Isothetic Polygonal Approximations of a 2D Object on Generalized Grid

Partha Bhowmick[1], Arindam Biswas[1], and Bhargab B. Bhattacharya[2]

[1] Computer Science and Technology Department,
Bengal Engineering and Science University, Shibpur, Howrah, India
{partha, abiswas}@becs.ac.in
[2] Advanced Computing and Microelectronics Unit,
Indian Statistical Institute, Kolkata, India
bhargab@isical.ac.in

Abstract. Determination of an isothetic polygonal approximation of the outer or inner contour of an object is a challenging problem with numerous applications to pattern recognition and image processing. In this paper, a novel algorithm is presented for constructing the outer (or inner) tight isothetic polygon(s) containing (or filling) an arbitrarily shaped 2D object on a background grid, using a classical labeling technique. The background grid may consist of uniformly or non-uniformly spaced horizontal and vertical lines. Experimental results for both uniform and non-uniform grids of varying sizes have been reported to demonstrate the applicability and efficiency of the proposed algorithm.

1 Introduction

Given a closed 2D contour (object) $\mathcal{C}$, we consider the problem of constructing the set of isothetic polygons that either fills the object (set of inner polygons, $\{\mathcal{P}_{in}(\mathcal{C})\}$) or contains the object (set of outer polygons, $\{\mathcal{P}_{out}(\mathcal{C})\}$), imposed by a set of horizontal and vertical grid lines, $\mathcal{G}$. The difficulty of successfully finding the isothetic polygon(s) corresponding to an arbitrarily shaped object lies in tracing a valid path defining the polygon boundary, especially in a convoluted region (on the background for outer polygons and on the object for inner polygons) that may propagate in every possible direction at every possible instant. An improper path surpassing a "dead end" may lead to a point whence no further advancement is possible. Further, if any backtracking is adopted, then in the case of complex and convoluted objects, the tracing procedure may become protracted, which would increase computational burden and difficulty in implementation.

It may be noted that the set of isothetic polygons corresponding to a given set of objects is useful to many interesting applications, such as area location [6], grasping objects by a robot [5, 8, 9, 17], newspaper page decomposition and article tracking [7], document image analysis [1], deriving free configuration space (path-planner) for robot navigation, lower and upper approximation in rough sets [14, 15] (in non-uniform grid spacing), image mining using rough sets [10],

S.K. Pal et al. (Eds.): PReMI 2005, LNCS 3776, pp. 407–412, 2005.

VLSI layout design [13, 16], and computational geometry [2]. For rough set applications, the grid lines or decision lines imposed on the object are usually non-uniformly spaced. The proposed algorithm, therefore, would be helpful for rough set applications in 2D, as well as in higher dimensions, if properly modified, for its fitness and aptitude in non-uniform grid.

Very recently, another algorithm [3] for finding the outer isothetic polygon has been published. However, the algorithm proposed here is different in several aspects. In [3], the outer isothetic polygon is derived from two matrices, namely "unit edge matrix" and "unit square matrix", defined on the uniformly spaced grid. On the contrary, the algorithm, proposed here, is based on labeling of grid points by 15 unique labels, meant for determining the class and type of a vertex of the polygon. Further, in the proposed algorithm, non-uniform grids, multiple polygons, self-intersections of polygons, polygons for holes, etc., have been addressed and solved, which are not considered earlier [3].

2 Proposed Algorithm

The algorithm consists of three stages. For an outer (inner) polygon, Stage 1 labels a grid point p if and only if the following conditions are satisfied simultaneously:

(**C1**) p lies on the background (object);
(**C2**) object (background) lies in the 4-neighborhood of p.

In Stage 1, each grid point that satisfies (**C1**) and (**C2**) simultaneously is assigned with left/right/bottom/top (L/R/B/T) or a combination of them. Stage 2 extracts the vertices of the polygon on the basis of their labels in Stage 1. Stage 3, called the construction stage, constructs the isothetic outer polygon with the help of the vertex types assigned in the Stage 2. In the subsequent discussions, we adhere to the process of finding the outer isothetic polygon only, since that for the inner one is very much similar, as evident from the object–background duality mentioned above.

2.1 Stage 1: Labeling the Grid Points

Let p_1 be a grid point on the background that lies in the 4-neighborhood of the object. A depth-first search (DFS)-Visit [4] is initiated from p_1, followed by its (ordered) adjacency list, Adj[p_1], containing eight neighboring grid points of p_1 in counter-clockwise order. For each of these eight neighbors, the DFS-Visit is recursively executed, provided (**C1**) and (**C2**) are satisfied simultaneously.

A grid point p is labeled as L/R/B/T or a combination of them depending upon which of the four grid edges incident on p intersect the object, as illustrated in Fig. 1. For example, in Fig. 1(a), only the right edge of p intersects the object, which implies that p lies left of the object, whence p is labeled as L. Similarly, in Fig. 1(b), since p lies left and top of the object simultaneously, it is labeled by LT, and in Fig. 1(c), since p lies bottom and top of the object simultaneously, it is labeled by BT.

It may be noted that a grid point p can be labeled in $2^4 - 1 = 15$ different ways, namely, L, R, B, T, LR, LB, LT, RB, RT, BT, LRB, LRT, LBT, RBT,

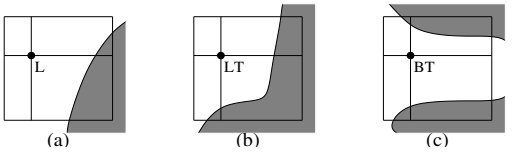


Fig. 1. An illustration of labeling for some cases of p

and LRBT. Out of these 15 cases, 4 cases (LB, LT, RB, RT) are candidate vertices (with internal angle 270^0) of $\mathcal{P}_{out}(\mathcal{C})$, and 4 other cases (L, R, B, T) are ordinary edge points of $\mathcal{P}_{out}(\mathcal{C})$. The remaining 7 cases (the labels containing LR or BT) are considered as **dead ends**. A dead end primarily signifies that it does not qualify for being a vertex or an edge point of $\mathcal{P}_{out}(\mathcal{C})$ as evident in Fig. 1(c), where a dead end, namely BT, has been illustrated. Further, a dead end signals a sharp change in curvature of the contour, which may call for a detailed examination in its neighborhood with a denser grid, if required.

2.2 Stage 2: Extracting the Candidate Vertices of $\mathcal{P}_{out}(\mathcal{C})$

Since $\mathcal{P}_{out}(\mathcal{C})$ is an isothetic polygon, each vertex of $\mathcal{P}_{out}(\mathcal{C})$ can have internal angle either 90^0 or 270^0. Based on this observation, Stage 2 extracts and classifies the vertices using object containments of the four quadrants, namely Q_1, Q_2, Q_3, Q_4 (see Fig. 1), associated with p, which, in turn, are derived from the labels of the grid points awarded in Stage 1. These vertices are of three classes, which are as follows.

(i) 90^0 **vertex:** It can be shown that, if a grid point p is a 90^0 vertex, then part of the object lies in exactly one of the neighboring four quadrants of p (Fig. 2). A 90^0 vertex can be sub-classified into four types, namely NE, NW, SE, SW, when the object lies in Q_1, Q_2, Q_3, Q_4 respectively.
(ii) 270^0 **vertex:** For a 270^0 vertex, p, object lies in exactly three quadrants of p (Fig. 2). This class is also sub-classified to NE, NW, SE, SW, when the object not lies in Q_1, Q_2, Q_3, Q_4 respectively.
(iii) **Cross vertex:** A cross vertex is a point of self-intersection. Since an isothetic polygon may have self-intersection(s), a cross vertex is merely the point of intersection of a horizontal edge and a vertical edge of $\mathcal{P}_{out}(\mathcal{C})$. A cross vertex is sub-classified into two types, namely NWSE (object is in Q_1 and Q_3) and NESW (object is in Q_2 and Q_4, see Fig. 2).

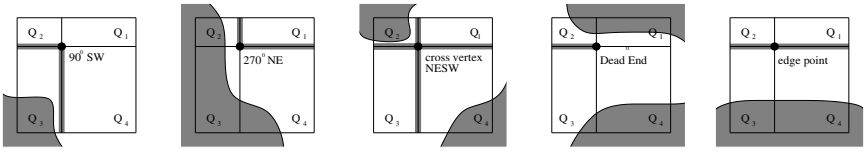


Fig. 2. Different vertex types

The sub-classification of the vertex types into NE, NW, SE, and SW aids the process of construction of the polygon. For example, if p is labeled as NE, then the outgoing edge from p will be its east-bound edge, provided the incoming edge for p is its north-bound edge; similarly, the outgoing edge from p will be its north-bound edge, provided the incoming edge for p is its east-bound edge.

2.3 Stage 3: Construction of $\mathcal{P}(\mathcal{C})$

Stage 3 constructs the outer polygon using the types of the detected vertices in Stage 2. It may be noted that the class of a vertex is not required, and only the type of the vertex is sufficient, for construction of the polygon.

The construction starts from a grid point, which is classified as a 90^0 vertex in stage 2. This vertex is specially marked as the start vertex, v_s, for the construction. It can be shown that a 270^0 vertex may not appear in the set of detected vertices in Stage 2. Hence a 90^0 vertex is always considered as a start vertex (v_s). Now, let v_c be the vertex that is currently under visit during the construction. Depending on the (incoming) edge along which v_c is visited and the type of v_c, the outgoing edge from v_c is determined. The traversal proceeds along the direction of this outgoing edge until the next vertex, v_n, is found. This process is repeated until v_n coincides with v_s.

2.4 Time Complexity

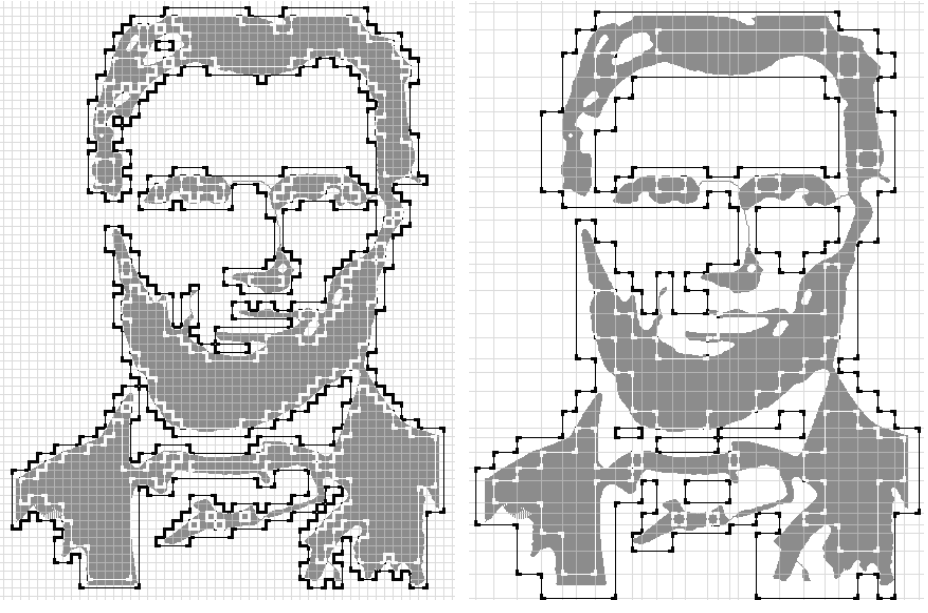
In Stage 1, since p_1 is the grid point lying in the 4-neighborhood of the object from where the DFS-Visit is initiated, the next grid point visited would be one of its eight neighbors in Adj$[p_1]$, which being recursively executed, would finally terminate when p_1 is reached. Since each node p in this recursive DFS-Visit satisfies conditions (**C1**) and (**C2**), the number of nodes visited in the entire process would be linearly bounded by the number of grid points satisfying conditions (**C1**) and (**C2**). In Stage 2, the labels are used for deciding the vertices, which are finally used in Stage 3 to construct the polygon. Hence, for an object with no hole and having one isothetic polygon (inner or outer), the time complexity for finding out its single isothetic polygon is given by $O(|\mathcal{C}|/\alpha^2)$, where $|\mathcal{C}|$ denotes the length of the contour $\mathcal{C}$ and α denotes the spacing (minimum for non-uniform) between the grid lines.

If the object is having holes and/or requires multiple polygons (inner or outer) for its representation, then the time complexity for finding out its complete set of isothetic polygons, containing m polygons, is given by $O(m|\mathcal{C}|/\alpha^2)$, provided the starting grid point p_1 is given for each polygon. If the starting grid point p_1 is unknown for each polygon, then the complexity is $O(MN/\alpha^2)$, where M and N denote the width and height of the 2D plane containing $\mathcal{C}$.

3 Results

The above algorithm has the ability to extract the inner and outer polygons of a digital object of any complex shape, whether defined on uniform or non-uniform

background grid. In addition, this algorithm also produces self-intersecting polygons when a cross vertex is treated as a point of self-intersection. We have tested it on a number of complex 2-D objects producing correct results in all cases, one of which being shown in Fig. 3.



(a) Uniform grid: $\alpha = 8$ (b) Non-uniform grid: $\alpha = 8 - 24$

Fig. 3. Inner and outer polygons of Lincoln image on uniform and non-uniform grids

It may be noted that the number of vertices of both the outer and the inner polygons decrease with increase in grid size, thereby reducing the complexity of the isothetic polygon. For the image (size 430×570) in Fig. 3, CPU times for construction of isothetic polygons for grid size 1 are 956 and 507 milliseconds, which drops drastically to 4 and 6 milliseconds for grid size 16, for inner and outer polygons respectively.

4 Conclusion and Future Work

In this paper, we have presented an elegant algorithm to capture approximate outer and inner isothetic polygons covering a 2D object on both uniformly and non-uniformly spaced grid lines. This algorithm is fast and can be extended to higher dimensions, facilitating the determination of outer and inner approximations in rough sets. A detailed analysis can be done with respect to the point of inflections to make it amenable to different critical applications like VLSI design and verification, and robot grasping.

References

1. O. T. Akindele and A. Belaid, *Page Segmentation by Segment Tracing*, ICDAR 1993, pp. 341–344.
2. A. Bemporad, C. Filippi, and F. D. Torrisi, *Inner and Outer Approximations of Polytopes using Boxes*, Computational Geometry - Theory and Applications, vol. 27, 2004, pp. 151–178.
3. A. Biswas, P. Bhowmick, and B. B. Bhattacharya, *TIPS: On Finding a Tight Isothetic Polygonal Shape Covering a 2D Object*, 14th Scandinavian Conf. on Image Analysis (SCIA 2005), LNCS, vol. 3540, pp. 930–939.
4. T. H. Cormen, C.E. Leiserson, and R.L. Rivest *Introduction to Algorithms*, Prentice Hall India, New Delhi, 2000.
5. L. Gatrell, *Cad-based Grasp Synthesis Utilizing Polygons, Edges and Vertices*, Proc. IEEE Intl. Conf. Robotics and Automation, 1989, pp. 184–189.
6. B. Gatos and S. L. Mantzaris, *A Novel Recursive Algorithm for Area Location using Isothetic Polygons*, ICPR 2000, pp. 492–495.
7. B. Gatos, S. L. Mantzaris, K. V. Chandrinos, A. Tsigris, and S. J. Perantonis, *Integrated Algorithms for Newspaper Page Decomposition and Article Tracking*, ICDAR 1999, pp. 559–562.
8. Y. Kamon, T. Flash, and S. Edelman, *Learning to Grasp using Visual Information*, Proc. IEEE Intl. Conf. Robotics and Automation, 1995, pp. 2470–2476.
9. J. Lengyel, M. Reichert, B. R. Donald, and D. P. Greenberg, *Real-time Robot Motion Planning Using Rasterizing Computer*, Computer Graphics, ACM, vol. 24(4), pp. 327–335.
10. M. Liu, Y. He, H. Hu, and D. Yu, *Dimension Reduction Based on Rough Set in Image Mining*, Intl. Conf. on Computer and Information Technology (CIT'04), 2004, pp. 39–44.
11. A. Morales, P. J. Sanz, and A. P. del Pobil, *Vision-Based Computation of Three-Finger Grasps on Unknown Planar Objects*, IEEE Intl. Conf. on Intelligent Robots and Systems, 2002, pp. 1711–1716.
12. S. C. Nandy and B. B. Bhattacharya, *Safety Zone Problem*, Journal of Algorithms, vol. 37, 2000, pp. 538–569.
13. T. Ohtsuki, *Layout Design and Verification*, North-Holland, Amsterdam, 1986.
14. S. K. Pal and P. Mitra, *Case Generation Using Rough Sets with Fuzzy Representation*, IEEE Trans. on Knowledge and Data Engg., vol. 16(3), 2004, pp. 292–300.
15. S. K. Pal and P. Mitra, *Pattern Recognition Algorithms for Data Mining*, Chapman and Hall/CRC Press, Bocan Raton, FL, 2004.
16. N. Sherwani, *Algorithms for VLSI Physical Design Automation*, 3rd Edition, Kluwer Academic Publishers, Boston, 1999.
17. G. Taylor and L. Kleeman, *Grasping Unknown Objects with a Humanoid Robot*, Proc. Australasian Conf. on Robotics and Automation, 2002, pp. 191–196.