A Note on Approximations of Rectilinear Polygons[*†]
(Extended Abstract)

Jian Huang Anil Maheshwari Doron Nussbaum
Jörg-Rüdiger Sack

School of Computer Science
Carleton University
Ottawa, Ontario, Canada K1S 5B6

1 Introduction

This paper studies a problem of computing optimal settings for multi-leaf collimators for conformal therapy arising in medical physics [4]. The problem translates to finding an area-optimal approximation of a rectilinear polygon P by a monotone rectilinear polygon Q enclosing P and having (almost) all its horizontal edges of equal width w.

Throughout, all geometric objects (polygons, paths, boundaries, etc.) are implicitly assumed to be rectilinear, i.e., each of their constituent segments is parallel to one of the coordinate axes. Rectilinear objects arise naturally in image processing. A (rectilinear) polygon Q is said to be *monotone* along the x-axis if any vertical line through Q intersects Q in one connected component.

Let P be a simple (rectilinear) polygon on n-vertices. Let Q be a simple monotone rectilinear polygon such that each of its horizontal edges, with the exception of four edges, is of length w, where $w > 0$. The four exceptional edges are the leftmost and the rightmost horizontal edges. We are interested in computing Q such that P is completely contained inside Q and the difference between the areas of P and Q is minimized. We call such a polygon Q an *optimal approximation polygon* for P, or simply an *optimal* polygon.

In this paper we present $O(n \log n)$ time algorithms for computing such an optimal polygon Q. When the input is given as a digitized image our algorithms run in optimal linear-time. Our $O(n \log n)$ algorithms have been developed and are stated so that optimal parallel algorithms are easily derived.

Note that there could be several optimal polygons; we are interested in computing one such optimal polygon. There have been several developments in the area of polygon approximations [1, 3, 5, 7]. We do not know of any previous results on the problems considered here.

In Section 2 we present algorithms for the case that P is a staircase polygon and in Section

[1]Research supported in part by NSERC (Natural Sciences and Engineering Research Council of Canada) grant.
[2]Research supported in part under an R&D agreement between Carleton University and ALMERCO Inc.

3 when P is a monotone polygon. In Section 4 we present algorithms for the case when P is a simple rectilinear polygon.

2 Staircase Polygon

A polygon is said to be a *staircase* polygon if it is monotone along both coordinate axes (i.e., it is $x - y$ monotone). Let the given polygon P be a staircase polygon on n vertices. We wish to compute an optimal approximation polygon, Q, for P.

To compute an optimal Q, first divide the boundary of P into two chains - the top chain and the bottom chain. Note that both chains are staircases since P is a staircase polygon. We first discuss the case that the approximation of the two staircases is independent. Then we discuss the situation that the vertical edges of Q for the top and the bottom chain are aligned, i.e., the leftmost (rightmost) edges of the top and bottom chain have equal length. This requirement arises in certain collimators. In this case, an optimal approximation of the top or the bottom chain is not necessarily an optimal approximation for the entire polygon.

2.1 Independent Chains

In this case, we compute Q by computing its optimal top and bottom chain. The boundary of Q consists of the top and bottom chain joined by two vertical edges. Since the computation of the top and bottom chain of Q is analogous, here we describe the computation of only the top chain of Q. To avoid the use of additional notation, we refer the top chain of P and Q by P and Q itself, respectively. It is easy to see that for an optimal Q, each horizontal edge of Q touches some horizontal edge of P. We prove the following crucial lemma.

Lemma 2.1 *At least one of the vertical edges of an optimal Q touches a vertical edge of P, where P is a staircase.*

The above lemma suggests the following algorithm for computing an optimal Q. Place a vertical edge of Q touching a vertical edge of P. Note that fixing one edge of Q uniquely determines the chain Q, since each horizontal edge of Q is of width w and it touches some horizontal edge of P. It is easy to see that it takes $O(n)$ sequential time to compute Q, once we fix one of its vertical edge. Since the total number of possibilities for Q is the same as the number of vertical edges of P, which is bounded by n, an optimal Q can be found in $O(n^2)$ time. In the following we show that an optimal Q can be computed more efficiently.

Assuming that the leftmost vertical edge of P is touched by a vertical edge of Q, compute the chain Q. This takes $O(n)$ time. For each vertical edge of P, compute the distance to the closest vertical edge of Q which is to the left of P. If we assume that P has n vertical edges, we get n distance values at the end of the above step. Sort these distance values in an ascending order and let they be $d_1 \leq d_2 \leq \cdots \leq d_n$. Note that each $d_i < w$ and $d_1 = 0$. Without loss of generality assume that y_i is the length of the vertical edge of P for which the distance to the closest vertical edge of Q is d_i. The difference between the area of P and Q is $E_1 = \sum_{i=1}^{n} y_i * d_i$. We call this the *error* of our approximation of P by Q. Our aim is to reduce this error as much as possible. Move Q to the right by a distance d_2. This results in a vertical edge of Q touching the vertical edge of P whose height is y_2. It is easy to see that the new error is $E_2 = \sum_{i=2}^{n} y_i * (d_i - d_2) + y_1(w - d_2)$. This expression can alternatively be written as $E_2 = C + w * y_1 - d_2 * Y$, where $C = \sum_{i=1}^{n} y_i * d_i$ and $Y = \sum_{i=1}^{n} y_i$. In general the expression for E_j, where the vertical edge of Q corresponding to the distance d_j touches P is $E_j = C + w * \sum_{i=1}^{j-1} y_i - d_j * Y$. The partial sums $\sum_{i=1}^{j-1} y_i$, Y and C can be computed in linear time. Therefore all the E_j's can be computed in linear time. Compute the minimum E_j out of the n-possible values of E_j's. The desired Q is the one corresponding to the minimum E_j. Note that each step other than the sorting of d_i values takes $O(n)$ time.

2.2 Aligned Chains

When the approximating chains are aligned due to hardware constraints, approximations for the top or bottom chain alone do not suffice to obtain an optimal approximation for the entire polygon. We can however show that the above Lemma 2.1 still holds and that our algorithm can be adapted to handle this situation.

We have developed and stated our sequential algorithms in such a way that they are easily parallelizable. We summarize our result for the entire section in the following theorem.

Theorem 2.1 *An optimal approximation polygon (aligned or independent) for a staircase polygon P on n-vertices can be found in $O(n \log n)$ sequential time, and in $O(\log n)$ parallel time with an optimal EREW PRAM algorithm.*

Since the input polygon is usually obtained from an image where all coordinates are integers in some bounded domain. Sorting the d_i' therefore takes linear time and we obtain:

Corollary 2.1 *An optimal approximation polygon (aligned or independent) for a staircase polygon P on n-vertices derived from an image can be found in $O(n)$ sequential time.*

3 Monotone Polygon

In this section we consider the problem of computing an optimal approximation Q for a simple monotone rectilinear polygon P on n vertices. Assume that P is monotone in the direction of x-axis.

To compute Q, we first divide the boundary of P into two chains - the top chain and the bottom chain. Note that both chains are monotone since P is a monotone polygon. We first discuss the case that the approximation of the two monotone chains is independent. Then we discuss the situation that the vertical edges of Q for the top and the bottom chain are aligned.

3.1 Independent Chains

We compute Q by computing its optimal top and bottom chain. Once we know the top and bottom chain of Q, Q can be computed easily. Here we describe the computation of the top chain of Q. Let us denote the top chain of Q (P) by Q (respectively, P).

It is easy to see that for an optimal Q, each horizontal edge of Q touches some horizontal edge of P. Now we prove a lemma similar to Lemma 2.1. We need the following definitions. P is comprised of staircases which are increasing and decreasing along y axis. Call a vertical edge of P as increasing (decreasing) edge if it is on an increasing (respectively, decreasing) staircase. Let y_i^I (y_i^D) denote the height of an increasing (respectively, decreasing) edge of P. Let $I = \sum y_i^I$ and $D = \sum y_i^D$. Let d_i^D (d_i^I) denote the distance between the decreasing (respectively, increasing) edge of P and the closest vertical edge of Q to its right (respectively, left). Let $d'^I_i = w - d_i^I$. Let $d_{\min} = \min\{d_i^D, d'^I_i\}$ for all possible choices of i.

Lemma 3.1 *At least one of the vertical edges of an optimal Q touches a vertical edge of P, where P is a monotone polygon.*

Proof We prove this by contradiction. Assume that no vertical edge of an optimal Q touches a vertical edge of P. The total error $E = \sum y_i^I * d_i^I + \sum y_i^D * d_i^D$. There are three cases depending upon whether (i) $I < D$ (ii) $D < I$ or (iii) $I = D$.

Consider the case when $I < D$. Since $I < D$, we move Q in $-x$ direction by $d_{\min}$. Due to this movement the distance between the vertical edges of Q and the decreasing (increasing) edges of P decreases (respectively, increases). Also a vertical edge of Q either touches a decreasing (increasing) edge of P, depending upon whether $d_{\min}$ is d_i^D (respectively, d'^I_i). Consider the case when a vertical edge of Q touches a decreasing edge of P. Then the error is $E' = \sum y_i^I * (d_i^I + d_{\min}) + \sum y_i^D * (d_i^D - d_{\min})$. The above expression can be simplified and it results in $E' = E - d_{\min} * (D - I)$. Since $D > I$, $E' < E$. This case contradicts the assumption that Q is optimal. Consider the case when a vertical edge of Q touches an increasing edge of P. Then the error is $E' = \sum y_i^I * (d_i^I + d_{\min}) + \sum y_i^D * (d_i^D - d_{\min}) - w * y_k$, where y_k is the height of the increasing edge of P corresponding to $d_{\min}$. On simplifying this we get $E' = E - w * y_k + d_{\min} * (D - I)$. It is easy to see that $E' < E$, and therefore the optimality of Q is contradicted.

The other two cases can be analyzed similarly and in each case we either obtain $E' < E$ contradicting the optimality of Q or $E' = E$ where the approximation polygon associated with E' is also optimal. ■

Using this lemma and the algorithms developed in Section 2 we can develop our algorithm for computing an optimal approximation Q of P. As before, we start with an initial approximation and compute the d_i's and the initial error E_1 of our approximation. After that we move Q in steps of $d_{\min}$ and obtain succesive approximations. The expression for the error in step j, where $D > I$, is $E_j = E_{j-1} + w * (y_{j-1}^D - y_j^I) - d_{\min} * (D - I)$. In the above expression y_j^I is the height of an increasing vertical edge of P that was not touched at the $j-1$st approximation but will be touched at the jth approximation and y_{j-1}^D is the height of a decreasing vertical edge of P that was touched at the $j-1$st approximation. From this iterative equation compute all possible E_j's. The optimal Q is the one corresponding to the minimum value of E_j. We omit the details here due to space limitation.

3.2 Aligned Chains

When the approximating chains are aligned due to hardware constraints, as before, approximations for top or bottom chain alone do not suffice to obtain an optimal approximation for the entire polygon. We can however show that the above Lemma 3.1 still holds and that our algorithm can be adapted to handle this situation. We summarize our result for the entire section in the following theorem.

Theorem 3.1 *An optimal approximation polygon (aligned or independent) for a monotone polygon P on n-vertices can be found in $O(n \log n)$ sequential time, and in $O(\log n)$ parallel time with an optimal EREW PRAM algorithm.*

Corollary 3.1 *An optimal approximation polygon (aligned or independent) for a monotone polygon P on n-vertices derived from an image can be found in $O(n)$ sequential time.*

4 Simple Polygon

In this section we consider the problem of computing an optimal approximation Q for a simple rectilinear polygon P on n vertices. We reduce this problem to that of computing an approximation of simple monotone rectilinear polygon. We achieve the reduction as follows.

Compute the external boundary of P that is visible from a point at +ve and -ve infinity along the y-axis. Let P' be the polygon formed by the visibility boundary of P. It is known that P' is a simple rectilinear polygon and is monotone along x-axis. Furthermore P' can be computed in linear time sequentially [6] and in work-optimal parallel time [2]. Regions of $P' - P$ are called invisible pockets. We have the following lemma.

Lemma 4.1 *An optimal approximation polygon Q of P' is also an optimal approximation polygon for P.*

PROOF Any optimal polygon Q contains all invisible pockets. Therefore, an optimal Q for P and P' is the same. ∎

Since P' is a simple monotone rectilinear polygon, we can use the results of the previous section to compute an optimal Q. We summarize the results in the following theorem.

Theorem 4.1 *An optimal approximation polygon (aligned or independent) for a simple rectilinear polygon P on n-vertices can be found in $O(n \log n)$ sequential time, and in $O(\log n)$ parallel time with a work-optimal EREW PRAM algorithm.*

Corollary 4.1 *An optimal approximation polygon (aligned or independent) for a simple rectilinear polygon P on n-vertices derived from an image can be found in $O(n)$ sequential time.*

5 Conclusion

We have developed efficient and easy to implement algorithms for approximating rectilinear polygons area-optimally. The problem arises in the context of conformal therapy when multi-leaf collimators are used. This work is part of an on-going collaboration in which we study a variety of related 2 and 3 dimensional problems arising in the context of medical physics.

References

[1] A. Aggarwal, J.S. Chang, C.K. Yap, *Minimum area circumscribing polygons*, Visual Computer, Vol. 1, 1985, pp. 112-117.

[2] M.J. Atallah, D.Z. Chen, H. Wagener, *Optimal parallel algorithm for visibility of a simple polygon from a point*, J. ACM, Vol. 38, 1991, pp. 516-553.

[3] L.J. Guibas, J. Hershberger, J.S.B. Mitchell, J. Snoeyink, *Approximating polygons and subdivisions with minimum link path*, Proc. 2nd ISAAC, LNCS 557, 1991, pp. 151-162.

[4] A.R. Hounsell, P.J. Sharrock, C.J. Moore, A.J. Shaw, J.M. Wilkinson, P.C. Williams, *Computer-assisted generation of multi-leaf collimator settings for conformation therapy*, The British Journal of Radiology, Vol. 65, 1992, pp. 321-326.

[5] C.C. Lu and J.G. Dunham, *Shape matching using polygon approximation and dynamic alignment*, Pattern Recognition Letters, Vol. 14, 1993, pp. 945-949.

[6] J.-R. Sack, *Rectilinear Computational Geometry*, Ph.D. Thesis, McGill University, 1984.

[7] J.S. Wu and J.J. Leou, *New polygonal approximation schemes for object shape representation*, Pattern Recognition, Vol. 26, No. 4, 1993, pp. 471-484.