2013 13th International Conference on Control, Automation and Systems (ICCAS 2013)
Oct. 20-23, 2013 in Kimdaejung Convention Center, Gwangju, Korea

Modeling any Obstacle Shapes for Motion Planning of Circular Robots

Md Nasir Uddin Laskar*, Seung Y. Choi, Ishtiaq Ahmed and TaeChoong Chung

Artificial Intelligence Lab, Dept. of Computer Engineering, Kyung Hee University, 446-701, South Korea.
E-mail: {nasir, sychoi84, ishtiaq.khu, tcchung}@khu.ac.kr

Abstract: This paper presents an algorithm to model the workspace obstacles of a circular robot. An obstacle of any shape can be represented by line and circular arc segments. Based on robot radius, geometric representation is applied for both line and circular arcs and the operation is known as offsetting. Result of our algorithm creates an efficient configuration space and helps planning high-quality motion paths. Our method works differently for line and circular arc segments. Two major steps are: *(i)* finding the raw offset curve for both lines and circular arcs and *(ii)* removing the global invalid loops. The process takes $O(n)$ times to find the raw offset curve and $O((n+k)\log m)$ times to remove global invalid loops, where n is the number of vertices in the original polygon, k is the number of self-intersections, and m is the number of segments in the raw offset curve and always $m \leq n$. Local invalid loops are removed before generating the raw offset curve by invoking a pair-wise intersection detection test (PIDT). Our algorithm is very fast and computational complexity is approximately linear.

Keywords: Mobile robot, obstacle, polygon offsetting, motion planning, configuration space.

1. INTRODUCTION

Minkowski sum of polygon with a disc is called polygon offsetting. Most offsetting methods concentrate on convex polygons only and they divide non-convex polygons into convex sub-polygons. Computing the Minkowski sum of two convex polytopes can have $O(n^2)$ time complexity [1], where n is the number of features. Minkowski sum of two non-convex polyhedra costs even more and can be as high as $O(n^6)$ combinatorial complexity. If a non-convex polygon is divided into m sub-polygons, their union can have $O(m^3)$ time complexity [2]. Our method applies local geometry and reduces the complexity to approximately linear.

Offsetting 2D point-sequence curve (PS-curve) is classified into four categories [3]: pair-wise, level set, Voronoi diagram and bisector based. Choi and Park [4] proposed a pair-wise offset algorithm for a polygon with no islands. Wein [5] presented a technique of Minkowski sum with a disc to compute exact and approximate offset of polygons. Another variant of Minkowski sum of two polygons is proposed by Guibas [1] to compute the convolution of the two bodies.

Bisector method is introduced by [6] which can automatically bridge the islands with main profile. It takes $O(n^2)$ time to compute the distance between two PS-curves, where n is the total number of curves. After that, Kim [7] and Kim *et al.* [8] used bisector method to calculate raw offset curve. They used the direction every time to report the global invalid loops.

Voronoi diagram implementation of polygon offsetting is another popular approach to the researchers. Q. Bo [9] introduced a recursive method to compute a trimmed offset of a polygon. Held [10] presented a $O(n \log n)$ algorithm to generate Voronoi diagram of curvilinear polygons and then finds the offset from the Voronoi diagram in $O(n)$ time, where n is the number of boundaries of the polygon. A circular arc extension to [10] is done based on a randomized incremental insertion [11]. S. McMains [12] presented an algorithm to build thin-walled parts in a fused deposition modeling machine. X. Chen and S. Mcmains [13] applied winding numbers to offset polygons with $O((n+k)\log n)$ times, where n is the number of vertices in the input polygon and k is the number of self-intersections in the raw offset curve.

Robot motion planning has been studied extensively in the robotics literature for many years. There is a rich set of motion planning algorithms: grid-based search, cell decomposition, Minkowski sum, potential field method, sampling-based and some others. A methodology for robot path planning among polygonal obstacles in a dynamic environment is introduced in [14]. This offsetting contradicts with the smooth path properties described in [15] and cause the robot many sharp turns in the obstacle corners. Schwartz and Sharir [16] was the first to apply exact cell decomposition technique to solve the motion planning problem by their famous piano movers problem. Motion of a polygonal robot in an environment with polygonal obstacles is planned successfully by [17]. By obtaining a set of deterministic samples, [2] presented an algorithm to find the complete path for a translating polyhedral robot in 3D space.

Remainder of the paper is organized as following manner. Section 2 describes our method in two subsections for offsetting lines and arcs. Section 3 illustrates the way to handle global invalid loops. In section 4 we showed experimental results and performance analysis of our algorithm. Finally we conclude in section 5.

This work was supported a grant from the NIPA (National IT Industry Promotion Agency) in 2013 (Global IT Talents Program). This research was also partly supported by the Basic Science Research Program through the National Research Foundation of Korea (NRF) funded by the Ministry of Education, Science, and Technology (2010-0012609).

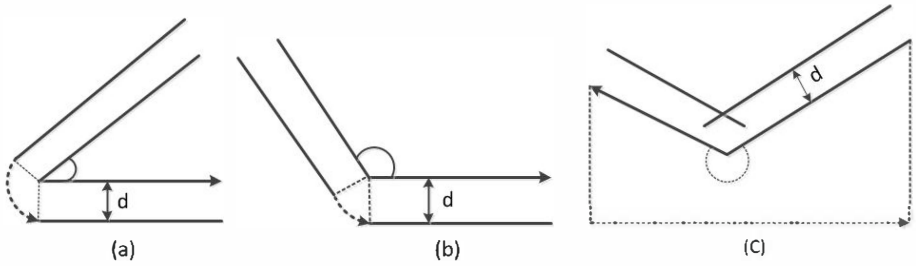


Fig. 1 Possible scenarios for offsetting polygon lines: (a) acute angle, (b) obtuse angle and (c) reflex angle.

2. MODELING OBSTACLES

Obstacle modeling is performed by computing the offsets of the workspace obstacles.Input to our algorithm is a point-sequence curve with rational coordinates oriented counterclockwise (CCW), a curve type for each segment, and the robot radius d. Curve type is used to trace the segments whether they are lines or arcs. Robot radius d works as the offset distance and we treat the polygon vertices differently depending on whether a vertex is convex or non-convex.

Throughout this paper, we indicate the points (or vertices) of the given polygon as p_i and the offset points as p'_j, where $i, j \in \mathbb{N}$. Since, in most of the cases, a vertex of the original polygon is associated with two points of the offset polygon, i and j are not necessarily be equal.

2.1 Offset Lines

A polygon can have convex as well as non-convex vertices. Fig. 1 demonstrates the possible offsetting cases for both convex and non-convex vertices of a polygon. Let a polygon P in Cartesian plane has n points $p_0(x_0, y_0)$, $p_1(x_1, y_1)$, $p_2(x_2, y_2)$ through $p_{n-1}(x_{n-1}, y_{n-1})$. Point $p_i(x_i, y_i)$ is directed to $p_{i+1}(x_{i+1}, y_{i+1})$. To offset the lines of the polygon P we shift each line segment by offset distance d along its normal direction or to the right side of the edge. As a result, we obtain n disconnected offset edges with $2n$ new points $p'_j(x'_j, y'_j)$ parallel to the original lines. For example, if we offset a line p_1p_2 we find the offset line $p'_1p'_2$, for line p_2p_3 we find the offset line $p'_3p'_4$ and for the line $p_{n-1}p_0$ we end up with the offset line $p'_{2n-2}p'_{2n-1}$. We test the convexity of a vertex p_i by computing the angle $p_\theta = \angle(p_{i-1}, p_i, p_{i+1})$ at that vertex. If $p_\theta < \pi$, we are turning left and the vertex is convex. If $p_\theta > \pi$, we are turning right and the vertex is non-convex. A vertex is considered differently if it is start or end of a polygon arc.

If the vertex is convex as in Fig. 1(a) and (b), where outside lines are always the offset lines, adjacent two new points of offset edges will be connected by a circular arc centered in the corresponding vertex with radius d, which is the radius of the robot. For a vertex p_i, we draw an arc centered in p_i, between the points p'_{2i-2} and p'_{2i-1}, which are the end points of adjacent offset edges, induced by $p_{i-1}p_i$ and p_ip_{i+1} respectively. The angle that defines such a circular arc is $\pi - \angle(p_{i-1}, p_i, p_{i+1})$.

Consider the line supporting p_1p_2 is $ax + by + c = 0$, where $a, b, c \in \mathbb{R}$. Representation of the offset edge is based on the locus of all points lying at offset distance d from the line $ax + by + c = 0$ is given by

$$d = \frac{|ax + by + c|}{\sqrt{(a^2 + b^2)}}. \tag{1}$$

If $p'_j(x'_j, y'_j)$ and $p'_{j+1}(x'_{j+1}, y'_{j+1})$ are the corresponding offset points of $p_i(x_i, y_i)$ and $p_{i+1}(x_{i+1}, y_{i+1})$ respectively, then line segments $p_i(x_i, y_i)\, p'_j(x'_j, y'_j)$ and $p_{i+1}(x_{i+1}, y_{i+1})\, p'_{j+1}(x'_{j+1}, y'_{j+1})$ both are perpendicular to the given line segment $p_i(x_i, y_i)\, p_{i+1}(x_{i+1}, y_{i+1})$. Normalized vector components x'_{perp} and y'_{perp} of $\overrightarrow{p_ip_{i+1}}$ are calculated based on

$$\begin{aligned} x'_{perp} &= \frac{(y_{i+1} - y_i) \times d}{\sqrt{(y_{i+1} - y_i)^2 + (x_{i+1} - x_i)^2}} \\ y'_{perp} &= \frac{-(x_{i+1} - x_i) \times d}{\sqrt{(y_{i+1} - y_i)^2 + (x_{i+1} - x_i)^2}}, \end{aligned} \tag{2}$$

where d is the robot radius which works as the offset distance. Offset coordinates x'_j and y'_j are found from the corresponding point (x_i, y_i) of the original polygon according to

$$\begin{aligned} x'_j &= x_i + x'_{perp} \\ y'_j &= y_i + y'_{perp}. \end{aligned} \tag{3}$$

For n lines of a polygon, we find n disconnected offset segments with $2n$ end points, as one point in the original polygon is corresponding to the end points of two offset lines. Starting point of a given line segment $p_i(x_i, y_i)$ is resulted with $p'_j(x'_j, y'_j)$ in d distance along its normal direction and the end point $p_{i+1}(x_{i+1}, y_{i+1})$ with $p'_{j+1}(x'_{j+1}, y'_{j+1})$ in the same d distance.

If the vertex is not convex, as in Fig. 1(c), PIDT test results positive as the offset lines of the adjacent two lines of non-convex vertex intersect each other. Since we are dealing with the outer offsetting only, it is most likely that, only non-convex vertices will cause two consecutive offset lines to intersect. For a non-convex vertex p_i, we find an intersection of two offset lines which causes a local invalid loop. We compute this intersection point to remove the local invalid loop caused by offset lines.

2.2 Offset Circular Arcs

Arcs are handled way differently than the lines. Input point sequence curve has arc end points, center and direction of the circle along with curve type. Start angle

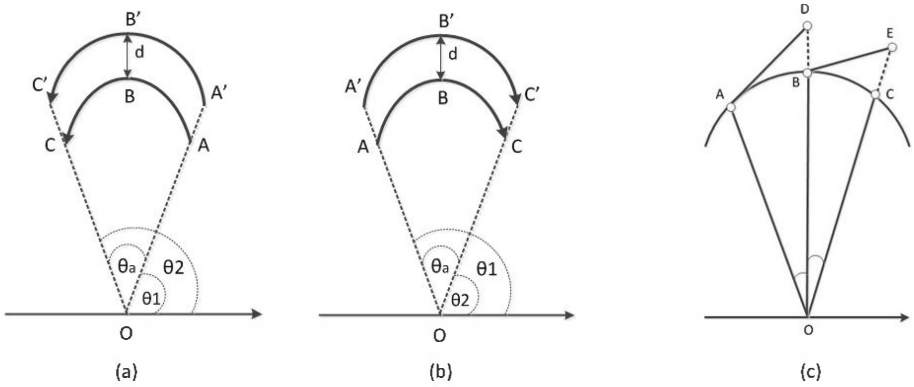


Fig. 2 Offsetting circular arcs: (a) CCW arc, $\theta_2 > \theta_1$, (b) CW arc, $\theta_2 < \theta_1$ and (c) draw the arc for offsets.

θ_1, end angle θ_2 and arc radius r are calculated from the given inputs. Local invalid loops are removed by invoking PIDT test.

According to Fig. 2 (a) and (b), a circular arc can be clockwise (CW) or counterclockwise (CCW). Points $A(x_1, y_1)$, $C(x_2, y_2)$ and $O(c_x, c_y)$ are the start, end and center points of the arc ABC. Points $A'(x_1', y_1')$ and $C'(x_2', y_2')$ are the corresponding offset points for A and C respectively. Start angle θ_1 is calculated from OA as $\theta_1 = \tan^{-1}(\frac{y_1 - c_y}{x_1 - c_x})$ and end angle θ_2 from OC based on $\theta_2 = \tan^{-1}(\frac{y_2 - c_y}{x_2 - c_x})$. Arc angle $\theta_a = \theta_2 - \theta_1$. For a CCW arc, $\theta_2 > \theta_1$ and for a CW arc $\theta_2 < \theta_1$. If θ_1 and θ_2 become negative, then $\theta_1 = \theta_1 + 2\pi$ and $\theta_2 = \theta_2 + 2\pi$.

Finding the arc radius r is straight forward from the arc center (c_x, c_y) and the start (or end) point of the arc as $r = \sqrt{(c_x - x_1)^2 + (c_y - y_1)^2}$. If we consider the circle equation as

$$x^2 + y^2 + 2gx + 2fy + c = 0, \tag{4}$$

where g, f and c are the constants, $(-g, -f)$ is the circle center and we can find the arc radius $OA = r$ from $\sqrt{g^2 + f^2 - c}$. However, radius of the offset arc OA' depends on arc direction. A CCW arc has its center inside the polygon area and the radius is $(r + d)$, where d is the offset distance. However, a CW arc center resides outside the polygon area and the radius is $(r - d)$. Following derivation is performed considering the arc direction towards CCW. So, $OA' = (r + d)$ and a similar derivation can be done for CW arc considering $OA' = (r - d)$.

Start and end points of the offset arc is calculated according to θ_1, θ_2 and r. Start point $A'(x_1', y_1')$ of the offset arc $A'B'C'$ is calculated as

$$\begin{aligned} x_1' &= c_x + (r + d)\cos\theta_1 \\ y_1' &= c_y + (r + d)\sin\theta_1. \end{aligned} \tag{5}$$

To find the next point to the (x_1', y_1'), we apply the theory of circular motion. During each calculation, we move the point tangentially to the circle for a fixed distance AD and BE and then back towards the center of the circle for another fixed distance DB and EC, as shown in Fig. 2(c). At this point we store the new location and repeat the process.

Let θ be the angular length of a segment ahead of the previous point and x_t is the tangent of θ. If N_s is the number of segments (500 in our case) in the arc between start angle and end angle, $\theta = \theta_a / N_s$. We can easily turn the vector $\overrightarrow{OA}$ by $\frac{\pi}{2}$ radians to CW direction that becomes tangent to the circle. Components of vector $\overrightarrow{OD}$ is calculated as

$$\begin{aligned} x_{new}' &= x_1' + x_t \tan\theta \\ y_{new}' &= y_1' + y_t \tan\theta, \end{aligned} \tag{6}$$

where (x_{new}', y_{new}') is the point D, θ is the angular length of a small segment of the arc, x_t and y_t are the tangent vector components of the radial vector (x_1, y_1). Vector $\overrightarrow{OB}$ is calculated as

$$\begin{aligned} x_{new} &= x_{new}' \cos\theta \\ y_{new} &= y_{new}' \cos\theta, \end{aligned} \tag{7}$$

where (x_{new}, y_{new}) is the point B. At this point we are back where we started and repeating this process will fill the entire angular length θ_a.

Now PIDT test will check whether this offset arc intersects with its neighboring offset segments. One of the three scenarios can happen: (i) both the offset segments attached to the offset arc came from same line, (ii) offset arc intersects two different lines in its two sides and (iii) an offset arc is a neighbor of another arc. If any intersection occurs, portion of the intersecting segments are discarded to remove local invalid loops.

3. GLOBAL INVALID LOOPS

Usually, invalid loops occur at the self-intersection points of raw-offset curves. Finding the intersections between two offset lines or between an offset line and an offset arc can prevent local invalid loops before generating the raw offset curve, but still global invalid loops may exist. Fig. 3 is a part of a pocket profile that shows a demonstration of two global invalid loops and final offsetting after removing them. Radius of the loop 1 is

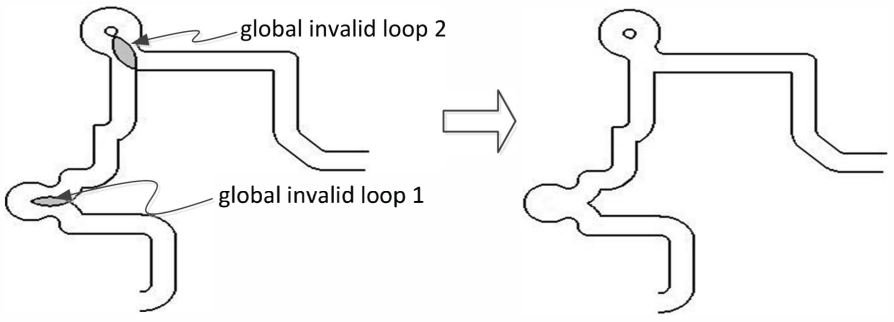


Fig. 3 Global invalid loops (grayed) and the final offset after removing them.

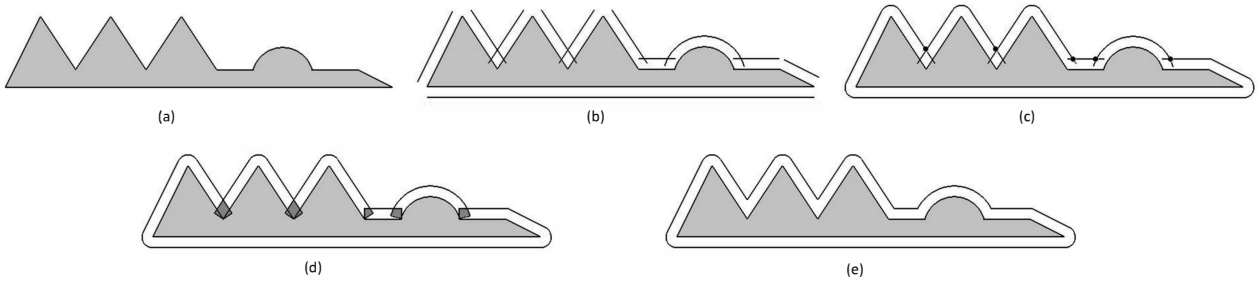


Fig. 4 Output of a polygon consists of convex vertex, non-convex vertex and arc: (a) input polygon (grayed), (b) disconnected offset segments for each lines and arcs, (c) for convex vertex, adjacent two offset points are connected by circular arcs of radius d. Self-intersection points caused by non-convex vertices and circular arc are indicated by black circles, (d) raw offset curve with local invalid loops (dark grayed) (e) final output.

$65.004mm$ and radius of the invalid loop 2 is $80.04mm$, while the offset distance d is $65mm$.

In a raw offset curve, it is always necessary to find the self-intersecting points to remove the global invalid loops. Instead of using the brute-force algorithm of $O(m^2)$, where m is the total number of segments in the raw offset curve, we use the well-known "Bentley-Ottmann Algorithm" [18] to find the global self-intersecting points which requires $O((n+k)\log m)$ time complexity, where n is the total number of vertices in the original polygon, k is the number of self-intersections and m is the number of valid segments in the raw offset curve and always $m \leq n$. We need to check for only the global self-intersection points instead of checking for all local and global self-intersection points, because local self-intersection points are already removed before generating the raw offset curve by PIDT test.

4. RESULTS AND ANALYSIS

4.1 Experimental Results

We implemented our algorithm in GLUT for Win32 version 3.7.6 on a personal computer with Pentium(R) D CPU 3.00GHz and 2.00 GB RAM. We have tested our algorithm for various types of polygons and it can successfully handle any shapes of geometric objects. In the literature section, we have mentioned several times that, most of the algorithms can not handle non-convex shapes and circular arcs. However, our method deals such problems very efficiently with minimum computational cost we have seen so far.

Fig. 4 demonstrates the whole process of our proposed method. Input polygon has total eleven vertices $(n = 11)$. Among them six are convex, three non-convex and remaining two are start and end point of the only arc. Fig. 4(a) is the original polygon. Fig. 4(b) shows disconnected offset segments for lines and arcs based on offset distance $d = 50$. Based on the type of the point sequence curve, angle between two neighboring segments are checked. For convex vertex, adjacent two new points of offset edges are connected by a circular arc centered in the corresponding vertex p_i as shown in Fig. 4(c). Pair-wise self-intersections are also shown for the non-convex vertices and circular arcs. A circular arc causes self-intersections with its two adjacent offset segments in its both ends. Fig. 4(d) is the raw offset curve for the original polygon with local invalid loops. Usually we remove the local invalid loops before generating the raw offset curves and in the raw offset curve, only global invalid loops remain. But in Fig. 4(d) local invalid loops are kept for better understanding of the whole process of getting the final output. After removing the local invalid loops, 4(e) shows the final output.

Together with Fig. 4, four more examples are given in Fig. 5 that shows the output of some random polygons consist of lines, arcs as well as convex and non-convex vertices. Fig. 5(a) is a comb which consists of 53 total points and among them 24 are non-convex. Fig. 5(b) is a random shape with 43 arc segments. Fig. 5(c) is a shape of a duck with 19 lines and 66 arc segments and Fig 5(d) is a complex shape in a PCB design process and consists of 7 non-convex vertices and 27 arcs. Poly-

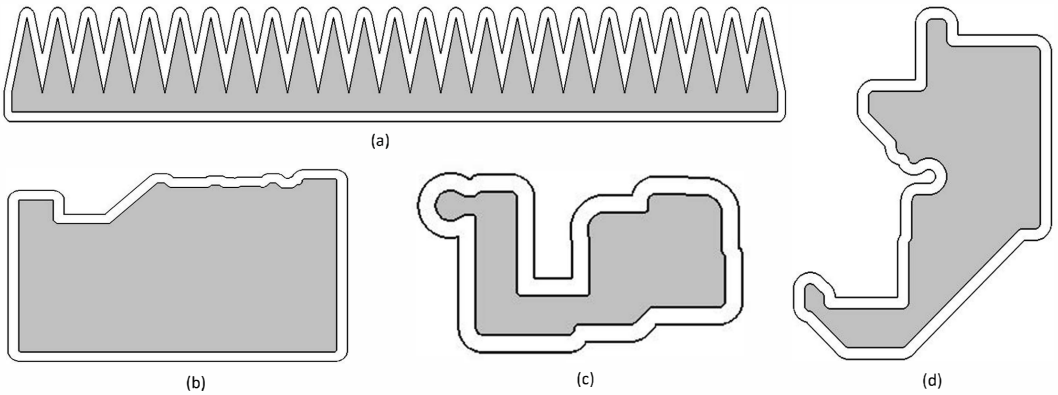
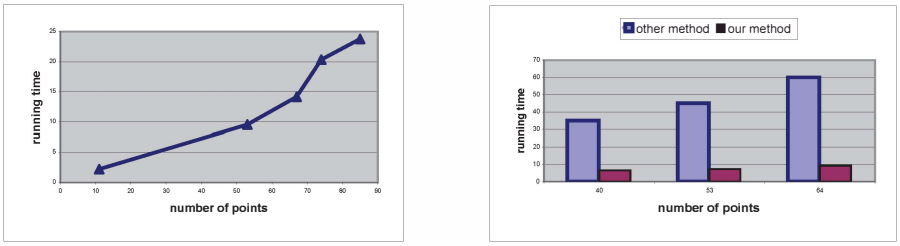


Fig. 5 Some arbitrary shaped polygons: (a) comb output, using offset distance $d = 150$, (b) a random shape, (c) shape of a duck, created with line and arc segments and (d) a complex polygonal shape in PCB design.

Table 1 Running times for various types of input polygons (taken as the average of multiple executions).

Input polygon	Size, n	Non-convex vertex, n_{nc}	Arc segment	Offset distance, d	Running time, ms
Output 1	11	3	1	75	2.21
Comb	53	24	0	150	9.55
Random	74	5	43	250	26.33
Duck	85	5	14	50	23.79
PCB	47	7	27	75	14.15



(a) (b)

Fig. 6 (a) running times for the outputs in Table 1 and (b) plotted result of Table 2, running time comparison with other methods.

gon statistics and algorithm execution times are shown in Table 1. Since the number of non-convex vertices and arc segments present in the polygon greatly influence the computation time, we mentioned them in the table. Fig. 6(a) is the plotted result of Table 1 that shows the running time is approximately linear to the input size n.

With the same experiment settings, table 2 shows comparison of our method with the approximate offset construction method described in [5]. Clearly, our algorithm performs better than the approximate construction of offset polygons. Fig. 6(b) is the plotted result of Table 2 that shows the superiority of our method with the previous ones. Our algorithm does not perform any redundant computation in the offsetting process, so it is way faster.

4.2 Complexity Analysis

Input of the algorithm is a polygon as point sequence curve and output is the offset polygon. The whole process performed in three steps: (1) raw offset curve generation, (2) checking for local invalid loops by invoking PIDT test and (3) removal of global invalid loops. To generate the raw offset curve, each vertex is inserted only once. Time complexity to construct raw offset curve is $O(n)$, where n is the number of vertices in the input point sequence curve.

For second part, three situations will cause local invalid loops: (i) non-convex vertex, (ii) type variation of two adjacent offset curves, for example a line is adjacent to arc or vice versa , and (iii) two adjacent offset arcs. A polygon has n total vertices and among them n_{nc} are non-convex and n_a vertices contain arcs, checking for local invalid loops with PIDT test will cost $O(n_{nc} + n_a)$ time and always $(n_{nc} + n_a) < n$.

In the third step, most of the time has been spent to report self-intersections. Time complexity to remove global invalid loops is $O((n + k) \log m)$, in which n is the total number of vertices in the original polygon, k is the number of self-intersections, and m is the number of segments in the raw offset curve and always $m \leq n$. In practice, number of global invalid loop is very small and makes the complexity $O(n \log m)$. Moreover, point se-

Table 2 Running times comparison with other methods

Input polygon	Size, n	Non-convex vertex, n_{nc}	Offset distance, d	Running times, ms	
				Approximate offset method	Our method
Comb	53	24	25	45	3.0
Wheel	40	14	50	35	2.55
Spiked	64	40	50	60	4.27

quence curves contain a large number of interfering segments and the value of m become much smaller than n ($m << n$), so the time for this step becomes approximately linear and so is the overall time complexity of the algorithm.

5. CONCLUSIONS

A new algorithm to offset arbitrary shapes for efficient motion planning of a circular robot is presented in this paper. The algorithm is able to deal with convex, non-convex as well as arcs in an object. Line and arc segments are treated differently and local invalid loops are removed before generating the raw offset curve. Our method is simple, mathematically well defined, and produces consistent results. It handles the non-convex shapes very efficiently where prior methods impose redundant computations. Overall time complexity of the algorithm is approximately linear.

REFERENCES

[1] L. J. Guibas and R. Seidel, "Computing convolutions by reciprocal search," *Discrete and Computational Geometry*, vol. 2, pp. 175–193, 1987.

[2] G. Varadhan, S. Krishnan, T. V. Sriram, and D. Manocha, "A simple algorithm for complete motion planning of translating polyhedral robots," *Int. Journal of Robotics Research*, vol. 24, no. 11, pp. 983–995, 2005.

[3] L. Zhiwei, F. Jianzhong, and G. Wenfeng, "A robust 2d point-sequence curve offset algorithm with multiple islands for contour-parallel tool path," *Computer-Aided Design*, vol. 45, pp. 657–660, 2013.

[4] B. Choi and S. Park, "A pair-wise offset algorithm for 2d point-sequence curve," *Computer Aided Design*, vol. 31, pp. 735–745, 1999.

[5] R. Wein, "Exact and approximate construction of offset polygons," *Computer-Aided Design*, vol. 39, pp. 518–527, 2007.

[6] T. N. Wong and K. W. Wong, "Toolpath generation for arbitrary pockets with islands," *Int.l Journal of Advanced Manufacturing Technology*, vol. 12, pp. 174–179, 1996.

[7] H.-C. Kim, "Tool path generation for contour parallel milling with incomplete mesh model," *Int. J. of Advanced Manufacturing Technology*, vol. 48, pp. 443–454, 2010.

[8] H.-C. Kim, S.-G. Lee, and M.-Y. Yang, "A new offset algorithm for closed 2d lines with islands," *Int. J. of Advanced Manufacturing Technology*, vol. 29, pp. 1169–1177, 2006.

[9] Q. Bo, "Recursive polygon offset computing for rapid prototyping applications based on voronoi diagrams," *Int. J. of Advanced Manufacturing Technology*, vol. 49, pp. 1019–1028, 2010.

[10] M. Held, "Voronoi diagrams and offset curves of curvilinear polygons," *Computer Aided Design*, vol. 30, no. 4, pp. 287–300, 1998.

[11] M. Held and S. Huber, "Topology-oriented incremental computation of voronoi diagrams of circular arcs and straight-line segments," *Computer-Aided Design*, vol. 41, pp. 327–338, 2009.

[12] S. McMains, J. Smith, J. Wang, and C. Sequin, "Layered manufacturing of thin-walled parts," in *Proceedings of ASME design engineering technical conference*, Baltimore (MD), 2000.

[13] X. Chen and S. McMains, "Polygon offsetting by computing winding numbers," in *Proc. of ASME Int. Design Engineering Tech. Conf. (IDETC)*, 2005.

[14] W. Esquivel and L. Chaiang, "Nonholonomic path planning among obstacles subject to curvature restrictions," *Robotica*, vol. 20, no. 1, pp. 49–58, 2002.

[15] R. Wein, J. P. Berg, and D. Halperin, "The visibility-voronoi complex and its applications," in *SCG Proc. of the twenty-first annual symposium on Computational Geometry*, 2005, pp. 63–72.

[16] J. T. Schwartz and M. Sharir, "On the piano movers problem: Ii. general techniques for computing topological properties of real algebraic manifolds," *Advances of Applied Maths*, vol. 4, pp. 298–351, 1983.

[17] K. Kedem and M. Sharir, "An automatic motion planning system for a convex polygonal mobile robot in 2-d polygonal space," in *ACM Symposium on Computational Geometry*, Urbana-Champaign, IL, 1998, pp. 329–340.

[18] J. L. Bentley and T. Ottmann, "Algorithms for reporting and counting geometric intersections," *IEEE Transactions on Computers*, vol. C-28, no. 9, pp. 643–647, 1979.