfrac{1}{2}) \le n+\tfrac{1}{2} \le n''+\tfrac{1}{2} \le n'+\tfrac{1}{2} \le f^+(m'+\tfrac{1}{2}) \le f^+(m''+\tfrac{1}{2}).$$

Thus $(m''+\frac{1}{2}, n''+\frac{1}{2}) \in S'$.

Since $\mathcal{D}(S) = \mathcal{D}(S' \cap R)$ contains no pixels $P(m'', n'') \notin \{P(m,n), P(m',n')\}$ for which $m'' + n'' \in \{m+n, m'+n'\}$, the pixel centers $(m'+\frac{3}{2}, n'-\frac{1}{2})$ and $(m-\frac{1}{2}, n+\frac{3}{2})$ cannot belong to $S' \cap R$. Thus there must be $i, j \in \{1,2\}$ such that

$$\begin{aligned} f_i(m'+\tfrac{1}{2}, n'+\tfrac{1}{2}) &\ge 0, & f_j(m'+\tfrac{1}{2}, n'+\tfrac{1}{2}) &\ge 0, \\ f_i(m+\tfrac{1}{2}, n+\tfrac{1}{2}) &\ge 0, & f_j(m+\tfrac{1}{2}, n+\tfrac{1}{2}) &\ge 0, \\ f_i(m'+\tfrac{3}{2}, n'-\tfrac{1}{2}) &< 0, & f_j(m-\tfrac{1}{2}, n+\tfrac{3}{2}) &< 0. \end{aligned} \tag{5.3.1}$$

Furthermore $i \ne j$ because $(m'+\frac{3}{2}, n'-\frac{1}{2}) - (m'+\frac{1}{2}, n'+\frac{1}{2}) = (m+\frac{1}{2}, n+\frac{1}{2}) - (m-\frac{1}{2}, n+\frac{3}{2})$ and thus $f_k(m'+\frac{3}{2}, n'-\frac{1}{2}) - f_k(m'+\frac{1}{2}, n'+\frac{1}{2}) = f_k(m+\frac{1}{2}, n+\frac{1}{2}) - f_k(m-\frac{1}{2}, n+\frac{3}{2})$ for $k = 1, 2$. In general the linearity of f_1 and f_2 implies that $f_i(x+d, y-d) < f_i(x,y)$ and $f_j(x-d, y+d) < f_j(x,y)$ for $d > 0$. Thus we have $f_i(m+\frac{1}{2}, n+\frac{3}{2}) \le f_i(m'+\frac{3}{2}, n'-\frac{1}{2}) < 0$ and $f_j(m'+\frac{1}{2}, n'-\frac{1}{2}) \le f_j(m-\frac{1}{2}, n+\frac{3}{2}) < 0$. Therefore ℓ_i passes between $(m+\frac{1}{2}, n+\frac{1}{2})$ and $(m+\frac{1}{2}, n+\frac{3}{2})$ and thus within one unit of $(m+\frac{1}{2}, n+\frac{1}{2})$, and similarly ℓ_j must pass within 1 unit of $(m'+\frac{1}{2}, n'+\frac{1}{2})$ in order to exclude $(m'+\frac{1}{2}, n'-\frac{1}{2})$ from R.

It is not possible that $f_i(m'+\frac{3}{2}, n'+\frac{1}{2}) > 0$, because

$$(n'-n)(m'+\tfrac{3}{2}, n'-\tfrac{1}{2}) = (n'-n-2)(m'+\tfrac{1}{2}, n'+\tfrac{1}{2}) \\ + (m'+\tfrac{3}{2}, n'+\tfrac{1}{2}) + (m+\tfrac{1}{2}, n+\tfrac{1}{2})$$

and therefore

$$(n'-n)f_i(m'+\tfrac{3}{2}, n'-\tfrac{1}{2}) = (n'-n-2)f_i(m'+\tfrac{1}{2}, n'+\tfrac{1}{2}) \\ + f_i(m'+\tfrac{3}{2}, n'+\tfrac{1}{2}) + f_i(m+\tfrac{1}{2}, n+\tfrac{1}{2}) \ge 0,$$

contradicting (5.3.1); similarly, it is not possible that $f_j(m-\frac{1}{2}, n+\frac{1}{2}) \geq 0$ because $f_j(m+\frac{1}{2}, n+\frac{1}{2}) \geq 0$, $f_j(m'+\frac{1}{2}, n'+\frac{1}{2}) \geq 0$, and $f_j(m-\frac{1}{2}, n+\frac{3}{2}) < 0$. Thus ℓ_i must pass within 1 unit of $(m'+\frac{1}{2}, n'+\frac{1}{2})$ in order to exclude $(m'+\frac{3}{2}, n'+\frac{1}{2})$, and ℓ_j must pass within 1 unit of $(m+\frac{1}{2}, n+\frac{1}{2})$ in order to exclude $(m-\frac{1}{2}, n+\frac{1}{2})$.

When $n' < n$, the digitization $\mathcal{D}(S')$ contains pixels $P(m''+\frac{1}{2}, n''+\frac{1}{2})$ with $m \leq m'' \leq m'$ and $n' \leq n'' \leq n$. Equation (5.3.1) still holds with $i \neq j$, except the roles of (m, n) and (m', n') are reversed. The linearity of f_1 and f_2 implies that $f_i(m'-\frac{1}{2}, n'+\frac{1}{2}) \leq f_i(m+\frac{3}{2}, n-\frac{1}{2}) < 0$ and $f_j(m+\frac{3}{2}, n+\frac{1}{2}) \leq f_j(m'-\frac{1}{2}, n'+\frac{3}{2}) < 0$, so that the distance between ℓ_i and $(m'+\frac{1}{2}, n'+\frac{1}{2})$ and the distance between ℓ_j and $(m+\frac{1}{2}, n+\frac{1}{2})$ are both at most 1. To complete the proof we argue as above that $f_i(m+\frac{1}{2}, n-\frac{1}{2}) \leq 0$ because

$$\begin{aligned}(m'-m)(m+\tfrac{3}{2}, n-\tfrac{1}{2}) = (m'-m-2)&(m+\tfrac{1}{2}, n+\tfrac{1}{2}) \\ &+ (m+\tfrac{1}{2}, n-\tfrac{1}{2}) + (m'+\tfrac{1}{2}, n'+\tfrac{1}{2}),\end{aligned}$$

and similarly $f_j(m'+\frac{1}{2}, n'+\frac{3}{2}) \leq 0$ because $(m'-\frac{1}{2}, n'+\frac{3}{2})$ is a nonnegative linear combination of $(m'+\frac{1}{2}, n'+\frac{1}{2})$, $(m'+\frac{1}{2}, n'+\frac{3}{2})$, and $(m+\frac{1}{2}, n+\frac{1}{2})$. ∎

Let Δ be the difference region for a simple offset change problem; and let ℓ_1 and ℓ_2 be the lines that bound the simplicity region. If the distance $\|z_1 - z_2\| > 2$ for any points $z_1 \in \Delta \cap \ell_1$ and $z_2 \in \Delta \cap \ell_2$, then the offset change problem is said to be *interesting*. Lemma 5.3.3 shows that the changeable pixels of an interesting positive simple offset change problem can be divided into *right blocks* and *left blocks* such that if B_1 and B_2 are both right blocks or both left blocks, and if $P(m_i, n_i) \in B_i$ for $i = 1, 2$, then the difference between $m_1 + n_1$ and $m_2 + n_2$ is at least 2. (Right blocks lie in the digitization of the right difference region and left blocks come from the left difference region.) In general a *block* is a set of pixels

$$\{\, P(m, n) \mid m_0 \leq m \leq m_1 \text{ and } n_0 \leq n \leq n_1 \,\}$$

where either $m_0 = m_1$ or $n_0 = n_1$. If $m_0 = m_1$ then the block is a *vertical block*; if $n_0 = n_1$ then the block is a *horizontal block*.

A right or left block B is said to be *broken* by a pair of breakpoints (k, k') if B and the complement of B both have nontrivial intersection with the digitization of the change envelope for the class of solutions determined by (k, k'). For any positive simple offset change problem and any pair of breakpoints (k, k'), there is at most one broken right block and at most one broken left block.

Right and left blocks are also referred to as *good* or *bad* depending on how the straightness of the boundaries of the digitized change envelope is affected when such blocks are broken. Consider an interesting positive simple offset change problem X, (u_1, v_1), (u_2, v_2). If the $u_2, v_2)$ is effectively wider and if $u_1 y' - v_1 x'$ and $u_2 y' - v_2 x'$ are positive for trajectory directions (x', y'), then horizontal right blocks and vertical left blocks are bad and vertical right blocks and horizontal left blocks are good. Making (u_1, v_1) effectively wider or making $u_1 y' - v_1 x'$ and $u_2 y' - v_2 x'$ negative reverses the above distinction. Since a block that contains only one pixel is both horizontal and vertical, it is possible for a block to be good and bad simultaneously. This is immaterial because such blocks cannot be broken.

Figure 27 illustrates the effect of broken good blocks and broken bad blocks on the digitization of the change envelope. Let $(\mathcal{T}, \mathcal{T}')$ be a solution to a positive simple offset change problem. When no bad blocks are broken, the digitizations $\mathcal{D}(\mathcal{T})$ and $\mathcal{D}(\mathcal{T}')$ contain only rightward and upward edges. A downward or leftward edge is introduced when a bad block such as the one labelled "A" in the figure is broken. When block A is broken, the downward edge causes $\mathcal{D}(\mathcal{T})$ to have straightness 0. When a good block such as B is broken, the corresponding digitized change envelope boundary tends to appear smoother than it does when there are no broken blocks; i.e., $\mathcal{D}(\mathcal{T}')$ has infinite straightness when B is broken in Figure 27.

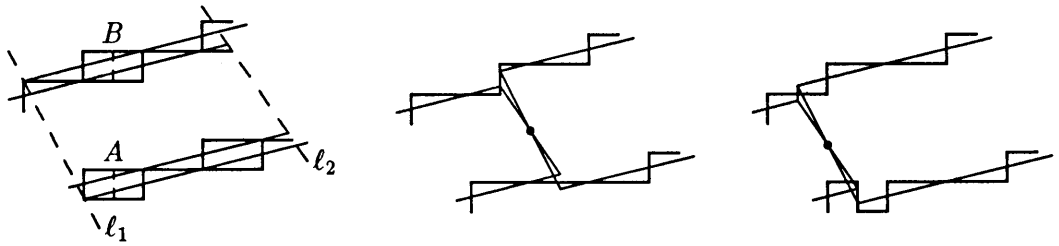

Fig. 27. A positive simple offset change problem and good and bad solutions.

In practice we have an offset change problem $X(t)$, (u_1, v_1) and (u_2, v_2), where we want the change to occur near some point $X(t_0)$. We try to choose some interval $[c_1, c_2]$ for which the problem is simple, where $t_0 \in [c_1, c_2]$. The size of the interval should be chosen so as to reflect the range in which we want the offset change to occur.

Consider a dynamic pen envelope specified by transition times $t_0, t_1, \ldots, t_n$ and offsets $(u_1, v_1), (u_2, v_2), \ldots, (u_n, v_n)$, where the offsets $\pm\frac{1}{2}(u_i, v_i)$ are to be used for $t_{i-1} \le t \le t_i$. We can produce modified transition times $t'_0, t'_1, \ldots, t'_n$ as follows: First let $t'_0 = t_0$, then for $i = 1, 2, \ldots, n-1$, find the largest possible interval I for which $X(t)$, (u_{i-1}, v_{i-1}), (u_i, v_i) is positive simple under rotation and $t_i \in I \subseteq [t'_{i-1}, t_{i+1}]$. If there is such an interval I for which the offset change problem is infinitely breakable, then we choose $t'_i \in I$ as close as possible to t_i so that no bad blocks are broken by the right and left breakpoints at t'_i. If possible, we choose a $t'_i \in I$ such that there are no broken bad blocks but there is a broken good block. In this case, t'_i should be chosen so that the change envelope divides the good block as evenly as possible. When all this is done for each i, we finally set $t'_n = t_n$ to complete the modified dynamic pen specification. The envelope to be digitized is composed of regions like S_0 from Theorem 5.3.2 for each transition time t'_i, together with all points $X(t) + \alpha(u_i, v_i)$ for $-\frac{1}{2} \le \alpha \le \frac{1}{2}$, $t'_{i-1} \le t \le t_i$, and $1 \le i \le n$.

If all the left and right blocks are horizontal, or if all of them are vertical, then all right blocks will be bad and all the left blocks will be good or vice versa. Suppose that only the right blocks are bad. Since infinitely breakable offset change problems provide complete freedom of choice for the standard right breakpoint, it is always possible to avoid broken bad blocks.

In the rare event that it is not possible to find an interval I containing t_i for which the offset change problem is positive simple under rotation, then we might as well just let $t'_i = t_i$. A more likely difficulty is that the offset change problem might not be infinitely breakable or that the simplicity region may be too small to contain the standard connecting lines. Under these circumstances, we can either let $t'_i = t_i$ or choose breakpoints other than the standard ones.

The methods discussed so far for adjusting breakpoints have been based on the idea of broken blocks. When no right or left block contains more than one pixel, the idea of broken blocks provides no useful guidelines on what breakpoints to use. Sometimes this means that the digitized change envelope will appear equally smooth no matter what breakpoints are used, but sometimes the concept of straightness from Section 5.2 indicates a particular choice. In this case, the method for selecting breakpoints is a simple application of a transformation T equal to T_k or U_k for some k. If t_i is the unadjusted transition time, and if the offset change problem $X(t)$, (u_{i-1}, v_{i-1}), (u_i, v_i) is positive simple for some interval I, then we choose T so that $T(X(t))$, $T(u_{i-1}, v_{i-1})$, $T(u_i, v_i)$ is positive simple for some interval J such that $t_i \in J \subseteq I$. We then set t'_i so that the transformed problem has broken good blocks but no broken bad blocks. If the transformed problem also has only single pixel blocks, then we can try to apply another transformation. Very few such transformations will be needed before multiple pixel blocks appear or it becomes impossible to apply further transformations.

Figure 28 shows good and bad solutions to an offset change problem that corresponds to the one shown in Figure 27. Both offset change problems are positive simple and infinitely breakable, but the changeable pixels in Figure 28 all belong to single pixel blocks. Applying T_1 to this problem yields the offset change problem of Figure 27 where good and bad solutions become apparent. Figure 28 illustrates good and bad solutions obtained by using the transition times from the good and bad solutions shown in Figure 27. The digitized envelope boundaries shown in Figure 27 can be obtained by polygonizing the digitized boundaries in Figure 28 and transforming them by T_1. Thus the digitized boundaries from Figure 28 all have straightness one more than the corresponding boundaries in Figure 27. Hence avoiding broken bad blocks in the transformed problem increases the straightness of the digitized boundaries in the corresponding solution to the original problem. The fact that few transformation steps are often needed is merely a consequence of the fact that solutions to offset change problems seldom have large finite straightness.

We have outlined a method for taking a dynamic pen envelope and modifying the transition times so as to produce a new envelope whose digitization has more straightness in the vicinity of the transition points. Our analysis of how to make a transition between integer offsets is based on cases where some transitions seem definitely superior to others. We are forced to make somewhat arbitrary decisions in cases where no particular solution really stands out. For this reason we have concentrated on the basic concepts rather than on implementation details.

There are many ways to choose adjusted transition times t'_1, t'_2, $\ldots$, t'_{n-1} so as to avoid broken bad blocks, and it is not clear what it means to do this "optimally." It may be that a more complex adjustment method will yield better

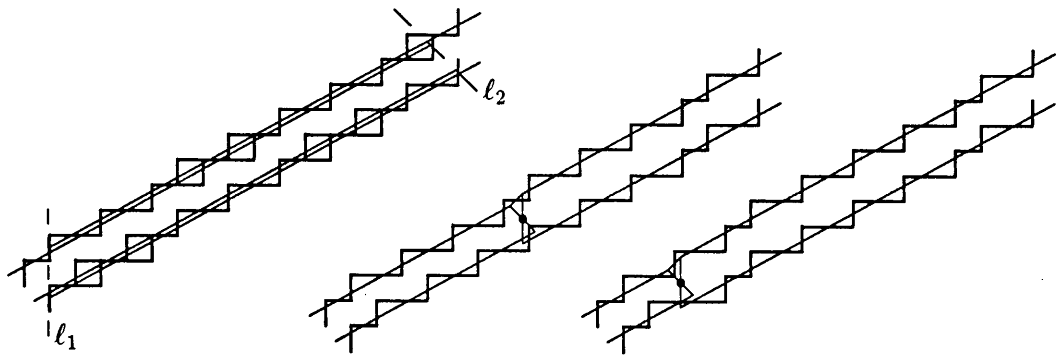

Fig. 28. A positive simple offset change problem and good and bad solutions corresponding to those of Figure 27.

results that the one outlined above. Another possible area for improvement is in choosing breakpoints independently rather than restricting our attention to the standard solution. When choosing breakpoints is not appropriate, it would be necessary to resort to some other idea such as digitizing the regions S_0, S_1, and S_2 from Theorem 5.3.2.

5.4. Choosing Integer Offsets

We have determined conditions under which discrete integer offsets are advantageous in representing a dynamic brush envelope with some width function W, and we have determined how to make smooth joins between portions of the envelope with different offsets. What remains is to determine an appropriate set of integer offsets based the width function W and the trajectory directions.

Let $U(t) = \big(x'(t), y'(t)\big)/\sqrt{x'^2(t) + y'^2(t)}$ be the unit direction vector tangent to the trajectory $\big(x(t), y(t)\big)$. The function U is continuous because the definition of dynamic brush envelopes requires this. Since both W and U are continuous, piecewise real analytic functions of the same parameter t, they determine a continuous, piecewise real analytic curve in a two dimensional space whose points represent (width, direction) pairs. Thus the relevant information about the dynamic brush envelope can be expressed as a parametric curve through this width-direction space. Since the trajectory may have inflection points, this curve is free to double back on itself.

The integer offsets to be chosen determine another, identically parameterized curve through the same width-direction space: Suppose that we choose integer offsets (r_i, s_i) for $1 \le i \le n$ and transition times t_i for $1 \le i < n$. When $t_{i-1} \le t \le t_i$, the idealized width of the dynamic pen envelope will be $\overline{W}(t) = r_i v(t) - s_i u(t)$, where $U(t) = \big(u(t), v(t)\big)$. We want to choose the offsets and transition times so that the width-direction curve $\big(\overline{W}(t), U(t)\big)$ somehow matches $\big(W(t), U(t)\big)$ as closely as possible.

For any fixed integer offset vector, the idealized width $\overline{W}$ depends only on the direction $U(t)$. Furthermore, there is a fixed range of directions (u, v) for which the tangent of the offset angle θ is less than $\gamma/\sqrt{\overline{W}}$ for some constant γ as suggested in Section 5.1. Thus each integer offset vector determines a curve in width-direction space, and $\big(\overline{W}(t), U(t)\big)$ must always lie on such a curve. This locus of allowable

$(\overline{W}(t), U(t))$ depends on the constant γ, but the overall character is always similar to that the example shown in Figure 29. For example if $(r,s) = (0,-1)$, the idealized width decreases from 1 to $1/\sqrt{2} \approx .7$ and the offset angle θ increases from 0° to 45° as the trajectory slope increases from 0 to 1; this leads to the bottom curve in Figure 29. If $(r,s) = (1,-1)$, the idealized width $\overline{W}$ increases from 1 to $\sqrt{2}$ while the offset angle decreases from 45° to 0°; the initial portion of this curve is not shown because the tangent of the offset angle exceeds $1/\sqrt{\overline{W}}$.

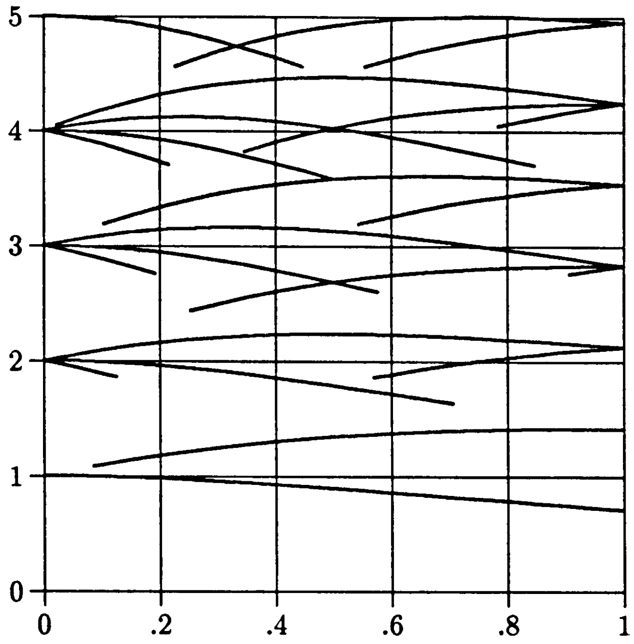

Fig. 29. Idealized width versus trajectory slope for integer offsets (r,s) where the tangent of the offset angle is $\leq (r^2+s^2)^{-1/4}$.

One way to find a set of integer offsets would be to always use the allowable offset vector that makes the idealized width as close as possible to the desired value $W(t)$. That is, take the integer offset vectors sufficiently close to perpendicular to the direction $X'(t)$ and look at their components perpendicular to $X'(t)$. The obvious drawback of this approach is that it produces an excessive number of discontinuities in the idealized width. The direction of every discontinuous change in the idealized width should agree with the sign of the derivative of $W(t)$, and the idealized width should be continuous when $W(t)$ is constant.

If the idealized width is to be continuous then it is possible to change offset vectors only when the widths match, and this can happen only at rational directions. Thus if the idealized width is continuous, there must be rational directions $(u_0,v_0), (u_1,v_1), (u_2,v_2), \ldots, (u_n,v_n)$ and corresponding times $t_0, t_1, \ldots, t_n$ such that each $(-v_i,u_i)$ is a positive multiple of $U(t_i)$ and there is a single integer offset vector (r_i,s_i) for each range $t_{i-1} \leq t \leq t_i$. The following lemma shows that (r_i,s_i) is determined by (u_{i-1},v_{i-1}) and (u_i,v_i) and the idealized widths at t_{i-1} and t_i. Note that the component of the vector (r_i,s_i) in any direction (u_j,v_j) is simply the idealized width $(r_iu_j + s_iv_j)/\sqrt{u_j^2+v_j^2}$.

Lemma 5.4.1. *Let (r,s) be an integer offset vector and let n and n' be integers. If (u,v) and (u',v') are reduced rational directions such that $uv' - vu' = d \neq 0$ and the components of (r,s) in these directions are $n/\sqrt{u^2+v^2}$ and $n'/\sqrt{u'^2+v'^2}$, then*

$$r = \frac{nv' - n'v}{d} \quad \text{and} \quad s = \frac{-nu' + n'u}{d}. \tag{5.4.1}$$

Proof. Since $d \neq 0$, the equations $ru + sv = n$ and $ru' + sv' = n'$ have (5.4.1) as their unique solution. ∎

In Figure 29, the only slopes at which it is possible to change offsets without width discontinuities are 0, $\frac{1}{3}$, $\frac{1}{2}$, and 1. This suggests the following approach: Declare that a reduced rational direction (u,v) is *simple for width w* if $\sqrt{u^2+v^2} \leq f(w)$ for some monotonic nondecreasing function f to be determined later. We always consider two such directions that are simple for $W(t)$ and surround the perpendicular to $X'(t)$. The offset vector chosen is the one whose width in the two simple directions is as close as possible to $W(t)$. It is shown in the appendix that when $W(t)$ is constant, this method for choosing offset vectors is equivalent to using a special generalization of pens.

Figure 30 shows the effective width of relevant offset vectors when the perpendicular to the path direction is between the simple directions $(0,-1)$, $(1,-3)$, and $(1,-2)$. The choice of the offset vector depends on which of the indicated rectangular regions the point $\bigl(U(t), W(t)\bigr)$ falls into. For instance, when the perpendicular direction is between $(0,-1)$ and $(1,-3)$ and $6.8 < W(t) < 8.5$, there are seven possible integer offsets corresponding to the seven rectangular regions shown. The ordering of the rectangular regions is identical to that of the seven width functions shown, but the width functions are not confined to the rectangles. Instead the dividing lines have been carefully placed so that the effective width of the chosen offset vector will be as close as possible to $W(t)$ when $U(t)$ is in the $(1,0)$ direction or in the $(3,1)$ direction.

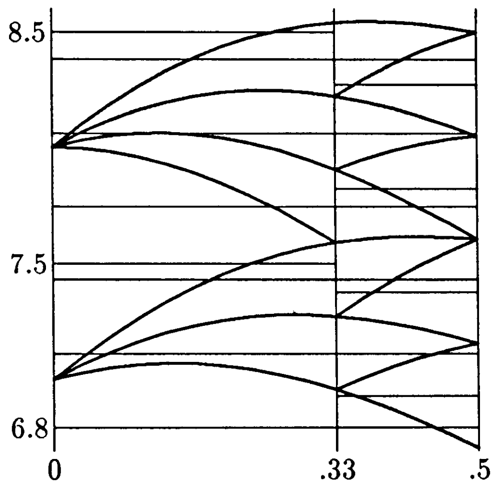

Fig. 30. Width versus slope for offsets between simple directions.

The general rule is that if the simple directions surrounding the perpendicular to the path are (u,v) and (u',v'), then the components of the chosen offset vector in these two directions should be $n/\sqrt{u^2+v^2}$ and $n'/\sqrt{u'^2+v'^2}$ where n and n' are the closest integers to $W(t)\cdot\sqrt{u^2+v^2}$ and $W(t)\cdot\sqrt{u'^2+v'^2}$. Since the simple directions will satisfy (4.1.1), (5.4.1) gives a unique integer offset vector. If the surrounding simple directions are always chosen from some fixed set S as close as possible to the perpendicular to the trajectory, then any discontinuities in the idealized width $\overline{W}(t)$ will be in the direction of $W'(t)$. For example, when $(U(t),W(t))$ is the dotted curve in Figure 31, the corresponding $(U(t),\overline{W}(t))$ curve is as shown in bold in the figure. Discontinuities in $\overline{W}(t)$ come when $(U(t),W(t))$ crosses one of the horizontal lines shown in the figure. The resulting discontinuity in $(U(t),\overline{W}(t))$ is shown as a vertical segment in the bold curve.

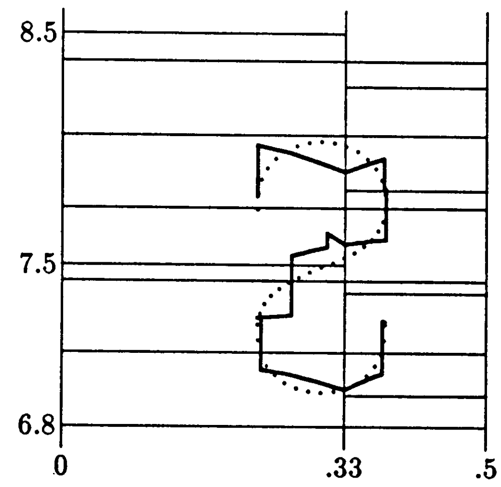

Fig. 31. Width-direction curves for a dynamic brush envelope and the corresponding dynamic pen envelope based on the simple directions of Figure 30.

There might not always be such a fixed set S since new directions can become simple when $W(t)$ increases, and directions can lose their simplicity when the width decreases. In order to keep track of simple directions under such circumstances, we need a way of finding the rational direction (u,v) that is simple for some width w and is as close as possible to some direction (x',y') subject to some limitation on the sign of the difference in angle. The direction (u,v) is called the clockwise or counterclockwise adjacent simple direction to (x',y') for width w, depending on the sign of the difference angle.

Adjacent simple directions may easily be found by using the Stern-Peirce wreath. For instance, to find the counterclockwise adjacent simple direction of a positive quadrant direction (x',y'), start with $(u_0,v_0)=(0,1)$ and treat $(1,1)$ as its left son. To find the simplest direction (u_{i+1},v_{i+1}) between (x',y') and (u_i,v_i), first let (u_{i+1},v_{i+1}) be the left son of (u_i,v_i) and then repeatedly replace (u_{i+1},v_{i+1}) with its right son until it is on the correct side of (x',y'). The counterclockwise adjacent simple direction is the (u_j,v_j) where j is maximized subject to the constraint that (u_j,v_j) must be simple for width w.

The algorithm below finds integer offsets and portions of the trajectory $X(t)=(x(t),y(t))$ where they should apply in order to approximate the width $W(t)$

for $a \le t < b$. The simple directions are (u,v) and (u',v') where

$$l = \sqrt{u^2+v^2} \quad \text{and} \quad l' = \sqrt{u'^2+v'^2}. \tag{5.4.2}$$

The current offset vector is always given by (5.4.1) with $d = 1$, where n and n' are two integers maintained so that $\lfloor l \cdot W(t) + \frac{1}{2} \rfloor = n$ and $\lfloor l' \cdot W(t) + \frac{1}{2} \rfloor = n'$. This always defines an integer offset vector because simple directions can be placed on a Stern-Peirce wreath so that adjacent directions must satisfy (4.1.1).

The algorithm works by successively finding times t when the dynamic brush envelope leaves the current rectangle, i.e., the last t such that the perpendicular to $X'(t)$ is between $(-v,u)$ and $(-v',u')$, and such that

$$\max\left(\frac{n-\frac{1}{2}}{l}, \frac{n'-\frac{1}{2}}{l'}\right) \le W(t) \le \min\left(\frac{n+\frac{1}{2}}{l}, \frac{n'+\frac{1}{2}}{l'}\right). \tag{5.4.3}$$

Algorithm 3 (Choose integer offsets for a dynamic pen envelope).

1) Find (u,v) and (u',v'), the clockwise and counterclockwise adjacent simple directions to $\bigl(y'(t), -x'(t)\bigr)$ for width $W(a)$. Now initialize l and l' according to (5.4.2) and set $n = \lfloor l \cdot W(a) + \frac{1}{2} \rfloor$, $n' = \lfloor l' \cdot W(a) + \frac{1}{2} \rfloor$, and $t = a$.
2) Find the next time $t' > t$ when the dynamic brush envelope leaves the current rectangle and set $t \leftarrow t'$, except set $t \leftarrow b$ if there is no such t'. Use the offset given by (5.4.1) until time t. Now if $t = b$ then stop, otherwise go to one of the following depending on how $\bigl(W(t), X'(t)\bigr)$ left the envelope.
3) Case $X'(t)$ perpendicular to (u',v'): Set $(u,v) \leftarrow (u',v')$, $n \leftarrow n'$, and $l \leftarrow l'$. Let (u',v') be the counterclockwise adjacent simple direction to (u,v), use (5.4.2) to initialize l', set $n' = \lfloor l' \cdot W(t) + \frac{1}{2} \rfloor$, and go to Step 2.
4) Case $X'(t)$ perpendicular to (u,v): Set $(u',v') \leftarrow (u,v)$, $n' \leftarrow n$, and $l' \leftarrow l$. Let (u,v) be the clockwise adjacent simple direction to (u',v'), use (5.4.2) to initialize l, set $n = \lfloor l \cdot W(t) + \frac{1}{2} \rfloor$, and go to Step 2.
5) Case $W(t)$ crosses upper bound in (5.4.3): Set $(u_0,v_0) \leftarrow (u,v)$; set $(u_0',v_0') \leftarrow (u',v')$; and repeat Step 1, using t in place of a. If (u_0,v_0) is between (u,v) and (u',v') then set $(u,v) \leftarrow (u_0,v_0)$, $n \leftarrow \lfloor l \cdot W(t) + \frac{1}{2} \rfloor$, and $l \leftarrow \sqrt{u^2+v^2}$; if (u_0',v_0') is between (u,v) and (u',v') then set $(u',v') \leftarrow (u_0',v_0')$, $n' \leftarrow \lfloor l' \cdot W(t) + \frac{1}{2} \rfloor$, and $l' \leftarrow \sqrt{u'^2+v'^2}$. Now go to Step 2.
6) Case $W(t)$ crosses lower bound in (5.4.3): Repeat Step 1, using t in place of a and set $n = \lceil l \cdot W(t) - \frac{1}{2} \rceil$, and $n' = \lceil l' \cdot W(t) - \frac{1}{2} \rceil$. Then go to Step 2.

Lemma 5.4.2. *The algorithm maintains the invariant that (u,v) is simple for width $(n+\frac{1}{2})/l$, (u',v') is simple for $(n'+\frac{1}{2})/l'$, and no direction between (u,v) and (u',v') is simple for* $\max\bigl((n-\frac{1}{2})/l,\ (n'-\frac{1}{2})/l'\bigr)$

Proof. The initialization in Step 1 ensures that (u,v) and (u',v') are simple for $W(a)$ and no direction between (u,v) and (u',v') is simple for $W(a)$. Hence the monotonicity of $f(w)$ implies that Step 1 preserves the invariant.

If (u,v) is reset at the end of Step 5, then at the beginning of that step (u,v) was simple for width $(n+\frac{1}{2})/l$ but not for width $W(t)$. Thus $W(t) \ne (n+\frac{1}{2})/l$ at the beginning of Step 5, so that the net effect of Step 5 is to preserve the values of both (u,v) and n. Hence (u,v) remains simple for width $(n+\frac{1}{2})/l$ when (u,v) is reset at the end of Step 5. Similarly (u',v') remains simple for width $(n'+\frac{1}{2})/l'$ when it is reset at the end of Step 5. Since the effect of the modifications to (u,v) and (u',v') at the end of Step 5 is to move these directions closer together, any direction between the modified (u,v) and (u',v') will also be between the original (u,v) and (u',v'). Since Step 1 ensures that no direction between the original (u,v) and (u',v') is simple for width $W(t)$, this also hold for the modified directions.

The only other steps that change the relevant variables are 3 and 4, but these ensure that (u,v) and (u',v') are simple for $W(t)$ and no direction between them is. The monotonicity of $f(w)$ again proves the invariant. ∎

Since $(u+u',v+v')$ is not simple for width $w-1$, and since by the triangle inequality $\sqrt{(u+u')^2+(v+v')^2} < l+l'$, it follows that $f(w-1) < l+l'$ where $w>1$ is the width $W(t)$. From (4.1.1), we also have $\sin\phi = 1/(ll')$ where ϕ is the angle between (u,v) and (u',v'). If θ is the offset angle, then

$$\sin\theta \le \max\left(\left|\frac{-rv+su}{(w-1)l}\right|, \left|\frac{rv'-su'}{(w-1)l'}\right|\right)$$

where (r,s) is the offset vector given by (5.4.1). Taking extreme values $n = lw \pm \frac{1}{2}$ and $n' = l'w \pm \frac{1}{2}$ and using $1-\cos\phi = 2\sin^2(\phi/2)$ yields an upper bound

$$\begin{aligned}\sin\theta &\le \frac{(2wll' + \max(l,l'))\cdot\sin^2(\phi/2) + (l+l')/2}{w-1} \\ &= 2ll'\sin^2(\phi/2)(1+O(1/w)) + \frac{l+l'}{2w}(1+O(1/w)). \qquad (5.4.4)\end{aligned}$$

Since $ll' \ge l+l'-1 > f(w-1)-1$, the term $2ll'\sin^2(\phi/2) \approx 1/(2ll')$ is at most $1/(2f(w-1)) + O(f(w-1)^{-2})$. Since $l,l' \le f(w+1)$, the term $(l+l')/(2w)$ is at most $f(w+1)/w$. Thus the tightest bound on the offset angle is obtained when $f(w) \approx \sqrt{w/2}$ so that the offset angle is less than $\sqrt{2/w}+O(1/w)$. A more detailed analysis shows that it is actually better to set $f(w) \approx \sqrt{w}$ and that a bound of $\theta \le 1/\sqrt{w}+O(1/w)$ can be obtained.

Let us take $f(w) = \sqrt{w/\alpha}$ for some constant α so that a reduced rational direction (u,v) is simple for width w if and only if $\alpha(u^2+v^2) \le w$. We shall now find conditions on α that ensure that the idealized width of the offset vector cannot decrease in Step 5 of the algorithm or increase in Step 6. The following lemma gives an important fact about simple directions.

Lemma 5.4.3. *Let (u,v) and (u',v') be reduced rational directions such that $uu'+vv' \ge 0$; let $l^2 = (u+u')^2+(v+v')^2$; and let $\ell^2 = \big((k+1)u+(k'+1)u'\big)^2 + \big((k+1)v+(k'+1)v\big)^2$, for integers $k,k' \ge 0$, not both zero. Then $\ell^2 - l^2 \ge 3$.*

Proof. We immediately have

$$\begin{aligned}\ell^2 - l^2 &= 2(u+u')(ku+k'u') + 2(v+v')(kv+k'v') + (ku+k'u')^2 + (kv+k'v')^2 \\ &= (2k+k^2)(u^2+v^2) + (2k'+k'^2)(u'^2+v'^2) + 2(k+k'+kk')(uu'+vv').\end{aligned}$$

Since $(k,k') \ne (0,0)$, either $2k+k^2 \ge 3$ or $2k'+k'^2 \ge 3$. Since $u^2+v^2 \ge 1$ and $u'^2+v'^2 \ge 1$, the result follows. ∎

The purpose of the next lemma is to show that when the surrounding simple directions (u, v) and (u', v') are updated in Algorithm 3, either the new simple directions are between the old ones or vice versa. The only complicating factors are the interpretation of betweenness in terms of positive linear combinations and the need to treat boundary cases carefully.

Lemma 5.4.4. *Let $\{(u,v), (u',v')\}$ and $\{(u_0,v_0), (u_0',v_0')\}$ be two bases for $\mathbb{R}^2$. If neither (u,v) nor (u',v') is a positive linear combination of (u_0,v_0) and (u_0',v_0'), and if some vector (u_1,v_1) is a nonnegative linear combination of (u,v) and (u',v') and a positive linear combination of (u_0,v_0) and (u_0',v_0'), then (u_0,v_0) and (u_0',v_0') are nonnegative linear combinations of (u,v) and (u',v').*

Proof. We know that $(u_1,v_1) = \alpha(u,v) + \alpha'(u',v')$ for some $\alpha, \alpha' \geq 0$. Thus $vu_1 - uv_1 = v(\alpha u + \alpha' u') - u(\alpha v + \alpha' v') = \alpha'(vu' - uv')$ and $v'u_1 - u'v_1 = v'(\alpha u + \alpha' u') - u'(\alpha v + \alpha' v') = \alpha(uv' - vu')$ have opposite signs in the sense that either one is nonnegative and the other is nonpositive, or vice versa. Similarly $v_0u_1 - u_0v_1$ and $v_0'u_1 - u_0'v_1$ have opposite signs. Since the given information is invariant when (u,v) and (u',v') are swapped or when (u_0,v_0) and (u_0',v_0') are swapped, we can assume without loss of generality that

$$vu_1 - uv_1 \geq 0 \quad \text{and} \quad v_0u_1 - u_0v_1 \geq 0. \tag{5.4.5}$$

It follows that $v'u_1 - u'v_1 \leq 0$ and $v_0'u_1 - u_0'v_1 \leq 0$.

Since (u_1,v_1) is a positive linear combination of (u_0,v_0) and (u_0',v_0'), it is nonzero and it is not a multiple of (u_0,v_0). Thus there are unique real constants a and b such that $(u,v) = a(u_0,v_0) + b(u_1,v_1)$. If $a = 0$ then $b \neq 0$ and $(u_1,v_1) = (u,v)/b$. In addition $b > 0$ because (u_1,v_1) is a nonnegative linear combination of (u,v) and (u',v'). This contradicts the given information that (u_1,v_1) is a positive linear combination of (u_0,v_0) and (u_0',v_0') but (u,v) is not. Thus $a \neq 0$, and therefore $(u_0,v_0) = c(u,v) + d(u_1,v_1)$ if and only if $c = 1/a$ and $d = -b/a$.

Since $a(u_0,v_0)$ and $c(u,v)$ are nonzero, equality cannot hold in either part of (5.4.5). Thus from

$$0 < vu_1 - uv_1 = (av_0 + bv_1)u_1 - (au_0 + bv_1)v_1 = a(v_0u_1 - u_0v_1)$$

we can conclude that $a > 0$ and therefore that $c > 0$. It follows that $b \leq 0$ since otherwise (u,v) is a positive linear combination of (u_0,v_0) and (u_1,v_1) and therefore also of (u_0,v_0) and (u_0',v_0'), contradicting the given information. Since $d \geq 0$ when $b \leq 0$, (u_0,v_0) is a positive linear combination of (u,v) and (u_1,v_1) and therefore a nonnegative linear combination of (u,v) and (u',v').

A similar argument shows that (u_0',v_0') is a nonnegative linear combination of (u,v) and (u',v'): We obtain the equations

$$\begin{aligned} 0 &> v'u_1 - u'v_1, & 0 &> v_0'u_1 - u_0'v_1, \\ (u',v') &= a(u_0',v_0') + b(u_1,v_1), & (u_0',v_0') &= c(u',v') + d(u_1,v_1), \end{aligned}$$

where $a, c \neq 0$ and b and d are negative multiples of each other. Then $0 > (av_0' + bv_1)u_1 - (au_0' + bu_1)v_1 = a(v_0'u_1 - u_0'v_1)$ so that $a > 0$ and therefore $c > 0$. As above (u_0',v_0') is a nonnegative linear combination of (u,v) and (u',v'). ∎

Consider the state of execution when Step 5 is about to start, and $W(t) = (n+\frac{1}{2})/l \le (n'+\frac{1}{2})/l'$ or $W(t) = (n'+\frac{1}{2})/l' \le (n+\frac{1}{2})/l$. Step 5 ensures that at the end of the step (u,v) and (u',v') can both be expressed as $k(u_0,v_0)+k'(u_0',v_0')$ where $k,k' \ge 0$. Let $\ell^2 = (u+u')^2+(v+v')^2$; let

$$S_0 = \{(u_0,v_0),\,(u_0',v_0'),\,(u_0+u_0',v_0+v_0')\};$$

and suppose that $(u,v) \notin S_0$ at the end of Step 5. Applying the invariant of Lemma 5.4.2 at the beginning of Step 5 shows that $\alpha l^2 > W(t)-1$. By Lemma 5.4.3, we also have $\alpha\ell^2 > W(t)-1$. The directions (u,v) and (u',v') are simple for width $W(t)$, and therefore αl^2 and $\alpha\ell^2$ must both be between $W(t)$ and $W(t)-1$ so that $\alpha\left|l^2-\ell^2\right| < 1$. If we choose $\alpha \ge \frac{1}{3}$ then Lemma 5.4.3 contradicts the assumption that $(u,v) \notin S_0$, and a similar argument shows that $(u',v') \in S_0$ at the end of Step 5.

Let $n_0 = n$, $(u_0,v_0) = (u,v)$, and $(u_0',v_0') = (u',v')$ before Step 6. Then Lemma 5.4.2 shows that no direction between (u_0,v_0) and (u_0',v_0') can be simple for width $W(t)$; hence letting $(u_1,v_1) = \big(x'(t),-y'(t)\big)$ in Lemma 5.4.4, we can conclude that (u_0,v_0) and (u_0',v_0') are between (u,v) and (u',v').

Let $l_0^2 = u_0^2+v_0^2$; let $\ell_0^2 = (u_0+u_0')^2+(v_0+v_0')^2$; and let

$$S_0' = \{(u,v),\,(u',v'),\,(u+u',v+v')\}.$$

If $(u_0,v_0) \notin S_0'$, then αl_0^2 and $\alpha\ell_0^2$ are both greater than $W(t)$ because no direction between (u,v) and (u',v') can be simple for width $W(t)$. Since (u_0,v_0) is simple for some width $(n_0+\frac{1}{2})/l_0 \le W(t)+1$, and since Lemma 5.4.3 implies that $\ell_0^2 < l_0^2$, it follows that both αl_0^2 and $\alpha\ell_0^2$ are between $W(t)$ and $W(t)+1$. As above, Lemma 5.4.3 shows that our assumption must be false when $\alpha \ge \frac{1}{3}$. Thus $(u_0,v_0) \in S_0'$ after Step 6, and by a similar argument $(u_0',v_0') \in S_0'$.

We have shown that if $\alpha \ge \frac{1}{3}$ then the only new simple direction that can be introduced in Step 5 is $(u,v)+(u',v')$, and any simple direction removed in Step 6 can be expressed as $(u,v)+(u',v')$ in terms of the new simple directions. If no new simple direction is added in Step 5 or if no old simple direction is deleted in Step 6 then n and n' can only increase in Step 5 and they can only decrease in Step 6. Since $\big(y'(t),-x'(t)\big) = a(u,v)+a'(u',v')$ for some $a,a' > 0$, the idealized width

$$\frac{an+a'n'}{\sqrt{(au+a'u')^2+(av+a'v')^2}}$$

of the offset vector determined by (5.4.1) at time t is a monotonic increasing function of n and n'. Thus any discontinuity in the idealized width at time t is in the direction of $W'(t)$.

We now consider discontinuities in the idealized width when (u,v) and (u',v') are updated in Steps 5 and 6. The following lemma simplifies the task of comparing the idealized width of the offset vector determined by (5.4.1) before and after such steps.

Lemma 5.4.5. *Let (r,s) and $(\bar{r},\bar{s})$ be integer offset vectors; let (u,v) and (u',v') be reduced rational directions; and let $n = ur + vs$, $n' = u'r + v's$, and $\bar{n}'' = u''\bar{r} + v''\bar{s}$ where $(u'',v'') = (u,v) + (u',v')$. If $\bar{n}'' \geq n + n'$ and $u\bar{r} + v\bar{s} \geq n$ then in any direction between (u,v) and (u'',v''), the idealized width of $(\bar{r},\bar{s})$ is no less than the idealized width of (r,s); if $\bar{n}'' \geq n + n'$ and $u'\bar{r} + v'\bar{s} \geq n$ then the same holds for directions between (u',v') and (u'',v'').*

Proof. Since $u''r + v''s = n + n'$, the idealized widths of (r,s) and $(\bar{r},\bar{s})$ in a direction $(\bar{u},\bar{v}) = a(u,v) + b(u'',v'')$ are

$$\frac{\bar{u}r + \bar{v}s}{\sqrt{\bar{u}^2 + \bar{v}^2}} = \frac{an + b(n+n')}{\sqrt{\bar{u}^2 + \bar{v}^2}} \quad \text{and} \quad \frac{\bar{u}\bar{r} + \bar{v}\bar{s}}{\sqrt{\bar{u}^2 + \bar{v}^2}} = \frac{a(u\bar{r} + v\bar{s}) + b\bar{n}''}{\sqrt{\bar{u}^2 + \bar{v}^2}}.$$

When $\bar{n}'' \geq n + n'$ and $u\bar{r} + v\bar{s} \geq n$, the latter idealized width is larger as required. For directions $(\bar{u},\bar{v}) = a(u,v) + b(u'',v'')$, the proof is exactly the same except the roles of (u,v) and (u',v') are reversed. ∎

Consider the state of execution on entering Step 5, and let $(u'',v'') = (u,v) + (u',v')$ be the new simple direction that is introduced in that step. If $l''^2 = u''^2 + v''^2$, then the effect of Step 5 is to choose an integer n'' such that $(n'' - \frac{1}{2})/l'' \leq W(t) < (n'' + \frac{1}{2})/l''$ and either to set $(u,v,n) \leftarrow (u'',v'',n'')$ or to set $(u',v',n') \leftarrow (u'',v'',n'')$. Either way, the following lemma shows that

$$(n + n' - \tfrac{1}{2})/l'' \leq \min((n + \tfrac{1}{2})/l,\, (n' + \tfrac{1}{2})/l').$$

Thus $n'' \geq n + n'$, and Lemma 5.4.5 shows that the idealized width of the new offset vector is at least that of the old.

Lemma 5.4.6. *Let (u,v) and (u',v') be rational directions such that $uu'+vv' \geq 0$ and $|uv' - vu'| = 1$. If (u,v) and (u',v') satisfy (5.4.2) and are simple for widths $(n+\frac{1}{2})/l$ and $(n'+\frac{1}{2})/l'$ respectively, where $(n'-\frac{1}{2})/l' \leq w = (n+\frac{1}{2})/l \leq (n'+\frac{1}{2})/l'$, and if $(u'',v'') = (u,v) + (u',v')$ has length l'' and is not simple for $(n - \frac{1}{2})/l$ or for $(n' - \frac{1}{2})/l'$, then $(n + n' - \frac{1}{2})/l'' \leq w$ if $\alpha < \frac{9}{10}$.*

Proof. The condition that (u'',v'') not be simple for $(n - \frac{1}{2})/l$ is equivalent to $\alpha l''^2 > w - 1/l$. Since $n = wl - \frac{1}{2}$ and $n' \leq wl' + \frac{1}{2}$, it is sufficient to prove that $wl + wl' - \frac{1}{2} \leq wl''$ or that

$$(\alpha l''^2 + 1/l)(l + l' - l'') \leq \tfrac{1}{2}.$$

By the law of cosines

$$l''^2 = l^2 + l'^2 + 2ll'\cos\phi = (l + l')^2 - 4ll'\sin^2(\phi/2)$$

where ϕ is the angle between (u,v) and (u',v'). Since $ll'\sin\phi = 1$ and $\cos^2(\phi/2) \geq \frac{1}{2}$, it follows that

$$(l + l')^2 - l''^2 = \frac{ll'\sin^2\phi}{\cos^2(\phi/2)} = \frac{\sin\phi}{1 - \sin^2(\phi/2)} < \frac{\sin\phi}{1 - \frac{1}{2}\sin^2\phi} = \frac{1}{ll'(1 - \frac{1}{2}(ll')^{-2})},$$

hence $l+l'-l'' < ((l+l')^2 - l''^2)/2l'' < 1/(2ll'l''(1-\frac{1}{2}(ll')^{-2}))$. Since $l'' < l+l'$ and $1/l'' < 1/l'$, we also have $\alpha l'' + 1/(ll'') < \alpha(l+l') + 1/(ll')$. Therefore

$$\left(\alpha l'' + \frac{1}{ll''}\right) l''(l+l'-l'') < \left(\alpha(l+l') + \frac{1}{ll'}\right) l''(l+l'-l'') < \frac{\alpha(l+l')+1/(ll')}{2ll'-1/(ll')}$$

and it suffices to show that

$$\alpha(l+l') \le ll' - \frac{3}{2ll'}. \tag{5.4.6}$$

It is not hard to check that (5.4.6) is satisfied when $l = 1$ and $l' \ge \sqrt{122}$, when $l = \sqrt{2}$ and $l' \ge \sqrt{13}$, or when l and l' are both $\ge \sqrt{5}$. Since the equation is symmetric in l and l', this leaves a finite number of cases to check. For each such case it is a simple matter to check all pairs n, n' for which $(n'-\frac{1}{2})/l' \le (n+\frac{1}{2})/l \le (n'+\frac{1}{2})/l'$ and

$$\max\left(\frac{n-\frac{1}{2}}{l}, \frac{n'-\frac{1}{2}}{l'}\right) < \frac{9}{10}l''^2.$$

(The first case that fails when $\alpha \ge \frac{9}{10}$ is $l = \sqrt{2}$, $l' = 1$, $n = 6$, and $n' = 5$.) ∎

Lemma 5.4.6 can also be used to show that the idealized width in direction $(y'(t), -x'(t))$ does not increase when (u,v) and (u',v') are updated in Step 6. Let the directions (u,v) and (u',v') in the lemma be the updated directions from Step 6, and use $(n,n') = (\bar{n}-1, \bar{n}')$ where $\bar{n}$ and $\bar{n}'$ are the values of the the variables n and n' after Step 6 of the algorithm. The lemma shows that

$$(n+n'-\tfrac{1}{2})/l'' \le (n+\tfrac{1}{2})/l = \min((n+\tfrac{1}{2})/l, (n'+\tfrac{1}{2})/l')$$

and therefore

$$(\bar{n}+\bar{n}'-\tfrac{3}{2})/l'' \le (\bar{n}-\tfrac{1}{2})/l = \max((\bar{n}-\tfrac{1}{2})/l, (\bar{n}'-\tfrac{1}{2})/l').$$

Thus if for some integer n'', $W(t) = (\bar{n}''-\frac{1}{2})/l''$ then $\bar{n}'' > \bar{n}+\bar{n}'-1$ and Lemma 5.4.5 shows that the execution of Step 6 cannot change the offset vector determined by (5.4.1) in such a way as to increase the idealized width.

If $W(t) = (n-\frac{1}{2})/l$ before Step 6 and (u,v) is not changed during that step, then $W(t) = (\bar{n}+\frac{1}{2})$ after Step 6, where $\bar{n}$ is the new n. Lemma 5.4.6 shows that if for some integer n'', $(n''-\frac{1}{2})/l'' \le W(t) < (n''+\frac{1}{2})/l''$ then $n'' \ge n+n'$. As before, Lemma 5.4.6 shows that the idealized width does not increase. We have proved the following theorem.

Theorem 5.4.7. *If $f(w) = \sqrt{w/\alpha}$ where $\frac{1}{3} \le \alpha < \frac{9}{10}$, then any discontinuities in the idealized width $\overline{W}(t)$ determined by Algorithm 3 are in the direction of $W'(t)$.* ∎

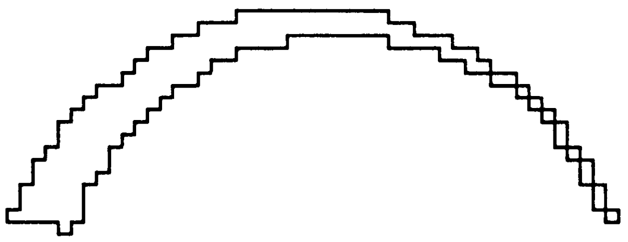

Fig. 32. A digitized envelope resulting from Algorithm 3.

Figure 32 shows an example of the results of the algorithm with offset transitions smoothed as outlined in Section 5.3. Of course much of the work is in computing the digitizations of the envelope boundaries. Part of the idealized width function for this example is shown in Figure 33. Since $W(t)$ is monotone increasing, all the discontinuous changes in idealized width are in the positive direction.

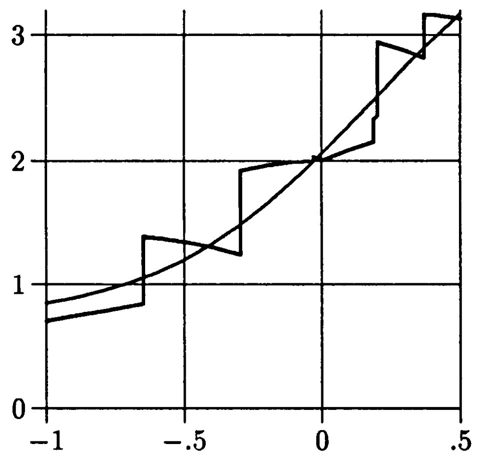

Fig. 33. Idealized width in bold and $W(t)$ versus trajectory slope for Figure 32.

We have seen that Algorithm 3 produces no width discontinuities that disagree with the sign of $W'(t)$ when $\frac{1}{3} \le \alpha < \frac{9}{10}$. When $\alpha \ge .76$, it follows from (5.4.4) that the offset angle is $< 1.01/\sqrt{w} + O(1/w)$. This gives $\gamma \approx 1$ in (5.1.1), a result roughly comparable to what we found for circular brush shapes in Algorithm 1.

The running time of Algorithm 3 is proportional to the number of different offsets computed plus whatever work is required for the actual plotting. If (5.1.2) is satisfied then the actual plotting time should be at least comparable to the other overhead.

The work required to compute adjacent simple directions can be reduced in several ways. One option would be to keep track of Stern-Peirce tree ancestors of (u', v') as given by Lemma 4.3.2, and use a similar trick for the clockwise ancestors of (u, v). With this modification, Steps 1 and 3 through 6 all run in constant time.

Chapter 6

Conclusion and Applications

We have developed a mathematical model for the apparent width of digitized brush envelopes, and we have defined a special class of brush shapes that produce relatively uniform and predictable apparent weight. Algorithm 1 constructs such pens from any symmetric convex brush outline, and Algorithm 2 does the same for asymmetric convex brush outlines.

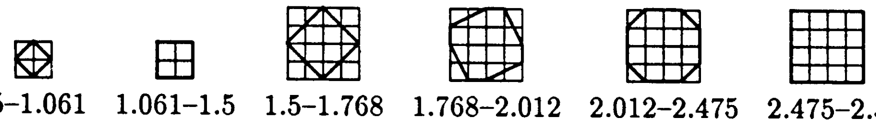

Fig. 34. Pens generated from circles of indicated diameters centered on the origin. (Shown with half integer grids.)

Figure 34 shows the results of Algorithm 1 for some of the cases of greatest practical importance: small circular brushes. To draw curved lines of constant thickness d, we apply Algorithm 1 to a circle of diameter d centered on the origin and then digitize envelopes with respect to the resulting pen.

Algorithm 1 does not have any built-in concept of reflective symmetry, so for circles of diameter 2 it prefers the asymmetrical shape shown to any symmetrical shape whose width is less accurate in certain directions. Figure 35 shows how such asymmetry can be avoided if the algorithm is modified to construct only one octant of the pen polygon and then use symmetry to obtain the rest. Note that a circle of diameter 2 now produces a symmetrical octagonal shape. This is avoided by the unmodified version of Algorithm 1 because of the width in directions $(1, -2)$ and $(2, 1)$.

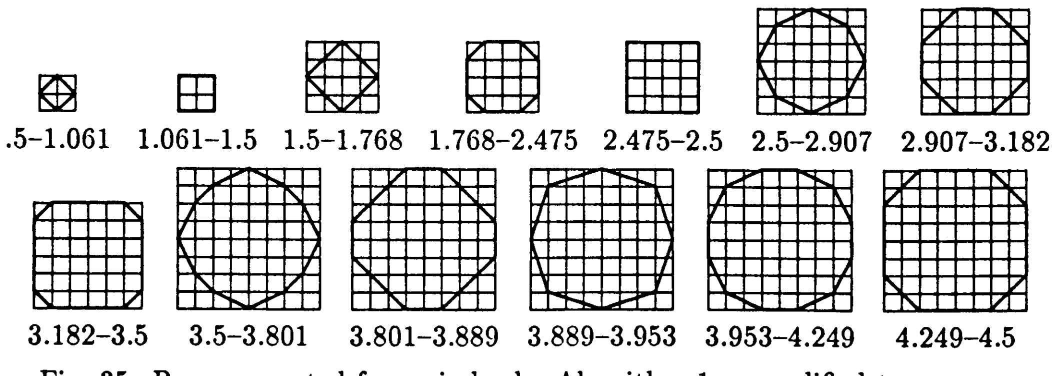

Fig. 35. Pens generated from circles by Algorithm 1 as modified to ensure 8-fold symmetry.

The reason that some of the pens shown in Figure 35 are so far from circular is that an attempt has been made to achieve the optimum width in different rational directions independently. Thus for $d = 2.49$ the width in direction $(0, -1)$ has been rounded down to 2 but the width in direction $(1, -1)$ has been rounded up to $4/\sqrt{2} \approx 2.82$. It would probably be possible to achieve more rounded shapes at the cost of more error in width, but Lemma 5.1 and Corollary A.2.3 show that the potential benefit is not large. One possibility would be to use only circles of integer diameter.

6.1. Comparison of Methods for Finding Digitized Envelopes

We developed pen polygons in order to produce envelopes with integer offset vectors. However, not all integer offset vectors are equally desirable, and in Chapter 5 this led us to consider more direct ways to get integer offsets. The direct approach turned out to be much more flexible, and we saw that it achieves offset angles that are only possible with nearly circular pens; however as we saw in Section 5.1, this direct approach is not always appropriate. In some cases, best results are obtained by finding a mathematical description for the desired envelope and just taking its digitization.

In Section 5.1 we considered the idea of using dynamic pen envelopes to simulate envelopes with respect to non-circular brushes. As is demonstrated in Figure 18, these two types of envelopes can produce very different results even without the effects of digitization. We have made it clear that dynamic pen envelopes should not be thought of as a poor approximation to ordinary brush envelopes, but rather as discrete versions of an important independent class of objects called dynamic brush envelopes. In spite of this, it is still useful for purposes of comparison to consider dynamic brush envelopes where the width function is chosen to match the directional width of a particular brush shape.

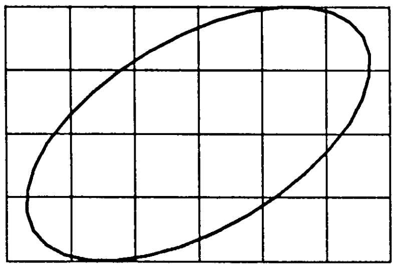
Fig. 36a. An elliptical brush.

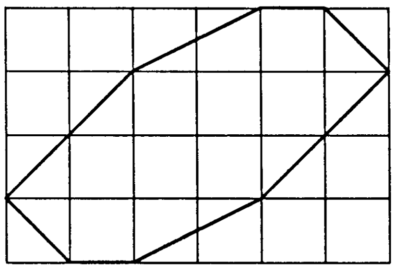
Fig. 36b. The corresponding pen from Algorithm 1.

For example, consider the ellipse and the pen generated from it by Algorithm 1 as shown in Figure 36. This pen can generate offset angles as large as 63°, but its digitized envelopes usually do not have seriously uneven weight. A fairly typical example is shown in Figure 37a. Compare this with Figure 37b which was generated by applying Algorithm 3 to a dynamic brush envelope whose width function is based on the ellipse of Figure 36a. The apparent weight of this envelope has fewer nonmonotonicities, and it is a little heavier in spots, but the differences are subtle. As indicated in Section 5.3, somewhat better results could probably be obtained with a more sophisticated smoothing strategy.

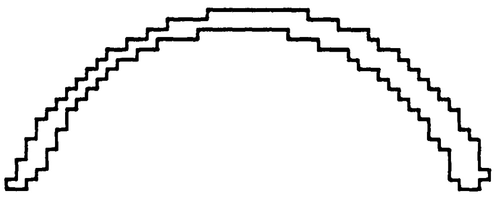
Fig. 37a. An envelope with respect to the pen of Figure 36b.

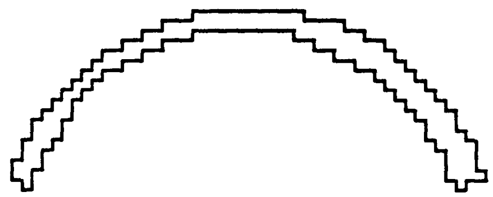
Fig. 37b. A similar envelope built directly from integer offset vectors.

For nearly circular brush shapes, the added complexity of using Algorithm 3 and repositioning transition points is probably not worthwhile. The process of selecting integer offsets is somewhat hard to control, and by generating a pen we can essentially choose a good set of offset vectors in advance. The displacements between opposite pen vertices will be the offset vectors, and the changes between offsets are made when the trajectory direction is parallel to the pen edges.

When the curvature of the trajectory is low enough that $\frac{dW}{ds} < W^{-3/2}$, dynamic pen envelopes usually do produce better results than noncircular brush envelopes. When the width changes much faster than this, it is sometimes best just to find curves that bound the ideal envelope and digitize these without worrying about offset vectors. Integer offset vectors are of little value when no single offset can apply to a portion of the envelope large enough to make (2.2.6) small. In extreme cases it becomes ridiculous to consider the width of an envelope when the intuitive concepts of "left side" and "right side" break down.

Since $\frac{dW}{ds}$ is independent of scaling but W is not, all the conditions that we have given for the usefulness of dynamic pen envelopes will eventually fail when such scaling is done. This algorithm is therefore limited to moderate resolutions or small width variations. However it is superior when W and $\frac{dW}{ds}$ are similar to what they are in Figure 37, and such cases come up often in practice. There is a moderate amount of overhead involved in finding and repositioning transition points, but when (5.1.2) holds this is not much more than what is required to digitize the trajectory. Better results may be achievable by using algorithms more complex than Algorithm 3 to select the integer offsets.

The behavior of pen envelopes under scaling is somewhat different. The benefits of integer offset vectors do become less important when the arc length over which each applies is less than the stroke width, but offset portions of the trajectory continue to dominate the envelope boundary regardless of scaling. Suppose one of the pen construction algorithms is being applied to a fixed brush shape and the brush and trajectory are both scaled uniformly. The only parts of the envelope tracing that are not offset portions of the trajectory is just proportional to the perimeter of the pen and this remains a fixed fraction of the total envelope.

Offset vectors are beneficial only when (2.2.6) is less than 1, and for noncircular brush shapes this requires arc lengths proportional to the stroke width. This arc length is proportional to the scale factor times the exterior angle at the pen vertex associated with the offset. That is, the arc length will be at most $O(w^{1/2})$, and will be more like $w^{1/3}$ on the average if the pen has $w^{2/3}$ vertices. Only if the brush is circular will the benefits of integer offset vectors be independent of scaling. Of course any pen will produce beneficial integer offsets if the trajectory is straight enough. This is exactly the reason why we first considered infinite straight line trajectories.

Another reason why pens continue to remain useful at high resolutions is that they simplify the process of computing digitizations. As explained in the introduction, shifted portions of the trajectory are usually much easier to digitize than the envelope of an arbitrary brush. The overhead involved in manipulating pen edges is sublinear in the resolution, and the overhead due to having to digitize curves that are not part of the envelope boundary remains a (usually small) constant factor of the total work. Thus some of the limitations of dynamic pens do apply to fixed pens, but there is no need to switch to some other algorithm at high resolutions.

6.2. Smoothing the Digitized Trajectory

Until now we have not placed much emphasis on the smoothness of the digitized envelope boundaries, except for the discussion of changing offsets in Section 5.3. This is not because the smoothness of digitized boundaries is unimportant, but rather because the smoothness problem is essentially independent of properties of pens such as uniformity of width. With integer offsets, both sides of the envelope have same shape, so it is possible to adjust the trajectory so as to smooth out both boundaries.

The basic idea of smoothing the trajectory is to avoid pimples and flat spots in the digitization such as those shown in Figure 38. With integer offset vectors, both sides of the digitized envelope can be made smooth by controlling either one of them.

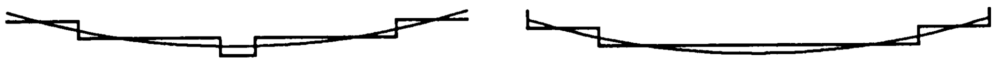
Fig. 38. Curves whose digitizations have pimples and flat spots.

Figure 38 should make both the problem and the solution clear. The indicated curves have horizontal tangent points with y-coordinates that are almost half way between pixel edges. Since the y-coordinates of the digitization are rounded versions of y-coordinates of the partial tracing being digitized, the unusual positioning relative to pixel edges causes the lowest edge of the digitization to be .extraordinarily long or short. The solution is to adjust the trajectory so that when the appropriate offset is added, horizontal and vertical tangent points are achieved only at integer coordinates.

We shall now use the ideas of Section 5.2 to see how pimples and flat spots can occur at any rational slope. Let $\mathcal{T}$ be a path obtained by taking a portion of the trajectory and shifting it so that it lies on the envelope boundary.

Suppose that $\mathcal{T}$ is confined to the positive quadrant, and no two adjacent edges both have length greater than 1. When a transformation T_k or U_k is applied to the polygonization of $\mathcal{T}$, a section of it will be mapped into a simple positive digital tracing as shown in Figure 39. If the segment at the boundary of such a section is particularly long or short, then the polygonization may be said to have a flat spot or a pimple. The polygonization shown in Figure 39 has two such sections: one which becomes a digital tracing when T_2 is applied, and another that requires T_1. A single segment of slope $\frac{1}{2}$ is common to both segments, and this common segment is shorter than the surrounding segments of slope $\frac{1}{2}$. One might say that the original digitized trajectory has a pimple at slope $\frac{1}{2}$.

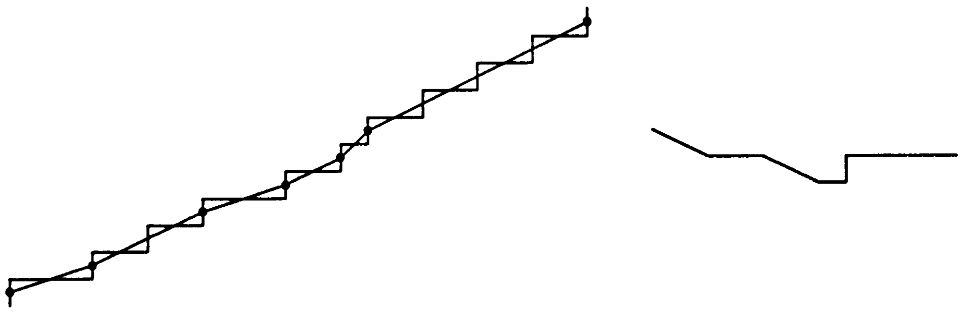

Fig. 39. A polygonized digital path and its image under T_1.

The pimple in Figure 39 is rather subtle. It would be more obvious if there were no edge of slope $\frac{1}{2}$ at the boundary between the T_2 section and the T_1 section, or if the pimple occurred at slope 1 instead of at slope $\frac{1}{2}$. The way to avoid such pimples at any slope $1/k$ is to adjust the trajectory so that all vertical tangents of $T_k(\mathcal{T})$ occur at integer x-coordinates. If desired, similar fixes can be made at other rational slopes by adjusting the vertical and horizontal tangent points of $\mathcal{T}$ after repeated applications of transformations such as T_k and U_k.

One way to make such adjustments to the trajectory is through linear transformations. Suppose we have a path $X(t)$ that represents the trajectory, and the rational slope tangents that we wish to adjust are attained at times $t_1, t_2, \ldots, t_n$. We wish to adjust the trajectory so that each $X(t_i)$ becomes some point X_i. Barring degeneracies, there will be for each i a unique linear transformation that takes $X(t_i)$ to $X(i)$, $X(t_{i+1})$ to X_{i+1} and preserves the directions $X'(t_i)$ and $X'(t_{i+1})$. It may be necessary to forgo some of the adjustments if the amount of distortion induced by some of these transformations becomes too great.

The appendix shows how the idea of polygonization from Section 5.2 can be extended so as to achieve smoothing without transforming and redigitizing the trajectory. This technique has other applications including data compression and curve fitting.

Appendix

Tracings and Convolutions

The purpose of this appendix is to introduce the theory of tracings and convolutions from [8] and [24], and to use this theory to extend the previous results. The theory provides important generalizations of key concepts such as paths, envelopes, and regions bounded by closed paths. By generalizing the concept of a path and factoring out extraneous details, Guibas, Ramshaw, and Stolfi are able to create an elegant mathematical framework in which brush envelopes are of central importance. Indeed the primary motivation for the theory of tracings and convolutions is to improve the understanding of brush envelopes.

Since the published description of the theory of tracings in [8] deals only with polygonal objects, we begin with a brief description of the generalization to piecewise real analytic curves from [24]. The concepts defined in the next section directly correspond to the similarly named concepts from [8], so the intuitive explanations from that source are also applicable to the following definitions.

A.1. Definitions

A basic feature of the theory of tracings is the need to keep track orientation as well as position. Thus we define a *state* to be a pair consisting of a position $(x, y) \in \mathbb{R}^2$ and a unit direction vector $(u, v) \in \mathbb{R}^2$.

A *trip* is a continuous, piecewise real analytic function from some interval I into the set of all states; i.e., a trip gives a sequence of states whose positions describe a continuous piecewise real analytic curve. We also require that the direction vector $\big(u(t), v(t)\big)$ must be a piecewise real analytic function for $t \in I$, and that $u(t)y'(t) = v(t)x'(t)$ where $\big(x(t), y(t)\big)$ is the position function. That is, when the velocity vector is nonzero, the direction vector must be collinear with it.

A *tour* is a closed trip; i.e., a trip whose initial and final states are identical. If $\mathcal{T}$ is a tour defined on an interval whose endpoints are a and b, then $\mathcal{T}(a) = \mathcal{T}(b)$. Note that it does not matter whether the interval is open or closed: $\mathcal{T}$ can be extended to a unique real analytic function on the closed interval $[a, b]$.

Trips and tours are made up of primitive parts called *germs*. A germ is a trip where both the position and the direction vector are real analytic functions and the direction function is 1 to 1.

If the position function is constant and the direction vector rotates through some angle $\leq 360°$ then the germ is a *turn*. Since no direction can be repeated, the direction must rotate monotonically either to the left or to the right. Thus turns can be distinguished as either *left turns* or *right turns*.

A germ is called a *segment* if the direction vector is constant and the position function describes a line segment. If the direction vector is a positive multiple of the velocity vector then the segment is called a *forward segment*, otherwise it is called a *backward segment.*

A germ that is neither a turn nor a segment is called an *arc*. In this case the position function describes a curve with no points of inflection. The velocity vector can be zero for at most a finite number of parameter values. For all other $t \in I$, the constant of proportionality between the velocity and direction vectors must have the same sign. If this constant is positive then the arc is a *forward arc*, otherwise it is a *backward arc*. Alternatively, arcs may be distinguished as *left turning arcs* or *right turning arcs* depending on how their direction vector changes.

For technical reasons, we also need a concept similar to a trip, except with less specific information about the parameterization. For this purpose we shall use *generalized multisets* of states. A generalized multiset of states is a multiset where the weight associated with each state is a 4-tuple of integers $(\tau^-, \tau^+, \sigma^-, \sigma^+)$. To take the union of such multisets we add corresponding weights componentwise using the convention that a state belongs to a multiset if and only if its weight is not the zero vector.

If $(\tau^-, \tau^+, \sigma^-, \sigma^+)$ is the weight of some state (U, X), then (τ^-, τ^+) is called the *curve weight* of (U, X) and (σ^-, σ^+) is called the *segment weight*. Intuitively, the curve weight gives the number of times the trip passes into and out of the state (U, X) with the direction vector rotating counterclockwise. Similarly, the segment weight describes the number of forward segments passing into and out of (U, X).

Let $\mathcal{G}$ be a segment defined on an interval I where $(a, b) \subseteq I \subseteq [a, b]$. Let $(\alpha, \beta, \gamma) = (a, b, 1)$ if $\mathcal{G}$ is a forward segment, and $(\alpha, \beta, \gamma) = (b, a, -1)$ otherwise. Then $\mathcal{G}$ corresponds to a generalized multiset where the weight of $\mathcal{G}(t)$ is $(0, 0, \gamma, \gamma)$ for $a < t < b$, the weight of $\mathcal{G}(\alpha)$ is $(0, 0, 0, \gamma)$, and the weight of $\mathcal{G}(\beta)$ is $(0, 0, \gamma, 0)$. All other states have weight zero.

Now let $\mathcal{G}$ be an arc or a turn defined on a similar interval I; and let $(\alpha, \beta, \gamma) = (a, b, 1)$ if $\mathcal{G}$ is a left turn or a left turning arc, otherwise let $(\alpha, \beta, \gamma) = (b, a, -1)$. The weight of the state $\mathcal{G}(t)$ is $(\gamma, \gamma, 0, 0)$ if $a < t < b$, $(0, \gamma, 0, 0)$ if $t = \alpha$, and $(\gamma, 0, 0, 0)$ if $t = \beta$. Again, all other states have weight zero.

It is instructive to compare these definitions to those of [8]. Trips referred to as *moves* in that source are called segments here so that we can use the term *move* for all germs that are not turns. We have introduced arcs to handle the generalization to piecewise real analytic curves. Our generalized multisets of states are analogous the signed multisets of [8] in that a generalized weight of $(\tau^-, \tau^+, \sigma^-, \sigma^+)$ corresponds to a simple weight of $(\tau^- + \tau^+ + \sigma^- + \sigma^+)/2$, but generalized weights contain the information obtained via state counting functions in [8].

The correspondence between germs and generalized multisets may be extended to arbitrary trips by subdividing the domain interval so as to obtain a set of germs. Specifically, let $\mathcal{T}$ be a trip defined on an interval $[a, b]$; and let $a = a_0 < a_1 < a_2 < \cdots < a_k = b$ such that the restriction of $\mathcal{T}$ to $[a_{i-1}, a_i]$ is a germ whenever $1 \leq i \leq k$. The generalized multiset corresponding to $\mathcal{T}$ is the union of the multisets corresponding to the indicated germs. As explained in Section 3.3 of [8], the properties of multisets guarantee that this is well defined.

A generalized multiset that corresponds to a germ is called a *primitive partial tracing*. In general, a *partial tracing* is the union of a finite number of primitive partial tracings. Thus the generalized multiset associated with a finite set of trips is a partial tracing. A partial tracing is *connected* if it corresponds in this manner to a single trip. A partial tracing is *bounded* if there is some bounded subset $R \subset \mathbb{R}^2$ such that for any state (U, X) whose weight is nonzero, $X \in R$. A partial tracing that corresponds to a tour or a set of tours is called simply a *tracing*. When this set of tours is empty, we obtain a generalized multiset where all weights are zero. This multiset is called the *null tracing*.

Partial tracings can be viewed as a representation of trips with extraneous details of the parameterization factored out. Specifically, a primitive partial tracing corresponds to a family of germs that are equivalent with respect to monotone increasing real analytic reparameterization; i.e., if $\mathcal{T}_1$ and $\mathcal{T}_2$ both correspond to a germ $\mathcal{G}$ then $\mathcal{T}_1(f(t)) = \mathcal{T}_2(t)$ for some monotone increasing real analytic function f that maps the domain of $\mathcal{T}_2$ into the domain of $\mathcal{T}_1$.

Trips made up of more than one germ correspond to partial tracings that are a union of primitive partial tracings. Different trips that correspond to a particular partial tracing are not necessarily equivalent under monotone reparameterization because the germs may be reordered, and one trip may contain germs that cancel out; i.e., the union of the corresponding primitive partial tracings might be the null tracing.

Brushes and trajectories as we have defined them correspond to trips and tracings in the following manner: The boundary of a convex brush is composed of a finite number of real analytic curves, and it can easily be parameterized as a trip. By convention, the boundary is traced counter-clockwise as t increases. A tracing that corresponds to such a trip is a union of left turns, forward segments, and forward, left turning arcs. In addition, for any unit vector U, if S_U is the set of all states having direction vector U, then the sum of the curve weights for all states in S_U must be $(1,1)$. In other words, the tracing should describe trips where the direction vector passes through each possible direction exactly once. Such a tracing is called a *convex tracing*.

The interpretation of a trajectory as a tracing is somewhat different. If the trajectory is not connected, then it corresponds to more than one trip. Otherwise, the trajectory can be thought of as a continuous, piecewise real analytic function $z(t)$ from some interval $[a, b]$ to a subset of $\mathbb{R}^2$, with associated directions $w(t)$ where turning at corners is by as small an angle as possible. The trajectory tracing corresponds to the following set of trips: the trip $z(t)$ with direction $w(t)$ for $a \le t \le b$, the trip $z(-t)$ with direction $-w(-t)$ for $-b \le t \le -a$, and counter-clockwise $180°$ turns at $z(a)$ and $z(b)$. In other words, we pretend that the trajectory is an infinitesimally thin sausage shape and trace around its boundary, as shown in Figure 40.

We also need to deal with unbounded trajectories where it is $z(a)$ and $z(b)$ are not defined. We can do this by simply omitting the $180°$ turns and obtaining

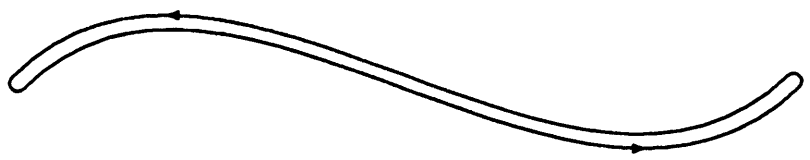

Fig. 40. A representation of the tracing corresponding to a trajectory.

a partial tracing instead of a true tracing. For instance, an infinite straight line trajectory is represented by a tracing that corresponds to two unbounded forward segments with opposite orientations.

An important concept introduced in [8] is the *convolution* of two tracings. The basic idea behind convolution is to take two tracings $\mathcal{T}_1$ and $\mathcal{T}_2$, and consider all pairs of states $(U,X) \in \mathcal{T}_1$ and $(U,X') \in \mathcal{T}_2$ where the direction vectors match. The corresponding state in the convolution tracing is $(U, X + X')$. If $(\tau_1^-, \tau_1^+, \sigma_1^-, \sigma_1^+)$ is the weight of (U,X) and $(\tau_2^-, \tau_2^+, \sigma_2^-, \sigma_2^+)$ is the weight of (U,X'), then the corresponding contribution to the weight of $(U, X + X')$ is

$$(\tau_1^-\tau_2^-, \tau_1^+\tau_2^+, \bar\tau_2\sigma_1^- + \bar\tau_1\sigma_2^-, \bar\tau_2\sigma_1^+ + \bar\tau_1\sigma_2^+)$$

where $\bar\tau_1 = (\tau_1^- + \tau_1^+)/2$ and $\bar\tau_2 = (\tau_2^- + \tau_2^+)/2$. The weight of a state (U,Y) in the convolution is the sum of all such contributions for all $(U,X) \in \mathcal{T}_1$ and $(U,X') \in \mathcal{T}_2$ where $X + X' = Y$.

Since tracings are required to be finite unions of primitive partial tracings, it is important to note that the above definition has a simple interpretation in such a representation if we take advantage of the fact that the definition also applies to partial tracings. A tracing $\mathcal{T}$ may be represented as a finite union $w_1\mathcal{G}_1 \cup w_2\mathcal{G}_2 \cup \cdots \cup w_n\mathcal{G}_n$ of integer multiples of primitive partial tracings $\mathcal{G}_i$, where $w_i\mathcal{G}_i$ is a tracing identical to $\mathcal{G}_i$ except that all weights are multiplied by w_i. The convolution of $w_1\mathcal{G}_1 \cup w_2\mathcal{G}_2 \cup \cdots \cup w_n\mathcal{G}_n$ and $w'_1\mathcal{G}'_1 \cup w'_2\mathcal{G}'_2 \cup \cdots \cup w'_{n'}\mathcal{G}'_{n'}$ is

$$\bigcup_{\substack{i \leq i \leq n \\ 1 \leq j \leq n'}} w_i w'_j \mathcal{G}_{ij}$$

where $\mathcal{G}_{ij}$ is the convolution of the primitive partial tracings $\mathcal{G}_i$ and $\mathcal{G}'_j$.

In general the convolution of two primitive partial tracings might not be a primitive partial tracing, but it can always be expressed as the union of a finite number of primitive partial tracings, and the required number is usually small. To simplify the following discussion, it is convenient to use the terms developed for germs to refer to the primitive partial tracings that they represent. Thus the convolution of two turns is a single turn, and the convolution of two arcs is a set of arcs. For instance, the convolution of two forward, left turning arcs is a union of k forward, left turning arcs and l backward, left turning arcs, where $k, l \geq 0$ and $|k - l| \leq 1$. The convolution of a segment with a turn or an arc is either a segment, or a segment with all the weights multiplied by $\frac{1}{2}$. It turns out that no such nonintegral weights will remain when all terms in the convolution of two tracings are considered.

We have already seen how to represent the brush and trajectory as tracings and how to compute their convolution. The interpretation of the resulting tracing as a brush envelope depends on the concept of *winding number.*

To find the winding number of a point p with respect to a tracing $\mathcal{T}$ containing no states with position p, choose a trip or set of trips that corresponds to $\mathcal{T}$, and follow a semi-infinite ray directed outward from p, taking the number of times the trip crosses it from right to left, minus the number of times the trip crosses from left to right. For instance, in Figure 41, the winding number of p with respect to the tracing shown is $2 - 1 = 1$. Guibas, Ramshaw and Stolfi have an elegant definition of the winding number that applies when p lies on $\mathcal{T}$, but it is not useful here because of our treatment of region boundaries in Section 1.1. Thus we leave the winding number undefined when p lies on $\mathcal{T}$.

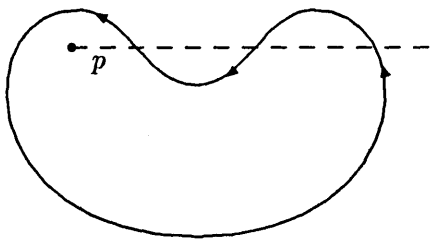

Fig. 41. How to find the winding number.

The winding number of a partial tracing depends on the orientation of the ray from p, so it is really better to define the winding number for states rather than just positions. As shown in [8], the winding number with respect to a tracing is independent of the direction, so we shall only give the direction when it matters.

The concept of winding number provides a precise definition of the interior of a tour or set of tours. The corresponding tracing contains all the information necessary to determine the winding number, and the interior is the set of points where the winding number is positive. For instance, the winding number of a point with respect to a convex tracing is always either 0 or 1, and the set of points where the winding number is 1 agree with the intuitive notion of "interior."

The interpretation of the convolution as an envelope depends on the convolution theorem given in [8] and generalized in [24] to cover non-polygonal tracings. For a convex brush $\mathcal{B}$ and a trajectory T, it follows from the convolution theorem that the winding number of a point p with respect to the convolution tracing is the number of distinct intervals of t for which $p - z(t) \in \mathcal{B}$, where $z(t) = \bigl(x(t), y(t)\bigr)$ is the position function for a trip that corresponds to T. Informally, the conclusion is that the convolution tracing not only gives the envelope, it also tells how many times each point gets painted. Point p is said to lie inside the brush stroke defined by $\mathcal{B}$ and $\mathcal{T}$ if its winding number is positive.

If the trajectory is unbounded then the trajectory is represented by a partial tracing and the convolution with any tracing $\mathcal{B}$ is only a partial tracing. In spite of this, we often can use winding numbers to interpret the result. For instance, if the trajectory is an infinite straight line, the convolution is a partial tracing that corresponds to a set of unbounded segments parallel to the trajectory. The winding number of a state (U, X) with respect to the convolution tracing is independent of the direction U' as long the direction vector U not parallel to the trajectory.

Thus convolution can be used to compute brush envelopes, but parts of the convolution tracing may not lie on the envelope boundary; they may instead delineate interior regions that are colored more than once. In practice, such regions of multiple coloring are often quite small, so that most of the convolution tracing does indeed describe the envelope boundary. For a convex polygonal brush, the convolution tracing is made up of shifted pieces of the trajectory, connected with straight lines that correspond to polygon edges as shown in Figure 42. Here dots mark the ends of such straight lines and arrows indicate the direction vectors, not the direction of motion. The brush and trajectory are shown as well as the convolution tracing.

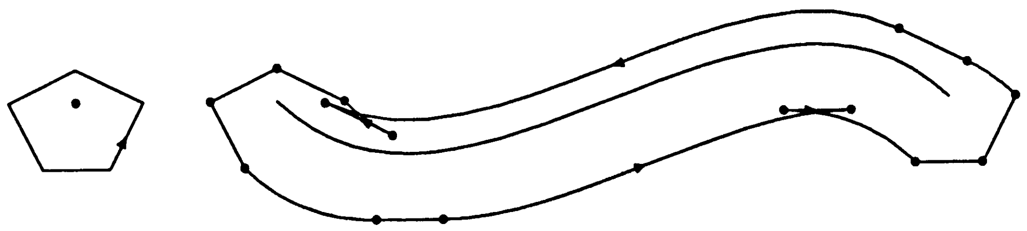
Fig. 42. A brush, a trajectory, and their convolution tracing.

A.2. Monostrophic Pens

Perhaps the most important application of the theory of tracings to this work is that it allows us to generalize the concepts of brushes and pens. We have seen how a convex brush can be described as a tracing and how the tracing can be used to compute envelopes. Any tracing can potentially be used as a brush in this fashion merely by taking the convolution with the trajectory tracing and using the positive winding number rule to interpret the result.

Since pens are brush shapes that lead to good envelope widths for infinite straight line trajectories of rational slope, we shall consider tracings that behave analogously. Let $\mathcal{T}$ be a partial tracing that represents an infinite straight line trajectory as outlined in Section A.1. We can restrict our attention to tracings $\mathcal{B}$ whose convolution any such $\mathcal{T}$ must have a single region of positive winding number of the form $|ax + by - c| < d$. If $\mathcal{B}$ is fixed, the region of positive winding number depends only on the choice of the trajectory line. If we are only interested in how this region depends on the trajectory line, we can restrict $\mathcal{B}$ to belong to a special class of tracings called *monostrophic tracings*. This is analogous to our assumption that pens must be convex: A nonconvex brush can have good widths in all rational directions, but only if its convex hull is a pen. Similarly a non-monostrophic tracing can function as a pen, but there is always a monostrophic version for which the region of positive winding number in the convolution with an infinite straight line trajectory tracing is always identical.

Monostrophic tracings are just like convex tracings, except that they can have backward segments and backward left turning arcs. Like convex tracings, monostrophic tracings also correspond to tours where the direction vector makes exactly one complete counterclockwise rotation. Figure 43 shows an example of such a tracing. Note that the arrows indicate the orientation of the direction vectors, not the order in which the states would be traversed by a corresponding tour. For more information on monostrophic tracings, see GRSa.

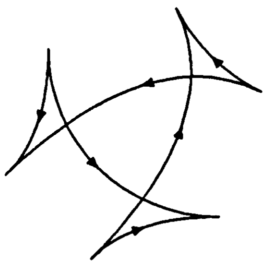

Fig. 43. A monostrophic tracing.

If we try to use winding numbers to assign a physical interpretation to a monostrophic tracing that is not convex, we find that it has regions of negative winding number. For instance, points in the central triangle of Figure 43 have winding number 1 with respect to the tracing, but the other three enclosed regions have winding number -1. Taking the convolution with respect to a trajectory tracing and analyzing winding numbers with respect to the result reveals that the behavior can be unlike that of any physical pen, even if we somehow allow parts of it to have "negative weight."

When the trajectory is an infinite straight line, the convolution with a monostrophic tracing has a single region of weight 1 bounded by two parallel lines. These lines may be thought of as supporting lines of the monostrophic tracing. If the trajectory is the set of points $\{\,\alpha U \mid \alpha \in \mathbb{R}\,\}$ for some unit vector U, then the supporting lines pass through the points on the monostrophic tracing where the direction vector is $\pm U$. Thus, it is natural to refer to these as the *points of support* of the monostrophic tracing in the directions $\pm U$.

The perpendicular component of the separation between points of support gives the *width* of the monostrophic tracing in the direction of the trajectory. We can now define a *monostrophic pen* to be a monostrophic tracing that has a *good width* in every rational direction; i.e., the width must be an integer multiple of $1/\sqrt{a^2+b^2}$ whenever the direction is a reduced rational pair (a,b). Since convex tracings are also monostrophic, ordinary polygonal pens can also be viewed as monostrophic pens.

To characterize monostrophic pens, we begin by generalizing Lemma 3.1.2 to monostrophic tracings. Instead of distinguishing between left and right supporting points, the distinction is based on the two opposite possibilities for the direction vector at the supporting points.

Lemma A.2.1. *If a monostrophic tracing $\mathcal{B}$ has a good width in every rational direction, then for any rational direction (a,b) that is not the direction of any segment corresponding to a subset of $\mathcal{B}$, the points of support of $\mathcal{B}$ in the directions $\pm(a,b)$ has integer components.*

Proof. The proof of Lemma 3.1.2 applies. ∎

All the arguments of Section 3.1 hold for monostrophic pens, except that we can no longer conclude so much from the fact that the difference between opposite supporting points is independent of direction. In fact, we shall introduce a whole class of monostrophic tracings for which this difference is always $(0,0)$.

In order to complete the characterization of monostrophic pens, we need to extend the ideas of Section 3.3 to cover monostrophic tracings. Let $\mathcal{B}$ be a monostrophic tracing, and let $z(t)$ and $\bar{z}(t)$ be two different trips represented by $\mathcal{B}$ with width functions $w(t)$ and $\bar{w}(t)$ respectively. If the trips are parameterized so that $w(t)+\bar{w}(t) = 0$ for all t, then the functions $z(t)$ and $\bar{z}(t)$ give the locations of points a and c in the discussion of Figure 6a. To construct a symmetrical version of $\mathcal{B}$ with the same width as a function of direction, let $\mathcal{F}(\mathcal{B})$ be the tracing represented by $\frac{1}{2}z(t)-\frac{1}{2}\bar{z}(t)$ with direction $w(t)$ for the same range of t. The path of the point b in Figure 6a corresponds to the tracing represented by $\mathcal{M}(\mathcal{B}) = \left(\frac{1}{2}z(t) + \frac{1}{2}\bar{z}(t)\right)$ with direction $w(t)$.

The tracing $\mathcal{M}(\mathcal{B})$ corresponds to a set of tours and their *directional reversals*, where the directional reversal of a trip $z(t)$ with directions $w(t)$ is obtained by just negating the function w. In other words, if a state (U,X) has weight $(\tau^-,\tau^+,\sigma^-,\sigma^+)$ with respect to $\mathcal{M}(\mathcal{B})$ then the state $(-U,X)$ has weight $(-\tau^+,-\tau^-,-\sigma^+,-\sigma^-)$. In general, monostrophic tracings with this property are called *double monostrophic tracings.* Such a tracing can be represented by a trip $z(t)$ with direction $w(t)$ on an interval $a \le t < b$ such that $z(t) = z(\bar{t})$ and $w(t) + w(\bar{t}) = (0,0)$ whenever $\bar{t} - t = (b-a)/2$ and t and $\bar{t}$ are both in the domain. Loosely speaking, double monostrophic tracings have star-shaped paths with odd numbers of cusps.

The functions $\mathcal{M}$ and $\mathcal{F}$ are constructed in such a way as to ensure that $\mathcal{B}$ is the convolution of $\mathcal{M}(\mathcal{B})$ and $\mathcal{F}(\mathcal{B})$. Any other monostrophic tracing $\mathcal{B}'$ whose width as a function of direction is identical to that of $\mathcal{B}$ will have $\mathcal{F}(\mathcal{B}') = \mathcal{F}(\mathcal{B})$, and $\mathcal{B}'$ will be the convolution of $\mathcal{F}(\mathcal{B})$ and $\mathcal{M}(\mathcal{B}')$. Thus any monostrophic tracing whose width as a function of direction matches $\mathcal{F}(\mathcal{B})$ is the convolution of $\mathcal{F}(\mathcal{B})$ with a double monostrophic tracing.

Since $\mathcal{P}$ is a monostrophic pen if and only if $\mathcal{F}(\mathcal{P})$ is, it is sufficient to characterize monostrophic pens that are symmetrical about the origin. If a point X is a supporting point for such a pen in some direction U, then the difference between the supporting points in directions $\pm U$ is $2X$. Thus Lemma A.2.1 proves the following characterization theorem.

Theorem A.2.2. *A monostrophic tracing $\mathcal{T}$ is a monostrophic pen if and only if $\mathcal{T}$ is the convolution of a monostrophic tracing $\mathcal{S}$ and a double monostrophic tracing $\mathcal{D}$, where S is symmetrical about the origin, is polygonal, and has vertices in $\frac{1}{2}\mathbb{Z}^2$.* ∎

We shall use Theorem A.2.2 to show that monostrophic pens have integer offset properties similar to polygonal pens, and thus they perform well for curved trajectories as well as for infinite straight lines. Figure 44 shows how the convolution tracing provides a convenient tool for finding integer offsets in a monostrophic pen envelope. The integer offset curves are delimited by parallel dashed lines, and arrowheads indicate the direction vector, not necessarily the direction of motion. It is not hard to see that all curved portions of the convolution tracing belong to such pairs of curves: If we use the terms "arc," "turn," and "segment" to refer to the classes of primitive partial tracings to which they correspond, all primitive partial tracings that make up the monostrophic pen are either turns or segments, and any arc in the convolution tracing can be obtained by taking the convolution of a turn in the brush tracing with an arc in the trajectory tracing. Theorem A.2.2 guarantees that for any two states (U, X) and $(-U, Y)$ belonging to turns in the brush tracing, $X - Y \in \mathbb{Z}^2$. Since arcs in the trajectory tracing come in pairs that describe identical curves with opposite orientations, it follows that all arcs in the convolution tracing come in pairs that describe curves that are identical except shifted by integer offset vectors.

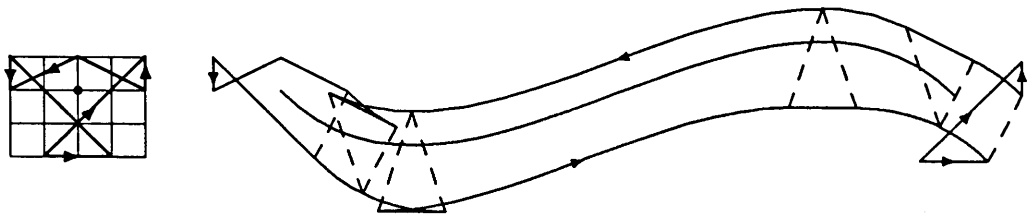

Fig. 44. A monostrophic pen tracing and its convolution with a trajectory, showing regions where integer offset vectors apply.

Figure 44 illustrates another property of monostrophic pen envelopes: the winding number can be negative. There are four regions of winding number -1 in the figure: two at the right end of the stroke, one at the left end, and one at the lowest point on the lower edge. It is necessary to be able to cope with such "negatively colored" regions when using monostrophic pens that are not convex. The most reasonable way to do this is probably to color only the regions where the winding number is positive.

We can also extend our analysis of offset angles to monostrophic pens. The lower bound of Lemma 5.1 can be extended to show that even arbitrary monostrophic pens must have offset angles that are $\Omega(d^{-1/2})$ in the pen diameter d. The *diameter* of a monostrophic pen $\mathcal{P}$ is the diameter of the set of all positions X for which there is some state (U, X) in $\mathcal{P}$.

Since $\mathcal{F}$ preserves the displacement between opposite supporting points, we can assume that the pen is symmetrical about the origin. The endpoints of the edge whose direction is counterclockwise adjacent to $(1,0)$ must still be separated by a half integer multiple of the reduced rational direction (u, v), hence we have the following corollary to Lemma 5.1.

Corollary A.2.3. *If $\mathcal{P}$ is a monostrophic pen with diameter at most $d + \frac{3}{4}$, and θ is its maximum offset angle over all possible trajectory directions, then* $\tan\theta \geq 1/(2\sqrt{d+1})$. ∎

A.3. Asymmetrical Brush Shapes and Monostrophic Pens

The functions $\mathcal{F}$ and $\mathcal{M}$ allow us to adapt the ideas of Chapter 4 to generate monostrophic pens if desired. Given a monostrophic tracing $\mathcal{B}$, we can find a monostrophic pen $\mathcal{P}$ that approximates $\mathcal{F}(\mathcal{B})$ and thus accurately reflects the desired width as a function of path direction. If we are interested in more than just the width as a function of direction for nearly straight trajectories, we can take the convolution of $\mathcal{P}$ with the double monostrophic tracing $\mathcal{M}(\mathcal{B})$ to yield a monostrophic pen that truly approximates $\mathcal{B}$.

This technique is also worth investigating even if we wish to require all pens to be polygonal. If $\mathcal{B}$ is a convex tracing then it describes an ordinary convex brush, and Algorithm 1 from Section 4.1 can be used to find a polygonal pen that approximates $\mathcal{F}(\mathcal{B})$. By using an appropriate polygonal approximation to $\mathcal{M}(\mathcal{B})$, it is possible to obtain a tracing that describes an ordinary polygonal pen, but the techniques of Section 4.2 turn out to be superior for most applications.

Let $\mathcal{B}$ be a monostrophic tracing that is symmetrical about the origin, and consider how Algorithm 1 could be modified to generate a symmetrical monostrophic pen that approximates $\mathcal{B}$. The function $\mathcal{B}(u,v)$ used in Algorithm 1 now just refers to the point of support of the tracing $\mathcal{B}$ in the direction (u,v). The modified algorithm could always set $\bar{z} \leftarrow \mathcal{B}(\bar{u},\bar{v})$ in Step 2, and it would not have to require that $\delta \le \min\bigl(rl(p_l), ll(p_r)\bigr)$ in Step 2. In fact it would not be necessary to go on to the next vertex when $\delta < 0$ in Step 3, so there are no limits as to what directions can be considered for pen edges. The sequence of edge and vertex nodes computed by the algorithm defines a polygonal monostrophic tracing where edge nodes determine segments and vertex nodes determine turns. Forward segments are associated with edges e where $ll(e) + rl(e) > 0$, and backward segments arise when $ll(e) + rl(e) < 0$.

Some stopping condition is clearly required in order to control the size of the set of edge directions considered. Any stopping condition that does not allow an excessive number of long backward segments is feasible, and the choice should probably depend on the intended application. One important special case is where $\mathcal{B}$ is simply a circle of some diameter w, and the stopping condition is that the direction $(\bar{u},\bar{v})$ in Step 2 must be simple for width w in the sense that $\alpha(\bar{u}^2+\bar{v}^2) \le w$ for some constant α as defined in Section 5.4. Lemma 5.4.1 shows that the monostrophic pen so generated produces exactly the same integer offset vectors as Algorithm 3 does. Thus as promised in Section 5.1, Corollary A.2.3 shows that the offset angle must be $\Omega(w^{-1/2})$ for some trajectory directions.

If $\mathcal{B}$ is a convex tracing and a polygonal pen is desired, then we can still begin by finding polygonal pen $\mathcal{P}'$ that approximates $\mathcal{B}' = \mathcal{F}(\mathcal{B})$. We can then try to find a polygonal double monostrophic tracing $\mathcal{M}'(\mathcal{B})$ that approximates $\mathcal{M}(\mathcal{B})$, such that the convolution of $\mathcal{P}'$ and $\mathcal{M}'(\mathcal{B})$ is a convex tracing. Since the convolution operator is associative and commutative, the effect of using such a pen with a trajectory tracing $\mathcal{T}$ is identical to what would be achieved by using $\mathcal{P}'$ as the pen and taking the convolution of the trajectory tracing $\mathcal{T}$ with $\mathcal{M}'(\mathcal{B})$.

Figure 45a shows an enlarged version of the double monostrophic tracing

$\mathcal{M}(\mathcal{B})$ shown in Figure 6a. Superimposed on the tracing is a half-unit grid compatible with the one shown in Figure 7. In the following discussion we shall refer to the convex tracings shown in Figures 6a and 6b as $\mathcal{B}$ and $\mathcal{B}'$, and we shall use $\mathcal{P}'$ for the pen tracing generated from $\mathcal{B}'$ as shown in Figure 7.

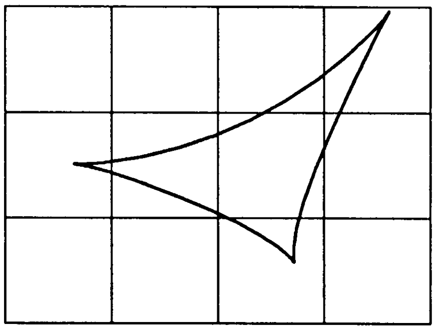
Fig. 45a. The double monostrophic tracing for obtaining Figure 6a from Figure 6b.

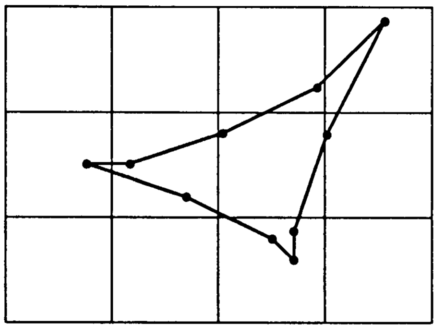
Fig. 45b. The polygonal double monostrophic tracing for skewing the pen of Figure 7 so that it approximates Figure 6a.

Let $\circ$ be the convolution operator so that $\mathcal{B} = \mathcal{B}' \circ \mathcal{M}(\mathcal{B})$. It seems natural to look for a tracing $\mathcal{M}'(\mathcal{B})$ similar to $\mathcal{M}(\mathcal{B})$ such that $\mathcal{P}' \circ \mathcal{M}'(\mathcal{B})$ corresponds to a convex polygonal pen that approximates $\mathcal{B}$. Figure 45b shows what $\mathcal{M}'(\mathcal{B})$ might look like. This tracing is composed of line segment moves, each of which lies on a line tangent to $\mathcal{M}(\mathcal{B})$. Furthermore there is a one to one correspondence between the edge directions in the data structure of Algorithm 1 and the directions of the segments in Figure 45b.

Thus we have a function $\mathcal{M}'(\mathcal{B})$ that yields polygonal double monostrophic tracings. If Algorithm 1 halts with edges $e_0, e_1, \ldots, e_k$ in its data structures, then $\mathcal{M}'(\mathcal{B})$ contains straight moves with directions $\pm w(e_i)$ for $0 \leq i < k$. For a particular i, these moves lie on a line midway between the supporting lines of $\mathcal{B}$ in the directions $\pm w(e_i)$, and its endpoints are where this line intersects similar lines for directions $w(e_{i-1})$ and $w(e_{i+1})$. (If $i = 0$ then the directions of the adjacent segments should be $w(e_{k-1})$ and $w(e_1)$.) Inserting the necessary turns and their directional reversals yields a polygonal double monostrophic tracing.

With the above definition of $\mathcal{M}'$, the polygonal tracing $\mathcal{P} = \mathcal{P}' \circ \mathcal{M}'(\mathcal{B})$ is not necessarily convex because some of the backward segments from $\mathcal{M}'(\mathcal{B})$ may be longer than the similarly directed forward segments in $\mathcal{P}'$. Figure 46 shows the polygonal monostrophic pen obtained by taking the convolution of the tracings illustrated in Figures 7and 45b. It fails to be convex because the monostrophic tracing contains segments of slopes 2 and 0, while the other does not.

The function $\mathcal{M}'$ is useful because it leads to monostrophic pens that are polygonal and hence much easier to work with than those obtained with non-polygonal double monostrophic tracings such as that shown in Figure 45a. However, it would be nice to have to be able to generate polygonal double monostrophic tracings like $\mathcal{M}'(\mathcal{B})$ where the backward segments are no longer than the similarly directed forward segments in the pen $\mathcal{P}'$. It is difficult to construct a simple algorithm for generating optimum polygonal double monostrophic tracings subject

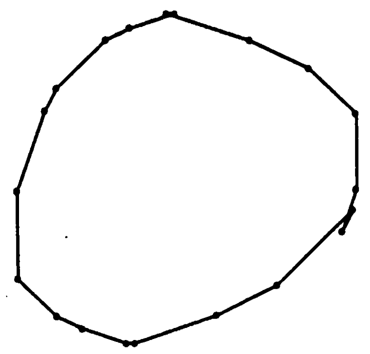

Fig. 46. The monostrophic pen obtained by taking the convolution of the tracings shown in Figures 7 and 45b

to the above restriction, but a nearly optimum solution can actually be found by linear programming. Specifically, let $e_0, e_1, \ldots, e_k$ be the edges in the data structure of Algorithm 1 after it has processed a brush $\mathcal{F}(\mathcal{B})$; let $(u_i, v_i) = w(e_i)$ and $l_i = ll(e_i) + rl(e_i)$ for $0 \le i \le k$; and let a_i be such that the tangent points $\mathcal{B}(u_i, v_i)$ and $\mathcal{B}(-u_i, -v_i)$ are equidistant from the line $xv_i - yu_i = a_i$ for $0 \le i \le k$.

The double monostrophic tracing D will be defined by parameters $\alpha_0, \alpha_1, \ldots, \alpha_k$ where $\alpha_k = -\alpha_0$. These in turn define lines $\ell_0, \ell_1, \ldots, \ell_k$ where ℓ_i is the line $xv_i - yu_i = \alpha_i$. (Note that $\ell_0 = \ell_k$.) The tracing D represents a trip that starts where ℓ_0 intersects ℓ_1, follows ℓ_1 until it intersects ℓ_2, then follows ℓ_2, etc. When following ℓ_i, the associated direction is always (u_i, v_i) regardless of whether this is forward or backward. From the intersection point of ℓ_{k-1} and ℓ_k, the tracing follows ℓ_k until reaching its starting point. The entire cycle is then repeated with the sense of direction of each segment reversed.

Ideally, we would like all $\alpha_i = a_i$ as in $\mathcal{M}'(\mathcal{B})$, but the constraints on D may not allow this. It follows from (4.1.1) that the movement in any direction (u_i, v_i) satisfying $0 < i < k$ is by an amount

$$\big((\alpha_{i-1} + \alpha_{i+1})u_i - \alpha_i(u_{i-1} + u_{i+1}),\ (\alpha_{i-1} + \alpha_{i+1})v_i - \alpha_i(v_{i-1} + v_{i+1})\big)$$

or $\alpha_{i-1} + \alpha_{i+1} - \lambda_i\alpha_i$ times (u_i, v_i) where λ_i is the integer such that $\lambda_i(u_i, v_i) = (u_{i-1} + u_{i+1}, v_{i-1} + v_{i+1})$. The length can be determined exactly the same way for $i = k$ if we let $(u_{k+1}, v_{k+1}) = (u_1, v_1)$ and $\alpha_{k+1} = -\alpha_1$. Thus we have constraints

$$-l_i/2 \le \alpha_{i-1} + \alpha_{i+1} - \lambda_i\alpha_i \le l_i/2 \quad \text{for } 1 \le i \le k \qquad (A.3.1)$$

and we wish to minimize

$$\max_{1 \le i \le k} \left(\epsilon_i + |\alpha_i - a_i| \Big/ \sqrt{u_i^2 + v_i^2}\right) \qquad (A.3.2)$$

where ϵ_i is the distance between the supporting line of $\mathcal{F}(\mathcal{B})$ in the (u_i, v_i) direction and the corresponding edge of the pen $\mathcal{P}'$. Thus we take the maximum over i of the distance between the supporting lines of $\mathcal{B}$ parallel to $w(e_i)$ and the parallel supporting lines of $\mathcal{P}' \circ D$.

A slightly better approximation to the true total error can be obtained by adding terms to (A.3.2) that express the distance between $\mathcal{B}$ and supporting lines of the final pen in intermediate directions. In any case linear programming can be used to find the α_1, α_2, …, α_k that minimize (A.3.2) subject to (A.3.1). Unfortunately, even the optimum D can lead to large errors if we always start with the result of Algorithm 1 as suggested.

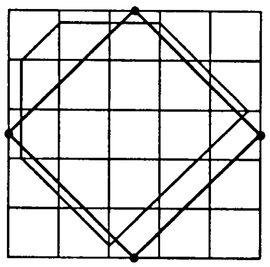

Fig. 47. A case where the symmetrical pen cannot be adjusted to yield a good approximation to the desired shape.

Figure 47 shows a brush $\mathcal{B}$ and the pen computed by Algorithm 1 from $\mathcal{F}(\mathcal{B})$. The brush has width 4.51 in directions $(1,0)$ and $(0,1)$, and width $5.49/\sqrt{2}$ in directions $(1,1)$ and $(-1,1)$. This causes Algorithm 1 to generate the simple diamond shape shown in bold lines. Since there is no non-trivial polygonal double monostrophic tracing whose moves have directions $(1,1)$ and $(-1,1)$, this diamond shape pen cannot be altered by convolution without generating backward moves. The figure shows how the diamond can have error $1.225/\sqrt{2}$ in approximating $\mathcal{B}$. We might try to reduce this figure by examining symmetrical pen shapes other than that produced by Algorithm 1, but it is difficult to limit the amount of computation required.

A.4. Complete Polygonization

The theory of tracings also allows some important extensions to the ideas of Section 5.2. We begin by extending the concept of a digital path to partial tracings. As promised in Section 1.1, this allows us to give a rigorous proof of the equivalence between digitization for paths and digitization for regions. By extending the concept of polygonization to tracings, we obtain a simple algorithm for generating polygonal approximations to digital paths.

Recall that a partial tracing is polygonal if it corresponds to a set of moves and segments. These segments are the *edges* of the partial tracing, and their endpoints are the *vertices*. Let a *digital tracing* be a polygonal partial tracing such that its vertices all lie in $\mathbb{Z}^2$ and its edges are all vertical or horizontal.

Let us first define digitization on trips and then extend it to tracings. Consider the trip $\big(U(t), X(t)\big)$ where $U(t) = \big(u(t), v(t)\big)$, $X(t) = \big(x(t), y(t)\big)$, and t ranges over an interval $[a, b]$. Let X_1, X_2, …, X_n be the vertices of the digitization of the path $X(t)$, and choose $t_0 \le t_1 \le t_2 \le \cdots \le t_n$ so that t_{i-1} and t_i are the endpoints of the interval during which $\big(\lceil x(t) - \frac{1}{2}\rceil, \lfloor x(t) + \frac{1}{2}\rfloor\big) = X_i$. A digitization of $\big(U(t), X(t)\big)$ is a polygonal trip that connects X_1, X_2, …, X_n by segments whose directions are chosen to match $U(t)$. Specifically the direction of the segment joining X_i to X_{i+1} should be as close as possible to $U(t_i)$ and the angle of the turn at X_i should be as close as possible to the angle through which $U(t)$ turns as t increases from t_{i-1} to t_i. We extend this to a function $\mathcal{D}$ on partial tracings by letting $\mathcal{D}(\mathcal{T})$ be the partial tracing that corresponds to a set of trips obtained by digitizing a set of trips that correspond to $\mathcal{T}$.

Theorem A.4.1. *Let $\mathcal{T}$ be a tracing and let R be the region of positive winding number with respect to that tracing. Then the region of positive winding number with respect to the digitization $\mathcal{D}(\mathcal{T})$ is the interior of the digitization of R.*

Proof. Since the moves of a digital tracing are confined to horizontal and vertical lines at integer coordinates, the winding number with respect to such a tracing must be constant on every pixel $P(m,n)$. Hence the interior of the region of positive winding with respect to $\mathcal{D}(\mathcal{T})$ is in fact the interior of a digital region. It only remains to prove that the set of points of the form $(m+\frac{1}{2}, n+\frac{1}{2})$ of positive winding number with respect to $\mathcal{D}(\mathcal{T})$ is identical to the set of such points in $(R \setminus B(R)) \cup B_L(R)$ where B and B_L are as in Section 1.1.

Let w be the winding number of a point $z = (m+\frac{1}{2}, n+\frac{1}{2})$ with respect to $\mathcal{D}(\mathcal{T})$. The definition of $\mathcal{D}(\mathcal{T})$ implies that w is the number of moves of $\mathcal{T}$ where the y-coordinate changes from $< n+\frac{1}{2}$ to $\geq n+\frac{1}{2}$ when the x-coordinate is $> m+\frac{1}{2}$, minus the number of similar moves where the y-coordinate changes from $\geq n+\frac{1}{2}$ to $< n+\frac{1}{2}$. If $\mathcal{T}$ has no states with position z, then $z \notin B(R)$ and w equals the winding number of z with respect to $\mathcal{T}$. Thus $z \in R$ if and only if $w > 0$.

If z is the position of a state in $\mathcal{T}$, then $z \notin R$ and we need only prove that $z \in B_L(R)$ if and only if $w > 0$. Let ϵ_1 be the minimum positive ϵ such that $\mathcal{T}$ contains a move whose y-coordinate changes from $< n+\frac{1}{2}$ to $\geq n+\frac{1}{2}$ or vice versa when the x-coordinate is $m+\frac{1}{2}+\epsilon$. If $0 < \delta_1 < \epsilon_1$, then if $0 < \delta_2 < \epsilon_2$ for sufficiently small ϵ_2, all moves of $\mathcal{T}$ that cross from $y < n+\frac{1}{2}$ to $y \geq n+\frac{1}{2}$ with $x > m+\frac{1}{2}$ also cross $y = n+\frac{1}{2}-\delta_2$ with $x > m+\frac{1}{2}+\delta_1$. Thus the winding number of any such point $z+(\delta_1, -\delta_2)$ with respect to $\mathcal{T}$ is equal to w. Hence $z \in B_L(R)$ if and only if $w > 0$. ∎

Since shapes are commonly represented via their boundaries, the above theorem provides a very convenient way of computing the digitization of a region. When building envelopes from integer offsets, there is the added benefit that if two partial tracings $\mathcal{T}_L$ and $\mathcal{T}_R$ are identical except shifted by some integer amount (a,b), then $\mathcal{D}(\mathcal{T}_L)$ and $\mathcal{D}(\mathcal{T}_R)$ also enjoy this property.

Now that we have extended digitization to tracings, we are ready to extend some of the other concepts from Section 5.2. For each $k \geq 1$ let $\mathcal{X}'_k$ be the set of partial tracings with no backward moves, containing only states whose direction is a nonnegative linear combination of $(k,1)$ and $(k+1,1)$. Similarly $\mathcal{Y}'_k$ is a set of polygonal partial tracings whose direction vectors are nonnegative linear combinations of $(1,k)$ and $(1,k+1)$.

Consider the transformations T_k and U_k defined in Section 5.2. The purpose of the sets $\mathcal{X}'_k$ and $\mathcal{Y}'_k$ is that applying T_k to a partial tracing in $\mathcal{X}'_k$ or applying

U_k to a member of $\mathcal{Y}'_k$ yields a partial tracing with no backward moves and no direction vectors outside of the first quadrant. A digital partial tracing with this property is called a *simple positive digital tracing.*

Lemma A.4.2. *If ℓ and ℓ' are connected unbounded partial tracings in $\mathcal{X}'_k$ for some integer k, then ℓ and ℓ' are digitally equivalent if and only if $T_k(\ell)$ and $T_k(\ell')$ are; similarly for $\{\ell, \ell'\} \subset \mathcal{Y}'_k$, ℓ and ℓ' are digitally equivalent if and only if $U_k(\ell)$ and $U_k(\ell')$ are.*

Proof. Let T be either T_k or U_k as appropriate. Since the directions of ℓ, ℓ', $T(\ell)$, and $T(\ell')$ are all confined to the first quadrant, their digitizations contain only upward and rightward edges. Let ℓ_r and ℓ'_r be the time reversal of the directional reversal of ℓ and ℓ' respectively, and let R and R' be the regions of positive winding number with respect to $\ell \cup \ell'_r$ and $\ell_r \cup \ell'$. Then the digitization of R and R' are both empty if and only if ℓ is digitally equivalent to ℓ'. Similarly the digitizations of $T(R)$ and $T(R')$ can be used to resolve the digital equivalence of $T(\ell)$ and $T(\ell')$.

It only remains to show that the digitizations of R and R' are both empty if and only if the digitizations of $T(R)$ and $T(R')$ are. This is equivalent to saying that

$$(R \cup B_L(R)) \cap (\mathbb{Z}^2 + (\tfrac{1}{2}, \tfrac{1}{2})) = (R' \cup B_L(R')) \cap (\mathbb{Z}^2 + (\tfrac{1}{2}, \tfrac{1}{2})) = \emptyset$$

if and only if

$$(T(R) \cup B_L(T(R))) \cap (\mathbb{Z}^2 + (\tfrac{1}{2}, \tfrac{1}{2})) = (T(R') \cup B_L(T(R'))) \cap (\mathbb{Z}^2 + (\tfrac{1}{2}, \tfrac{1}{2})) = \emptyset,$$

which is true if $B_L(T(R)) = T(B_L(R))$ and $B_L(T(R')) = T(B_L(R'))$.

The limitations on the directions of ℓ and ℓ' ensure that ℓ, ℓ' $T(\ell)$, and $T(\ell')$ never have negative slope. Thus the following holds for $\delta_1, \delta_2 > 0$ and $\bar{R} \in \{R, R', T(R), T(R')\}$: In the neighborhood of some point z on the boundary of $\bar{R}$, either all points $z + (\delta_1, -\delta_2)$ lie in $\bar{R}$, or none of them lie in $\bar{R}$. When $\delta_1, \delta_2 > 0$, T maps points of the form $z+(\delta_1, -\delta_2)$ into points $T(z)+(\delta'_1, -\delta'_2)$ where $\delta'_1, \delta'_2 > 0$. Thus $z \in B_L(R)$ if and only if $T(z) \in B_L(T(R))$, and similarly for R'. ∎

We now consider what happens when the process of polygonization of digitized paths is carried to its logical conclusion. We shall not go into great detail, because the intent is merely to explain the how the ideas of Section 5.2 can be extended, and what the possible applications are. See also Ramer [22] for a completely different approach to the problem of finding a polygonal approximation to a digitized curve. This also has well known applications to image processing and scaling digitized shapes.

The approach described here is unique in that the polygonal approximation is a polygonal tracing that may contain backward moves. It is possible to remove the backward edges by introducing additional vertices, but the description involving backward moves is simpler and easier to compute. If the input is assumed to be the digitization of a smooth curve $\mathcal{S}$, the polygonal version can be constructed so as to determine the minimum possible number of points of inflection on $\mathcal{S}$. The backward moves in the polygonal version of $\mathcal{S}$ contain information about $\mathcal{S}$, and Section A.5 shows how to put this to good use.

Let $\mathcal{T}$ be a digital tracing and assume for simplicity that it is connected and has no backward moves. Let a *simple subtracing* of $\mathcal{T}$ be a digital tracing that is a subset of $\mathcal{T}$ and has the following properties: It must be a rotation of a simple positive digital tracing, and either its nonterminal horizontal edges or its nonterminal vertical edges must all have length 1. A simple subtracing of $\mathcal{T}$ that is not a subset of any other simple subtracing of $\mathcal{T}$ is a *maximal simple subtracing* of $\mathcal{T}$. Figure 48 shows a digital tracing and its maximal simple subtracings.

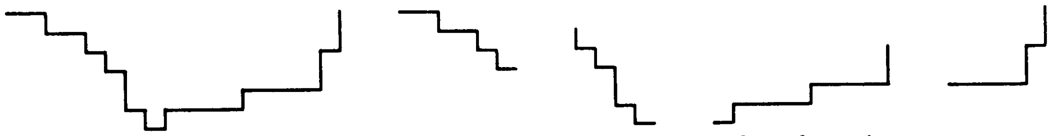
Fig. 48. A digital tracing and its maximal simple subtracings.

The concept of polygonization defined in Section 5.2 can easily be extended to maximal simple subtracings since they are just rotations of simple positive digital tracings. As usual we are free to adjust the lengths of the terminal edges of each subtracing before taking the polygonization. The only restriction on the lengths of these terminal edges is that the actual tracings being polygonized must be such that either all their vertical edges have length 1 or all their horizontal edges do.

The simple subtracings may be ordered according to their order of occurrence in a trip that corresponds to the original digital tracing. The polygonizations of the subtracings in this order define a polygonal tracing whose digitization is the original digital tracing: Each pair of consecutive polygonizations is connected by a single forward or backward edge so as to obtain the correct digitization, and similar edges must be added to connect the starting point of the trip to the first polygonization, and to connect the last polygonization to the ending point of the trip. Figure 49 shows the result of this process for the previous example. In this case, all the edges connecting the polygonizations of the simple subtracings are backward edges, but the edges that connect to the endpoints of the trip are forward edges. When this polygonal partial tracing is digitized, the backward edges cancel out edges that do not belong to the original digital path or would otherwise occur with weight 2. The circled intersections show points where vertices could be introduced in order to remove the backward moves if desired.

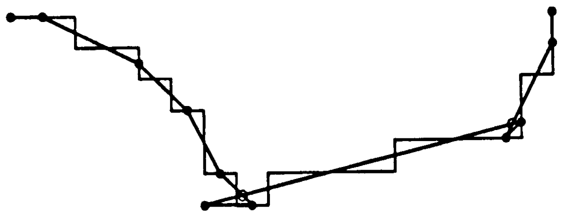
Fig. 49. A polygonal tracing and its digitization.

In the example of Figure 49 there is little to be gained from any further polygonization, but it is possible to do this by first rotating the polygonal path by a multiple of 90° and then applying T_k or U_k for some k. Any digital path that is a subset of the transformed path may be treated as above so as to obtain a polygonal version with the same digitization. It is a simple consequence of Lemma A.4.2 that the digitizations will still match when the transformations are reversed. An example of such multiple polygonizations is shown in Figure 50.

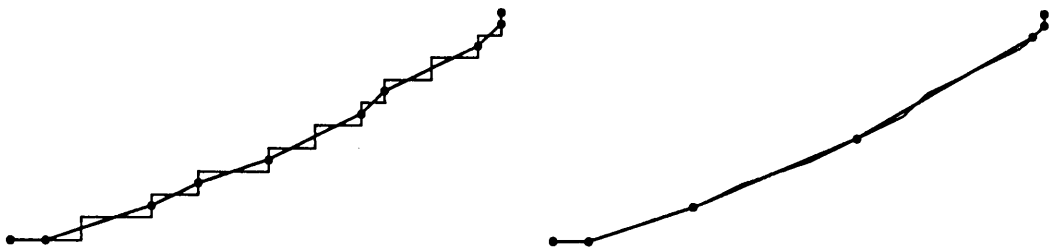
Fig. 50. A digital tracing and polygonal versions created by two polygonization steps.

The process of replacing segments of the polygonal path with transformed polygonizations can be continued as long as it is possible to find subtracings that can be transformed into nontrivial digital paths. Each such step removes points of inflection and results in a smoother polygonal path. Let us call the resulting polygonal path a *complete polygonization* of the original digital path. The complete polygonization is not unique because we have not given explicit stopping conditions and the result depends on what terminal segment length adjustments are done before polygonization. The best way to make such choices depends somewhat on the application, but the goal is usually to remove as many points of inflection as possible.

A.5. Smoothing through Complete Polygonization

Complete polygonization affords another approach to smoothing that does not require manipulating the trajectory. The smoothness of a digital tracing is closely related to the edge lengths in its complete polygonization. For instance, the backward edges in Figure 49 in directions $(1,0)$ and $(1,1)$ correspond to pimples.

We have already seen that we can avoid pimples and flat spots in a the digitization of a tracing $\mathcal{T}$ at a rational direction (u,v) by ensuring that a suitably transformed version of $\mathcal{T}$ has horizontal and vertical tangent points at integer coordinates. Without the transformation, these good tangent positions are separated by $1/r$ where $r = \sqrt{u^2+v^2}$ as shown in Figure 51. Dashed lines indicate equivalence classes of pixel centers.

The figure shows a preliminary stage in the computation of the complete polygonization of the digitization $\mathcal{D}(\mathcal{T})$ when there are edges of direction (u,v) near the tangent point A. We shall assume that, in the neighborhood of A, the tracing $\mathcal{T}$ can be approximated with a circle of curvature k (shown as a dotted line in the figure). Let C and D be points where this circle crosses (u,v)-directed lines containing pixel centers as shown in the figure.

Let us estimate the length of the (u,v)-directed edge in the complete polygonization of $\mathcal{D}(\mathcal{T})$ when the tangent point A is halfway between equivalence classes of pixel centers as shown in Figure 51. The edge adjacent to the (u,v) edge in the complete polygonization will fall very close to the line CD. If B is the point where

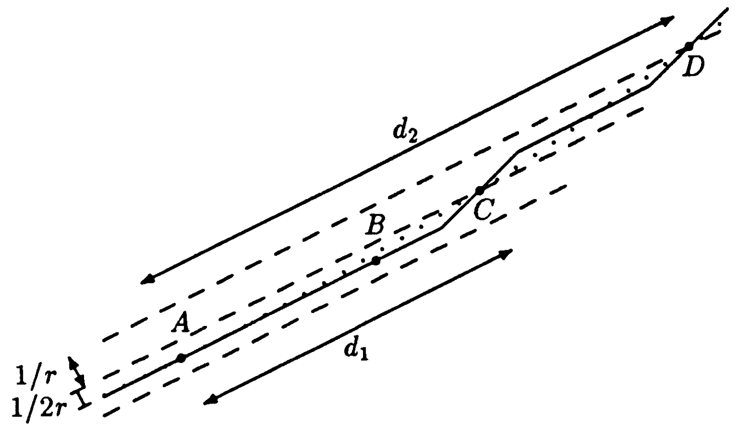

Fig. 51. A tracing $\mathcal{T}$ and a polygonized version well placed relative to pixel centers in direction (u, v).

the line CD intersects the tangent to $\mathcal{T}$ through A, the length of the (u, v) edge in the complete polygonization will be approximately twice the distance between A and B.

It is convenient to shift and rotate the coordinate system so that A becomes the origin and the tangent line through A becomes the x-axis. Thus the new coordinates for points B, C and D are $(\frac{3}{2}d_1 - \frac{1}{2}d_2, 0)$, $(d_1, \frac{1}{2}r^{-1})$, and $(d_2, \frac{3}{2}r^{-1})$, for some constants d_1 and d_2. The equation for the circle that approximates $\mathcal{T}$ is $x^2 + (y - 1/k)^2 = 1/k^2$ or $x = \sqrt{(2/k)y - y^2} = \sqrt{2y/k}(1 + O(ky))$. Thus up to a factor of $1 + O(k/r)$, $d_1 \approx 1/\sqrt{kr}$ and $d_2 \approx \sqrt{3/kr}$. The length estimate for the (u, v) edge of the complete polygonization is twice the distance between points A and B. This is $3d_1 - d_2$ or approximately r times

$$\frac{3 - \sqrt{3}}{r^{3/2}\sqrt{k}}. \tag{A.5.1}$$

We shall not discuss the actual smoothing process in detail, but the idea is to make some estimate of the curvature and then take the complete polygonization and adjust the edge lengths so as to approximate (A.5.1). The digitization of this new polygonal path is the smoothed version of the original digital path.

It is convenient to give edge lengths as *relative lengths* with a factor of r divided out because the edges in direction (u, v) in the complete polygonization have lengths that are multiples of $r/2$. Table 2 summarizes the relative edge lengths for polygonal tracings derived from one octant of a circle of diameter 401. The lengths predicted by (A.5.1) are compared to those in the polygonization of the digitization of the unmodified circle and in a similar polygonization after the techniques of Section 6.2 have been applied. The smoothed and unsmoothed digitizations are shown in Figure 52.

The unsmoothed path has a pimple at direction $(3, 1)$ and a slight flat spot at direction $(3, 2)$. After smoothing, the relative lengths are mostly very close to those given by (A.5.1). This indicates that the result of smoothing by adjusting edge lengths may be very similar to what is achieved by using linear transformations to adjust the curve before digitizing it. Figure 52 shows that, as mentioned in

Table 2. Relative edge lengths before and after smoothing.

	Relative lengths		
Direction	Eqn (A.5.1)	Before	After
(1,0)	17.95	19	18
(11,1)	.49	0	1.5
(10,1)	.56	1.5	0
(9,1)	.66	0	0
(8,1)	.78	0	0
(7,1)	.95	1	1
(6,1)	1.2	1	1
(5,1)	1.56	1	1
(4,1)	2.14	2.5	1.5
(7,2)	.91	0	1.5
(10,3)	.53	1.5	0
(3,1)	3.19	−2	3
(8,3)	.72	1.5	0
(5,2)	1.44	.5	1
(7,3)	.85	1.5	1.5
(2,1)	5.37	3.5	4.5
(7,4)	.78	1.5	1.5
(5,3)	1.28	0	1
(3,2)	2.62	4.5	2.5
(4,3)	1.61	1	1
(5,4)	1.11	1	1
(6,5)	.82	1.5	0
(7,6)	.64	0	1.5
(1,1)	10.68	12.5	11.5

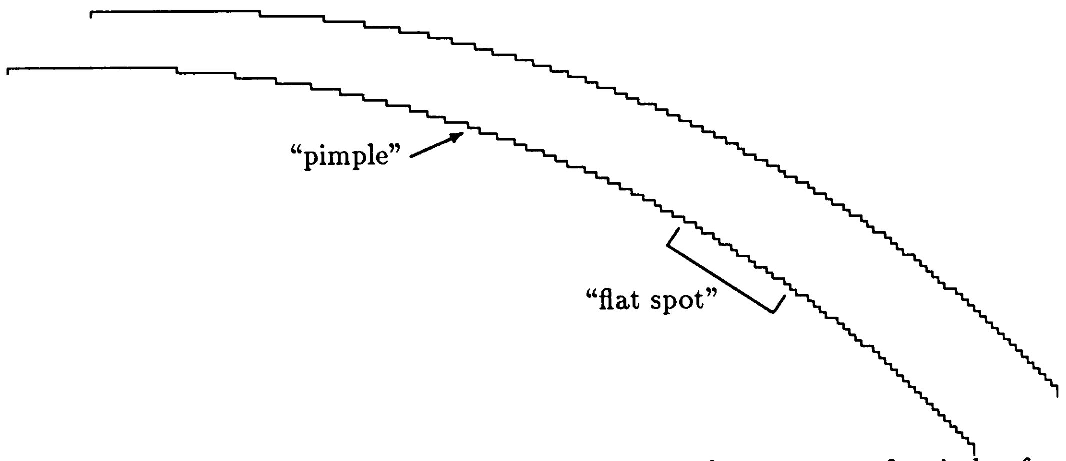

Fig. 52. Smoothed and unsmoothed digitizations of one octant of a circle of diameter 401.

Section 6.2, pimples and flat spots in rational directions other than $(0,\pm1)$, $(\pm1,0)$, and $(\pm1,\pm1)$ tend to be rather subtle. Section A.6 describes one application where such smoothing is important.

A.6. Data Compression

The data compression application depends on the fact that the complete polygonization is a polygonal path with edges of easily describable lengths and simple rational slopes. Consider the polygonization of a maximal simple subtracing $\mathcal{T}$ and assume without loss of generality that the vertical edges all have unit length. Since nonterminal vertices of the polygonization can occur only in the middle of vertical edges of edges of $\mathcal{T}$, interior edges of the polygonization all have integral relative length.

If the result of such a polygonization contains no points of inflection, then the complete polygonization will contain a string of adjacent edges whose directions are of the form

$$\begin{aligned}\ldots\quad &(3p+p',\,3q+q'),\quad (2p+p',\,2q+q'),\\ &(p+p',\,q+q'),\quad (p+2p',\,q+2q'),\quad (p+3p',\,q+3q'),\quad \ldots \qquad (A.6.1)\end{aligned}$$

When further levels of polygonization are required, the effect will be to insert similar sequences of edges between existing edges. For instance, a sequence of the form

$$\begin{aligned}\ldots,\quad &(2(2p+p')+(p+p'),\,2(2q+q')+(q+q')),\\ &((2p+p')+(p+p'),\,(2q+q')+(q+q')),\\ &((2p+p')+2(p+p'),\,(2q+q')+2(q+q')),\quad \ldots \qquad (A.6.2)\end{aligned}$$

may be inserted between $(2p+p', 2q+q')$ and $(p+p', q+q')$. When there is a point of inflection, we turn around and start taking the directions in reverse order.

We can encode a sequence of relative lengths and the corresponding directions by just listing the relative lengths for each sequence of directions similar to (A.6.1) and using special commands **down**, **up**, and **inflection** to help determine the directions. The **down** command takes a numeric argument k and starts a new sequence similar to (A.6.2). The parameter k determines where the sequence starts: If the current direction is (p,q) and the next direction would be (p',q') if there were no **down** command, then the next direction is $(k+1)(p,q)+(p',q')$ if $k \geq 0$, and $(p,q)+(1-k)(p',q')$ if $k \leq 0$. A sequence started with a **down** command is terminated by an **up** command. An **inflection** command causes previous directions to be repeated by reversing the order in which each sequence is traversed.

It is possible to deduce from the directions of the surrounding edges whether the relative length of any particular edge should be 0 or $\frac{1}{2}$ (modulo 1). We can take advantage of this by replacing non-integer relative lengths l by the integer $l-\frac{1}{2}$. Thus the unsmoothed polygonal path described by Table 2 can be represented

19 **down**(9) 1 0 0 1 1 1 2 **down**(-1) 1 **up** -2
down(1) 1 0 1 **up** 3 **down**(2) 1 0 4 1 1 1 **up** 12.

Similarly the smoothed version is

18 **down**(10) 1 0 0 0 1 1 1 1 **down**(0) 1 **up** 3
down(0) 1 1 **up** 4 **down**(2) 1 1 2 1 1 0 1 **up** 11.

Of course we need another three bits to give the starting direction (up, down, left, or right) and the initial direction of curvature. Both of these polygonal paths represent one octant of a circle of diameter 401. In each case the total length of the corresponding digital path shown in Figure 52 is 235.

The above representation can be encoded into a series of 4-bit nibbles so that a typical polygonal path requires an average of about 7 bits per segment. For paths like the one in our example, this amounts to about .6 bits per unit of the original digital path. At higher resolutions the segments of the polygonal path are longer and the encoding is more efficient. The results are roughly comparable with those of Hobby and Gu [10].

The encoding just described leaves much room for improvement. Perhaps the most obvious inefficiency is that an **up** command is almost always followed by a simple number and then a **down** command. This would allow one nibble to be saved for almost every **up** command.

A much larger inefficiency is that the numbers in the encoding are highly predictable, especially if the polygonal path has been smoothed as described in the previous section. The relative lengths will be very close to the values given by (A.5.1), and the argument to each **down** command is closely related to the immediately preceding relative length. If the preceding segment has direction (p,q) and relative length l, then the argument to the **down** command is approximately

$$\frac{l}{\sqrt{3}} - 1 - \frac{p'p + q'q}{p^2 + q^2}$$

where (p',q') is what the direction of the next segment would have been if there were no **down** command. (This can be derived by an argument similar to that used to derive (A.5.1) in Section A.4.) Taking full advantage of this would require a complicated encoding scheme for the numeric arguments, but this may allow smoothed polygonal tracings to be encoded with as few as 2 bits per segment.

A.7. Curve Fitting

Suppose we are given a digital tracing and we need to find a spline curve whose digitization matches this. For simplicity, we shall consider Bézier splines of order 2, but the techniques outlined here can be adapted to parametrically defined splines of other orders. This problem is quite difficult because it is necessary to choose the endpoints of each spline segment and ensure that the slope is continuous whenever two spline segments meet. If these knots and the directions there are fixed in advance, then the problem becomes much easier because each spline segment can be treated separately. Here we outline an approach that requires only the directions to be fixed in advance.

We begin by taking the complete polygonization of the given digital tracing $\mathcal{S}$, and choosing edges of the resulting polygonal partial tracing $\mathcal{T}$ near which we want to place spline knots (x_i, y_i). The spline direction at each such knot must match the direction (p_i, q_i) of the edge associated with it, but the knots may be repositioned.

The spline will be restricted to lie in a polygonal region $R(\mathcal{T})$ near $\mathcal{T}$ containing a strip of width $1/\sqrt{p^2+q^2}$ around each edge of $\mathcal{T}$ with reduced rational direction (p,q). This will be contained in the digitization region of the original digital tracing as shown in Figure 53.

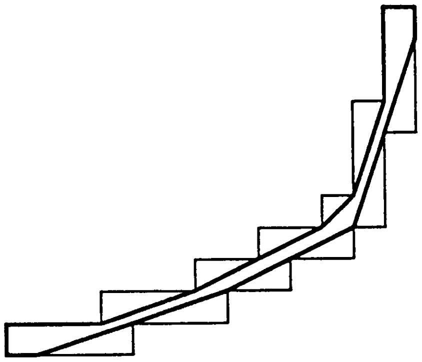
Fig. 53. The region $R(\mathcal{T})$ and the digitization region of $\mathcal{T}$.

We start with an initial estimate of where the knots should be placed. For each pair of adjacent knots (x_i, y_i) and (x_{i+1}, y_{i+1}), the control point $(\bar{x}_i, \bar{y}_i)$ should be such that $(\bar{x}_i, \bar{y}_i) - (x_i, y_i)$ is a positive multiple of the direction (p_i, q_i), and $(x_{i+1}, y_{i+1}) - (\bar{x}_i, \bar{y}_i)$ is a positive multiple of (p_{i+1}, q_{i+1}). If there is no such point, we must revise our initial choice of directions and start over. Otherwise the three points (x_i, y_i), $(\bar{x}_i, \bar{y}_i)$, and (x_{i+1}, y_{i+1}) are the control points of the Bézier quadratic spline segment between (x_i, y_i) and (x_{i+1}, y_{i+1}).

If the spline chosen so far does not have the correct digitization, we refine it as follows: For each segment s of $\mathcal{T}$, we find an appropriate point c on the spline where the direction matches that of s as shown in Figure 54. This is a point of closest approach between the spline and one of the edges of $R(\mathcal{T})$. Let t_c be the value of the parameter at which c occurs. We can write down an inequality stating that the t_c point on the spline is on the desired side of the bound line, and this equation is linear in the coordinates of the Bézier control points.

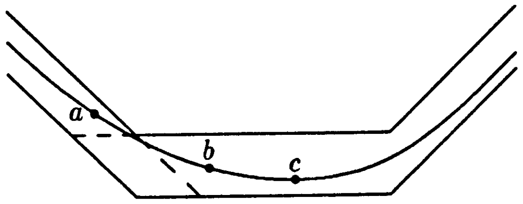

Fig. 54. The selection of constraint points for use in curve fitting.

We also find two points a and b bracketing each inside corner of the polygonal region. Originally, these will coincide with the surrounding c points, but as further refinements are made, these points will approach each other. We can write two inequalities that must be satisfied whenever some portion of the spline between a and b passes through parallelogram formed by the bound lines at the corner in question. (See the dashed lines in Figure 54.) In the case shown in the figure, this amounts to saying that the t_a point on the spline is to the left of the rightmost line of slope -1 shown, and the t_b point on the spline is below the upper horizontal line.

There are also other inequalities that constrain the behavior of the spline near segments where the complete polygonization has a point of inflection, but in the interest of brevity we shall omit these.

Suppose the t_c values are accurate estimates of when the closest approaches to the bound lines occur in subsequent refinements and the t_a and t_b values are accurate estimates of when the spline either passes through the parallelograms just mentioned or leaves the polygonal region by crossing on the wrong side of the corresponding corner. Then the inequalities mentioned so far are a good estimate of whether or not subsequent refinements of the curve remain in the polygonal region as desired. Furthermore, all these inequalities are linear in the coordinates of the Bézier control points.

We now add linear equations that force the spline segments to have the required rational directions where they join at knots:

$$q_i(\bar{x}_i - x_i) - p_i(\bar{y}_i - y_i) = q_i(x_i - \bar{x}_{i-1}) - p_i(y_i - \bar{y}_{i-1}) = 0.$$

We have developed a system of linear equalities and inequalities in the coordinates of the control points. The inequalities are conservative in the sense that they allow sets of control points that define splines that do not stay within the required polygonal region, but any solution to our curve fitting problem should satisfy them. We can use linear programming to find a solution to the equations and thus a revised estimate for the positions of the knots.

The algorithm consists of successively revising estimates for the positions of the knots until either the spline produced is confined to the required polygonal region, or we generate an infeasible linear programming problem. In the latter case we must try again with more knots. After each iteration we refine the inequalities by bringing the a and b points closer together. The separation between each t_a and t_b should reflect our uncertainty about when the curve should make its closest approach to the appropriate corner of the polygonal region.

There is no guarantee that this algorithm will always find a parametric spline with the required digitization when one exists, and the algorithm remains to be tested in practice, but it should illustrate the possible application of complete polygonizations to curve fitting: The polygonization gives good estimates for the position and direction of the required curve.

Index to Terminology

References

1. J. R. Armstrong, "Design of a graphic generator for remote terminal application," *IEEE Transactions on Computers*, **C-22**, May 1973, pp. 464–468.
2. K. Belsner, "Comment on an improved algorithm for the generation of nonparametric curves," *IEEE Transactions on Computers*, **C-25**, January 1976, p. 103.
3. J. E. Bresenham, "Algorithm for computer control of a digital Plotter," *IBM Systems Journal*, **4**, 1965, pp. 25–30.
4. J. E. Bresenham, "A linear algorithm for incremental digital display of circular arcs," *Communications of the ACM*, **20**, February 1977, pp. 100–106.
5. Daniel E. Field, *Algorithms for Drawing Simple Geometric Objects on Raster Devices*, Ph.D. Thesis, Princeton University, June, 1983.
6. Kenneth P. Fishkin and Brian A. Barsky, "Algorithms for brush movement in paint systems," *Graphic Interface '84*, pp. 9–16.
7. James D. Foley and Andries Van Dam, *Fundamentals of Interactive Computer Graphics*, Addison-Wesley, Reading, Massachusetts, 1982.
8. Leo Guibas, Lyle Ramshaw, and Jorge Stolfi, "A kinetic framework for computational geometry," *Proceedings of the 24th annual Symposium on the Theory of Computing*, 1983, pp. 100–111.
9. John D. Hobby, "Smooth, Easy to Compute Interpolating Splines," to appear.
10. John D. Hobby and Gu Guoan, "Using string matching to compress Chinese characters," *Proceedings of the 1982 International Conference of the Chinese-Language Computer Society*, September 1982, pp. 56–70.
11. B. K. P. Horn, "Circle generators for display devices," *Computer Graphics and Image Processing*, **5**, 1976, pp. 280–288.
12. B. W. Jordan, W. J. Lennon, and B. C. Holm, "An improved algorithm for the generation of non-parametric curves," *IEEE Transactions*, **C-22**, December 1973, pp. 1052–1060.
13. Donald E. Knuth, *TEX and METAFONT, New Directions in Typesetting*, American Mathematical Society and Digital Press, Providence, Rhode Island, 1979.
14. Donald E. Knuth, *The Art of Computer Programming*, **2**, Addison-Wesley, Reading, Massachusetts, 1981.
15. Donald E. Knuth, *Computers and Typesetting*, Vol. 2: METAFONT, Addison-Wesley, Reading, Massachusetts, to appear.
16. Bruce D. Lucas, *Drawing Lines and Curves on Raster Devices*, IBM Research Report RC 9648 (#42610), October 1982.
17. A. Lindgård, R. Moss, "Parametric spline curves in integer arithmetic designed for use in computer controlled plotters," *Computers and Graphics*, **4**, 1979, pp. 51–61.
18. F. Mertens, "Über einige asymtotische Gesetze der Zahlentheorie," *Journal für die reine und angewandte Mathematik* **77** (1874), pp. 289–291.
19. William M. Newman and Robert F. Sproull, *Principles of Interactive Computer Graphics*, McGraw-Hill, New York, 1979.

20. Charles S. Peirce, *The New Elements of Mathematics,* Mouton, The Hague, 1976.
21. Vaughan Pratt, "Techniques for conic splines," *SIGGRAPH '85,* pp. 151–159.
22. Urs Ramer, "An iterative procedure for the polygonal approximation of plane curves," *Computer Graphics and Image Processing,* **1**, 1972, pp. 244–256.
23. J. Ramot, "Nonparametric Curves," *IEEE Transactions on Computers,* **C-25**, January 1976, pp. 103–104.
24. Lyle Ramshaw, private communication, 1984.
25. Jerome Rothstein and Carl Weiman, "Parallel and sequential specification of a context sensitive language for straight lines on grids," *Computer Graphics and Image Processing,* **5**, 1976, pp. 106–124.
26. Thomas Spencer, unpublished manuscript, 1981.
27. R. F. Sproull, "Using program transformations to derive line drawing algorithms," *ACM Transactions on Graphics,* **1**, October 1982, pp. 259–273.
28. M. A. Stern, "Über eine Zahlentheoretische Funktion," *Journal für die reine und angewandte Mathematik* **55** (1858), pp. 193–220.