An Efficient Filling Algorithm for Non-simple Closed Curves using the Point Containment Paradigm

ANTONIO ELIAS FABRIS[1] LUCIANO SILVA[1] A. ROBIN FORREST[2]

[1] Instituto de Matemática e Estatística, Universidade de São Paulo
Caixa Postal 66281, 05315-970, São Paulo – SP, Brazil
aef,lucianos@ime.usp.br

[2] School of Information Systems, University of East Anglia
Norwich, NR4 7TJ, U.K.
forrest@sys.uea.ac.uk

Abstract. The Point Containment predicate which specifies if a point is part of a mathematically defined shape or not is one of the most basic notions in raster graphics. This paper presents a technique to counteract the main disadvantage of Point Containment algorithms: their quadratic time complexity with increasing resolution. The implemented algorithm handles complex geometries such as self-intersecting closed curves.

1 Introduction

The Point Containment paradigm is a natural way to perform raster graphics operations such as filling and stroking. However, Point Containment-based algorithms generally have been judge to be too slow [14]. In [9], Forrest expressed the need for an eficient and robust algorithm to decide if a given point is part of a mathematically defined shape. Such an algorithm was later developed by Corthout et al. [2, 3, 4, 5] and implemented in dedicated silicon, the Pharos chip fabricated by Philips, on support of the PostScript page description language.

The most widespread approach to filling is scan conversion but even for simple concave polygons known scan conversion algorithms require a computationally expensive presorting and marking phase before they can compute the actual intervals of points contained in the region. The latter phase – specially for curved boundaries – requires careful attention to both geometric and numeric detail in order to provide robust algorithms. Conventional filling of curved regions is based on algorithms for scan conversion of polygons where the end points of spans are incrementally updated and pixels in between are filled. At some stage the intersection formula derived in the continuous plane and usually computed using real arithmetic has to be mapped to the discrete plane. This mapping necessitates an implicit or explicit epsilon-test that may cause incorret results [17, 8, 2].

Brooks [1] quoting Poulton points out that if current hardware trends continue, the number of pixels per primitive rendered by hardware will approach unity, and in such circumstances we might as well compute pixels directly from the underlying geometry rather than first approximating the geometry by polygons or line segments. The point containment approach is an example of this strategy, generating pixels directly from curves and thus avoiding the difficulties in curve rendering tackled by Klassen [12] and Lien et al. [13]. In Corthout–Pol point containment technique [2, 3, 4, 5], the problems of discretisation are acknowledged and tackled by casting the problem to be solved as a discrete integer problem from the outset rather than attempting to accommodate all the problems of numerical accuracy which follow from floating point discretisation, geometric approximation, or a less rigorous approach to discretisation later in the rendering process. Corthout and Pol's thesis contains details of the overall integer precision required for rendering on a chosen device together with proofs of robustness and accuracy. Furthermore, the method lends itself to hardware implementation: the Pharos chip implements the point containment algorithm in full and could be used serially or in a variety of parallel configurations. The algorithm is fast on parallel hardware using Pharos chips.

The major disadvantage of the Point Containment approach, even from the hardware implementation point of view, is its quadratic behaviour with respect to resolution. In [5], Corthout and Pol describe a way to reduce the time complexity of the Point Containment approach to quasi–linear. This paper presents a method whose time complexity depends not on the resolution but only on the perimeter of the polygon boundary.

2 The mathematical structure

In [5], Corthout and Pol state and prove a version of the Jordan curve theorem for regions bounded by non-simple discrete curves which is the basis of their point containment technique. To enunciate this theorem, we describe the basic mathematical structure built on the discrete plane $\mathbb{Z}^2$, which will also be used to develop the fast

point containment algorithm presented in this paper.

Building on the discrete plane, the notion of list will be used to represent the vertices of a polygon as well as the control points of a Bézier curve. A *list of length* n is an element of the set

$$\Lambda_n = \{L : [0..n] \to \mathbb{Z}^2\}.$$

A list L is called a *closed list*, or alternatively a polygon, iff $L(0) = L(n)$. The set of all lists will be denoted by Λ and the set of closed lists by Λ_c. Let d_8 be the metric on $\mathbb{Z}^2 \times \mathbb{Z}^2$ defined by

$$d_8\,(x, y) = \max\{|x_1 - x_2|, |y_1 - y_2|\}\ .$$

A list L is called 8-connected iff the d_8 distance between any two consecutive points is at most 1. Using the well known metrics d_4 and d_6 [15, 5] we can define respectively 4- and 6-connected lists. A region R in $\mathbb{Z}^2$ is m-connected ($m \in \{4, 6, 8\}$) iff for all x,$y \in R$, there exists a m-connected list L embedded on R such that $L(0) = x$ and $L(n) = y$.

The point containment algorithm must detect, for each given query point and outline, whether it is contained in the region enclosed by the outline or not. This detection is based on the concept of *winding number*. There are two ways to define the winding number: the analytic version employs a complex line integral and the geometric version counts the number of direct intersection with a ray.

For the geometric version, let $\varepsilon \in \mathbb{R}^2$ be given and consider the function $W_\varepsilon : \Lambda \times \Lambda \to \mathbb{Z}$ defined by:

- $W_\varepsilon : \Lambda_0 \times \Lambda \cup \Lambda \times \Lambda_0 \to \mathbb{Z}, W_\varepsilon(L^1, L^2) = 0$
- $W_\varepsilon : \Lambda_1 \times \Lambda_1 \to \mathbb{Z}$,

$$\begin{cases} \varphi(L^1) \cap \varphi_\varepsilon(L^2) = \emptyset \Rightarrow W_\varepsilon(L^1, L^2) = 0 \\ \varphi(L^1) \cap \varphi_\varepsilon(L^2) \neq \emptyset \Rightarrow W_\varepsilon(L^1, L^2) = \\ signal((\Delta(L^1) \times \Delta(L^2))^z) \end{cases}$$

  where $(\Delta(L^1) \times \Delta(L^2))^z$ denotes the z-coordinate of the cross product between the vectors $\Delta(L^1)$ and $\Delta(L^2)$ (embedding in $\mathbb{R}^3$ is implicit), $\varphi(L)$ is the usual embedding in $\mathbb{R}^2$ that represents the line segment $\overline{L(0)L(1)}$ and $\varphi_\varepsilon(L) = \varphi(L) + \varepsilon$.

- $W_\varepsilon : \Lambda_{n_1} \times \Lambda_{n_2} \to \mathbb{Z}$ for $n_1 > 1$ or $n_2 > 1$ by: $W_\varepsilon(L^1, L^2) = \sum_{i=0}^{n_1-1} \sum_{j=0}^{n_2-1} W_\varepsilon(< L^1_i, L^1_{i+1} >, < L^2_j, L^2_{j+1} >)$

Taking ε as a pair $\varepsilon = (q, r)$ or $\varepsilon = (r, q)$, where $q \in \mathbb{Q} - \mathbb{Z}$ and $r \in \mathbb{R} - \mathbb{Q}$, Corthout and Pol show that $\varepsilon \notin \varphi(L)$ for all $L \in \Lambda_1$. Under this restriction on ε they prove that

$$W_\varepsilon(L^1, L^2) = \frac{1}{2\pi i} \int_{\varphi(L^1)} \frac{dz}{z - \varphi_\varepsilon(L^2_{n_2})}$$

$$-\frac{1}{2\pi i} \int_{\varphi(L^1)} \frac{dz}{z - \varphi_\varepsilon(L^2_0)}$$

and

$$W_\varepsilon(L^2, L^1) = \frac{1}{2\pi i} \int_{\varphi_\varepsilon(L^1)} \frac{dz}{z - \varphi(L^2_{n_2})}$$

$$-\frac{1}{2\pi i} \int_{\varphi_\varepsilon(L^1)} \frac{dz}{z - \varphi(L^2_0)}$$

where L^1 is a closed list[1]. With ε as defined above, W_ε is called *cross-weight function*. Note that, differently than the analytic version, the geometric version is more suitable to implementation.

With this linking between the geometric and the analytic version of the winding number, Corthout and Pol state and prove the discrete version of the Jordan curve theorem for non-simple closed curves.

Theorem 2.1 (Corthout–Pol) *Let ρ be an angle with irrational tangent. Given any closed list L^1, the cross weight function W_ρ divides the plane $\mathbb{Z}^2$ into a finite number of regions with points P of equal winding numbers*

$$\frac{1}{2\pi i} \int_{\varphi(L^1)} \frac{dz}{z - \varphi_\rho(P)}.$$

Precisely, one of these regions is infinite, and that region contains points with zero winding number. Furthermore, when for any list L^2 of length n_2 we have

$$\frac{1}{2\pi i} \int_{\varphi(L^1)} \frac{dz}{z - \varphi_\rho(L^2_0)} \neq \frac{1}{2\pi i} \int_{\varphi(L^1)} \frac{dz}{z - \varphi_\rho(L^2_{n_2})},$$

we must have $\varphi(L^1) \cap \varphi(L^2) \neq \emptyset$.

The last part of this theorem states that under certain conditions, the polygonal embeddings of two lists must have a point in commom. The Corthout–Pol Point Containment technique is based on this result.

3 Filling

In this section, we will describe a specific rasterizing function based on the cross weight W_ρ to fill the interior of polygons and discrete Bézier closed curves.

Consider the functions

$$F_\rho(L, P) = \lim_{Q \to \infty} W_\rho(L, < Q, P >).$$

and let the rasterizing function F be defined by

$$F = \lim_{\rho \uparrow 0} F_\rho.$$

Taking the ρ size infinitesimally small minimizes the effect of the ρ translation [5].

As F distributes over concatenations, the first step of its implementation is to sum over the polygonal sides of the list argument. The algorithm for this is:

[1] Here $\mathbb{R}^2$ is embedded in $\mathbb{C}$ for the evaluation of the complex line integral.

```
winding_number(List L, Point P)

1. sum<-0;
2. For each position i of L (not including the
    last element)
3.  sum<-sum+Contribution(L[i]-P,L[i+1]-P);
4. return sum;
```

The step (3) computes the contribution of each line segment to the winding number. The translation of each element in the list is to speed up the evaluation of the Contribution procedure whose goal is to compute each term on the decomposition of $F(L,P)$. The elements used to compute the contribution of each line segment are represented in the Figure 1.

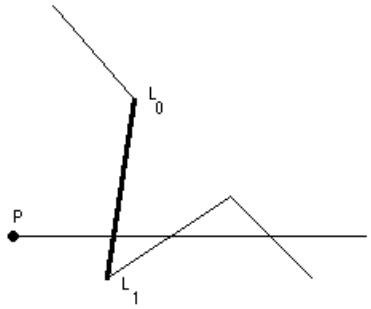


Figure 1: The elements of the contribution procedure

To compute the value of each term on the decomposition of $F(L,P)$ we use the following remarks [5]:

(R1) If $(P^y > max\{L_i^y\} \vee P^y \leq min\{L_i^y\} \vee P^y < min\{L_i^x\})$ then $F(L,P) = 0$.

(R2) If $(P^y < max\{L_i^y\} \wedge P^y > min\{L_i^y\} \wedge P^x \geq max\{L_i^x\})$ then $F(L,P) = signal(L_0^y - L_1^y)$.

(R3) Let $min\{L_i^y\} < P^y \leq max\{L_i^y\}$ and $min\{L_i^x\} \leq P^x < max\{L_i^x\}$. Let l be the line through the line segment L.

- If l is parallel to the x-axis, then $F(L,P) = 0$.
- If l is not parallel to the x-axis, then $\Delta(L)^y \neq 0$. Let v be the vector perpendicular to l, such that $v^x > 0$: $v = +-(\Delta(L)^y, -\Delta(L)^x)$. Let $c = L_0 \cdot v$, thus l is described by $(x,y) \cdot v = c$. We have:

$$P \cdot v \geq c \Rightarrow F(L,P) = signal(L_0^y - L_1^y)$$
$$P \cdot v < c \Rightarrow F(L,P) = 0$$

The Contribution procedure implements these remarks which evaluates the contribution of each line segment to the winding number:

```
Contribution(L0,L1)

1. Compute the values
      minx = min(L0.x,L1.x)
      maxx = max(L0.x,L1.x)
      miny = min(L0.y,L1.y)
      maxy = max(L0.y,L1.y)
2. If minx>0 or miny>=0 or maxy<0
      return 0    ( According to remark (R1))
3. Compute s=signal(L0.y - L1.y)
4. If maxx<=0
     return s     ( According to remark (R2))
5. Compute the vector v
     v.x = L0.y - L1.y
     v.y = L1.x - L0.x
6. If v.x=0
     return 0     ( According to remark (R3),
                    first condition)
7. If v.x<0
     v.x = -norm.x
     v.y = -norm.y
8. If InnerProduct(L0,v)>0 (Second condition
                             of the remark (R3))
      return 0
9. return s
```

Bézier curves can be incorporated on the case of polygonal lists, first converting the curve into a polygonal list, and evaluating the rasterizing function on this list. As one of the essential aspects of the Point Containment paradigm is the exclusive use of integer arithmetic, both Bézier curve and the polygon into which the curve is converted must be described with integers. But, numerical errors may arise due to this polygonalisation. To avoid this, Corthout and Pol [5] describe the notion of discrete Bézier curves in a discrete model space which is of higher precision than device space. Based on this, they derive a recursive algorithm to compute the curve winding number based on the corresponding control points.

4 Coherence with Point Containment

As we have remarked earlier, for each point in the image its winding number with respect to the given outline needs to be executed. Thus, if we have an image of resolution n, the time complexity is $O(n^2).c_{wn}$, where c_{wn} is the cost of the interior/exterior test, that is, the Point Containment has quadratic behaviour with respect to resolution.

There is a way to reduce the time complexity of the Point Containment to almost linear behaviour using a technique called *coherence testing*. Let L a closed list and F a rasterizing function be given. A region R is called *coherent with respect to the filled outline* L iff R is a subset of a single class of the equivalence relation on $\mathbb{Z}^2$ induced by F.

Note that if in the coherence testing a region is found, it is only necessary to evaluate the winding number for just one element inside the region.

The problem is: how to find these regions? Corthout and Pol [5] describe a general method to detect coherence with filling:

Theorem 4.1 (Corthout–Pol) *Let* $(m_1, m_2) \in \{(4,4), (4,6), (4,8), (6,4), (6,6), (8,4)\}$. *If* L *is a closed,* m_1*-connected list, and* T *is a* m_2*-connected region, then* $P \notin \{L_i\} \oplus T \Rightarrow P - T$ *is coherent with respect to* L.

The symbol $\oplus$ denotes the usual Minkowski addition (e.g. see [10], [16]). This theorem states that the coherence of the tile $P-T$ is implied when a single Point Containment test with a stroked outline returns negative. In Figure 2, regions indexed by P_0 and P_2 are filling–coherent, but the region P_1 is not. Note that the $\Leftarrow$ assertion is not true.

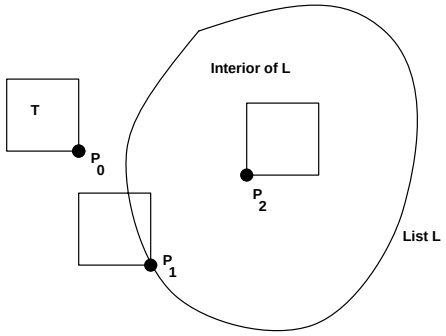


Figure 2: Coherence tests for filling

For this, take the tail $L = L_1 \uplus R(L_1)$; since a filled tail contains no points, any region $P - T$ is coherent, while $\{L_i\} \oplus T$ need not be empty.

4.1 Time complexity with coherence

Following the above methods, we can derive an algorithm for filling curves with coherence.

When an image is to be generated, first we detect whether the entire image is coherent, and if so, what colour it has. If not, the image is subdivided into four quadrants, and applied the described procedure recursively to each of the four quadrants. Note that this procedure always finishes, because a point is coherent. The stopping points of this algorithm are coherent regions.

The recursive structure of this algorithm creates a *quadtree*. Hunter and Steiglitz [11] show that the number of nodes in the quadtree is of order $O(p+n)$, where p is the perimeter of the contour, and n is the maximum subdivison level. As the number of Point Containment tests needed to build the tree depends linearly on the number of nodes, the number of tests is also of order $O(p+n)$, that is, if coherence is used, the algorithm has almost linear time complexity.

5 The new approach to filling coherence

Note that the approach of quadtrees to detect coherent regions is not optimal, because it finds squared coherent regions that can be frequently embedded into greater regions. For example, in Figure 3 the white leaves could be embedded into a 2×2 square as they have the same winding number.

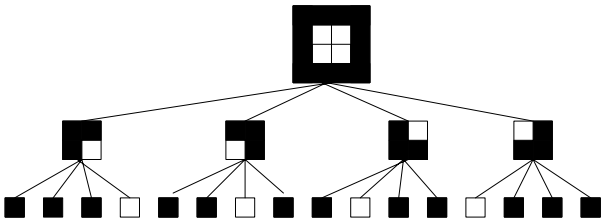


Figure 3: Decomposition to find coherence

Our approach is to detect an element of each coherent regions and propagate its colour to the whole region. The algorithm uses two auxiliar procedures: *Detection* and *Propagation*.

The Detection procedure tries to find a point of interior of L:

```
Boolean Detection(Point P,List L, Point Q)

1. If( P has a neighbour V belonging
     to the interior of L, with
     diferent from INTERIOR_COLOUR)
2.    Q <- V
3.    return TRUE
4.  else return FALSE
```

Given a point Q belonging to interior of L, the Propagation procedure assigns the INTERIOR_COLOUR to all points on the coherent region that contains Q:

```
Propagation(Point Q)

1.   Queue <- EMPTY;
2.   enqueue(Q,Queue);
3.   Colour(Q)<- INTERIOR_COLOUR;
3.   While(Queue is not EMPTY)
4.      P <- dequeue(Queue);
5.   For all neighbour T of P that
     has not colour INTERIOR_COLOUR
6.     Colour(T)<-INTERIOR_COLOUR
7.     enqueue(T,Queue);
```

Finally, the Filling_with_coherence procedure fills L by calling the Detection procedure to find a coherent region and, if such a region exists, it fills the region using the Propagation procedure :

```
Filling_with_coherence(List L)

1. For each point P in the image
     Colour(P) <- EXTERIOR_COLOUR
2. Embedd the points of L on the image, setting
   their colours to INTERIOR_COLOUR
3. For each point P in L
4.   If Detection(P,L,Q) is TRUE
5.       Propagation(P,L,Q)
```

The following results analyses the steps of the proposed algorithm. Firstly, we introduce some notation. The *interior* of a closed list L will be denoted by $int(L)$. Let W be a m-connected region. The *boundary* of L will be denoted by $\partial W = \{P \in W | \exists Q \notin W, d(P,Q) = 1\}$, where d is the metric associated to W. A *path* (or a $\mathbb{Z}^2$*–path*) between two points P and Q(the path ends) is a sequence $\{V_1 = P, V_2, \cdots, V_{n-1}, V_n = Q\}$, where $d(V_i, V_{i+1}) = 1$ for all $i = 1, ..., n-1$. The points $V_i (i \neq 1, n)$ are called internals. Note that, if W is a m-connected region then always there exists a path linking P and Q for all $P, Q \in W$.

Lemma 5.1 *Let L be a closed list and $P \in int(L)$. Then there exists a path with end points P and Q ($Q \in L$), whose internal points are contained in $int(L)$.*

Proof: If $int(L) = \emptyset$, then the result holds. If $int(L) \neq \emptyset$, according to the Theorem 2.1, $P \in W \subset int(L)$, $|W|$ (the cardinality of the set W) finite and W is m-connected, for any $m \in \{4, 6, 8\}$. W is limited by a closed sublist $L^1 \neq \emptyset$ of L, and more, $\forall T \in L^1$ exists $S \in W$ such that $d(T, S) = 1$. It is clear that $W \cup L^1$ is m-connected. Thus, there exists a $\mathbb{Z}^2$-path $\{V_1, ..., V_n\}$ linking P to any point $Q \in L^1$. If $n = 2$, there are no internal vertices and then the lemma holds. If $n > 2$, consider the set $I = \{V_1, ..., V_n\} \cap int(L)$. It is clear that it can be choosen a subset $I' = \{V_{i_1}, ..., V_{i_k}\}$ of I, such that $V_{i_k} = P$, $d(V_{i_l}, V_{i_{l+1}}) = 1, l = 1, ..., k-1$ and consequently there exists a point $Q \in L$, $d(Q, V_{i_1}) = 1$. Thus, the path of the lemma is $\{Q, V_{i_1}, ..., V_{i_k} = P\}$.

To introduce the next lemma we need the following definition:

Definition 5.1 *Let W be a limited region of $\mathbb{Z}^2$ and L be a closed list. W is called maximal connected coherent if W is m-connected, $w(L, P) = w(L, Q)$ for all $P, Q \in W$ and if $V \subset \mathbb{Z}^2$ satisfies:*

- $W \subseteq V$
- V *m-connected*
- $\exists T \in V$,$\exists S \in W$, *such that* $w(L, T) = w(L, S)$

then $V = W$.

Lemma 5.2 *If a point Q is detected by the Detection procedure, then the Propagation procedure assigns INTERIOR_COLOUR to all points $P \in W$, where W is a maximal connected coherent region that contais Q.*

Proof: Suppose that the Detection procedure found a point Q. According to the theorem 2.1: $Q \in W \subset int(L)$, $|W|$ is finite and W is m-connected, for any $m \in \{4, 6, 8\}$. Observe that W is maximal connected coherent. Now let us prove that the Propagation procedure covers exactly all points inside W. As W is connected, there exists a $\mathbb{Z}^2$–path linking Q to P for all P in W. Thus, P is enqueued just once. From this, we conclude that the procedure covers W. The region W is limited by a closed sublist L^1 of L and $L^1 \cup W$ is m-connected. Observe that all points $P \in L^1$ satisfy Colour(P)=INTERIOR_COLOUR. Thus, they are not enqueued and the Propagation procedure can not reach the exterior of W.

The following theorem shows that our algorithm finds the maximal connected coherent regions of the interior of L.

Theorem 5.1 *The Filling_with_coherence algorithm finds the maximal connected coherent regions of the interior of L, where L is a closed list.*

Proof: If $int(L) = \emptyset$, there are no interior points and the Detection procedure always returns FALSE. Thus, the Propagation procedure is never called.

If $int(L) \neq \emptyset$, then $\exists W_1, W_2, \cdots, W_n \subset int(L)$, such that $int(L) = \bigcup_{i=i}^{n} W_i$ and $W_i \neq \emptyset$. Using theorem 2.1 we have $W_i \cap W_j = \emptyset$, $\forall i \neq j$. Suppose that there exists W_i not detected by the algorithm. Observe that the only way to detect a point of W_i is by using the sublist L^1 that bounds W_i. Therefore, Detection(P, L_1, Q) is always FALSE. According to the Lemma 5.1, such a region does not exist.Applying lemma 5.2 for each Q_i returned by Detection(P, L, Q_i), the result holds.

The Corthout–Pol Point Containment test is only performed in the Detection procedure. This test is called, at most m times for each query point, $m \in \{4, 6, 8\}$.

Thus, the Detection procedure has time complexity $O(1).c_{wn}$, where c_{wn} is the cost of Point Containment test.

Finally, for each point in the list, the Detection test is called just once. Thus, the total number of calls to the Point Containment test is $O(p)$, where p is the perimeter[2] of the list.

5.1 Comparison of theoretical results

Three approaches to perform the filling task were given: the initial quadratic version, the quasi-linear version and our resolution-independent version.

Observe that any rasterization process has two common steps:

[2] Perimeter of the discrete curve represented by the list.

- **Testing:** the points in the image are tested to decide if they are interior or not.
- **Propagation:** According to the interior/exterior position, the colour of points are changed. Note that, for any rasterization process, this step has quadratic behaviour with respect to resolution.

The Table 1 summarize the theoretical complexities of these approaches, according to the preceding step classification.

Approach	Testing step	Propagation step
Initial	$O(n^2).c_{wn}$	$O(n^2)$
Quadtrees	$O(n+p).c_{wn}$	$O(n^2)$
New	$O(p).c_{wn}$	$O(n^2)$

Table 1: Results summary

In the testing step, the new algorithm reduces the theoretical complexity. We observe that the the testing step is highly more expensive than the propagation step.

6 Results

Images were generated on a Macintosh Quadra 605, with monitor resolution 256x256 and no arithmetic coprocessor, saved as PostScript files and printed on a Hewlett–Packard LaserJet 4 Plus printer at 600 dpi.

In the Plate 1, we have a polygonal self-intersecting list. It shows the ability of the new method to captures details, such as the small interior regions.

Plate 2 explores an example of a curve composed by 100 cubic Bézier segments and 1 line segment. Note that there is a variety of thicknesses of the interior region. and the small exterior region is also preserved.

Table 2 compares the efficiency (in seconds) of the examples.

Plate	Initial	Quadtrees	New Method
1	105	29	15
2	605	40	10

Table 2: Computational time on the plates

7 Conclusions and further work

In this paper we have presented an efficient way to perform the filling operation for non-simple closed curves using the Corthout–Pol Point Containment test with coherence. The coherence approach enables the development of a simple algorithm, whose main characteristics are: resolution-independent complexity, simple structures and suitable to hardware implementation.

The research done in this paper can be further extended in both theoretical and practical directions.

Current work is going to investigate the optimality of our method and to perform other raster operations such as stroking. By doing this we would be providing a framework to support the efficient production of antialiased two-dimensional images using Point Containment based techniques [6, 7].

References

[1] F.P. Brooks. Springing into Fifth Decade of Computer Graphics – Where We've Been and Where We're Going! In *Computer Graphics (SIGGRAPH '96 Proceedings)*, volume 29, page 513, August 1996.

[2] M.E.A. Corthout and H.B.M. Jonkers. A new point containment algorithm for B-regions in the discrete plane. In R.A. Earnshaw, editor, *Theoretical Foundations of Computer Graphics and CAD*, NATO Advanced Study Institute Series, Series F, F40, pages 297–306. Springer-Verlag, 1988.

[3] M.E.A. Corthout and H.B.M. Jonkers. A point containment algorithm for regions in the discrete plane outlined by rational bézier curves. In J. André and R.D. Hersch, editors, *Raster Imaging and Digital Typography*, pages 169–179. Cambridge University Press, 1989.

[4] M.E.A. Corthout and E.-J.D. Pol. Supporting outline font rendering in dedicated silicon: the pharos chip. In *Raster Imaging and Digital Typography II*, pages 177–189. Cambridge University Press, 1991.

[5] M.E.A. Corthout and E.-J.D. Pol. *Point Containment and the PHAROS Chip*. PhD thesis, University of Leiden, Leiden, March 1992.

[6] A.E. Fabris. *Robust Antialiasing of Curves*. PhD thesis, University of East Anglia, Norwich, November 1995.

[7] A.E. Fabris and A.R. Forrest. Antialiasing of curves by discrete pre-filtering. In *Computer Graphics (SIGGRAPH '97 Proceedings)*, August 1997.

[8] A.R. Forrest. Computational geometry in practice. In R.A. Earnshaw, editor, *Fundamental Algorithms for Computer Graphics*, NATO Advanced Study Institute Series, Series F, F17, pages 707–724. Springer-Verlag, 1985.

[9] A.R. Forrest. *Presentation at the Panel Session on Fundamental Algorithms: Retrospect and Prospect*. ACM SIGRAPH 1985, San Francisco, July 1985.

[10] L.J. Guibas, L.H. Ramshaw, and J. Stolfi. A kinetic framework for computational geometry. In *Proceedings of 24th IEEE Symposium on the Foundations of Computer Science*, pages 100–111, 1983.

[11] G.M. Hunter and K. Steiglitz. Operations on images using quadtrees. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, pages 145–153, 1979.

[12] R.V. Klassen. Drawing antialiased cubic spline curves. *ACM Transactions on Graphics*, 10(1):92–108, January 1991.

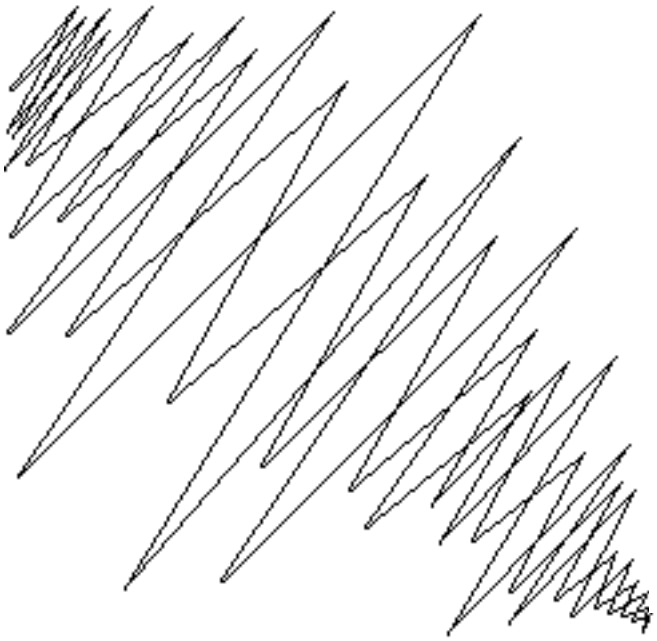
[13] S.-L. Lien, M. Shantz, and V.R. Pratt. Adaptive forward differencing for rendering curves and surfaces. In *Computer Graphics (SIGGRAPH '87 Proceedings)*, volume 21, pages 111–118, July 1987.

[14] W. Newman and R. Sproull. *Principles of Interactive Computer Graphics*. McGraw–Hill, second edition, 1983.

[15] A. Rosenfeld and J.L. Pfaltz. Distance functions on digital pictures. *Pattern Recognition*, 1:33–61, 1968.

[16] J. Serra. *Image Analysis and Mathematical Morphology*. Academic Press, 1982.

[17] C.J. Van Wyk. Clipping on the boundary of a circular-arc polygon. *Computer Vision, Graphics, and Image Processing*, 25:383–392, 1984.



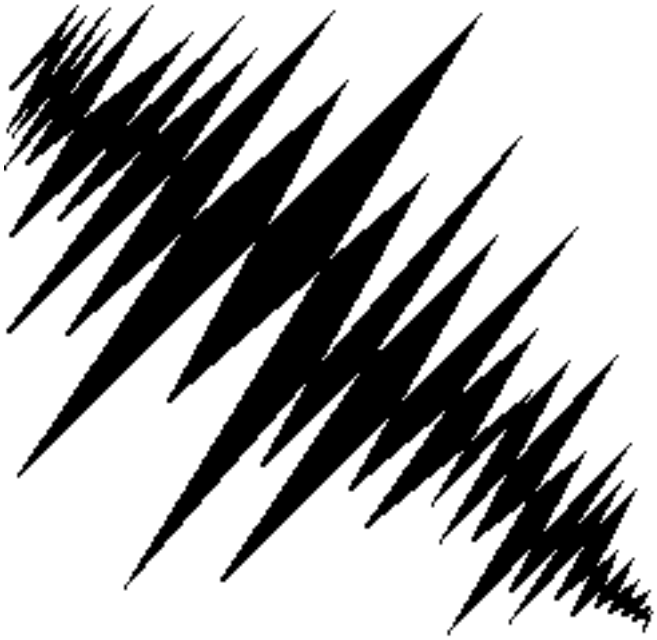


Plate 2: Cubic Bézier segments

Plate 1: Polygonal curve