IECON2015-Yokohama
November 9-12, 2015

Coverage Path Planning Based on a Multiple Sweep Line Decomposition

Xin Yu
Electrical and Computer Engineering
Auburn University, AL, USA
Email: xzy0008@auburn.edu

John Y. Hung
Electrical and Computer Engineering
Auburn University, AL, USA
Email: hungjoh@auburn.edu

***Abstract*—Coverage path planning determines a path that guides an autonomous vehicle to pass every part of a workspace completely and efficiently. Since turns are often costly for autonomous vehicles, minimizing the number of turns usually produces more working efficiency. This paper presents an optimization approach to minimize the number of turns of autonomous vehicles in coverage path planning. For complex polygonal fields, the problem is reduced to finding the optimal decomposition of the original field into simple subfields. The optimization criterion is minimization of the sum of widths of these decomposed subfields. Here, a new algorithm is designed based on a multiple sweep line decomposition. The time complexity of the proposed algorithm is $O(n^2 \log n)$. Experiments show that the proposed algorithm can provide nearly optimal solutions very efficiently when compared against recent state-of-the-art. The proposed algorithm can be applied for both convex and non-convex fields.**

***Index Terms*—Coverage Path Planning, Sweep Line Algorithm, Mobile Robots, Dubins Vehicles.**

I. Introduction

Coverage path planning is a special type of path planning, which requires the robot to determine a path that passes all points of an area. It is a common challenge in many industrial applications, and extensively studied in recent years. Examples include demining robots [1], autonomous lawn mowers [2], indoor service robots [3]–[7], exploration robots [8]–[10], autonomous underwater vehicles [11] and automated harvesters [12]. Several solutions to the coverage path planning have been reviewed and categorized in surveys [13], [14]. The surveys show that most existing algorithms adopt cellular decomposition of the given field to achieve the provable guarantee of complete coverage. A cellular decomposition finds efficient ways to subdivide the given field into cells that can be easily traversed by a coverage path.

Based on the main method they use, cellular decompositions can be subdivided into three types: approximate, semi-approximate and exact. Approximate cellular decomposition approximates the target field by using cells of same size and shape. Semi-approximate decomposition uses cells with fixed width to approximate the target field, but the top and bottom of cells can have any shape. Exact cellular decomposition uses a set of non-intersecting cells, and the union of cells exactly fills the target field.

In this paper, only exact cellular decomposition is considered. The exact cellular decomposition algorithms usually include three procedures: (1) decomposition of the complex coverage field into subfields with parallel tracks; (2) selection of a traversal sequence of those subfields; and (3) generation of a boustrophedon path (straight parallel paths with alternate directions) that covers each subfield individually.

One popular exact cellular decomposition technique is the trapezoidal decomposition [15], in which the free space is decomposed into trapezoidal cells. Since each cell is a trapezoid, coverage in each cell can be easily achieved with the boustrophedon path. Coverage of the field is achieved by visiting each cell in the adjacency graph. The shortcoming of this method is that it requires too many redundant turns to guarantee complete coverage. Since turns are often costly and considered as non-working time, minimizing the cost of turns usually produces higher working efficiency.

Choset and Pignon [16] develop a boustrophedon decomposition to reduce the redundant turns. In this method, a line segment, termed a slice, is swept through the environment. Whenever there is a change in connectivity of the slice, a new cell is formed. When the connectivity increases, two new cells are spawned. Conversely, when connectivity decreases, two cells are merged into one cell. The tracks in each cell are parallel to the slice.

Huang [17] introduces a decomposition algorithm to minimize the total number of turns required to cover a field. The algorithm adopts multiple line sweeps to divide the coverage field into cells, then combines cells into larger subfields by dynamic programming, and finally assigns each subfield a sweep direction according to the minimum sum of altitudes. The parallel tracks in each subfield are perpendicular to the sweep direction of that subfield.

Oksanen and Visala [12] propose an algorithm that incrementally decomposes the field into subfields using trapezoidal decomposition, merges small subfields into larger ones, then searches for the merged subfield with the best cost and removes it from the original field. In searching for the subfield with best cost, the layout of parallel tracks is determined by a heuristic approach. The process is repeated for the remaining field until the whole field is computed.

Fang and Anstee [18] propose an iterative decomposition scheme based on the generalized Voronoi diagram [19]. They firstly compute an approximate generalized Voronoi diagram of the given field, then apply boustrophedon decomposition along the longest line segment of the approximate generalized Voronoi diagram, select the subfields that contain the longest line segment, and remove these subfields with well planned path. The algorithm is repeated for the remaining field until the whole field is fully covered.

Jin and Tang [20] adopt a divide-and-conquer strategy to the decomposition. The algorithm firstly searches the optimal layout of parallel tracks without any decomposition, then finds "all possible ways" to splitting the field into two and for each possible way sees if the field would be more efficient as two subfields instead of one. The algorithm is implemented recursively on each subfield until there is no valid decomposition that achieves a better solution.

Li et al. [21] propose a decomposition algorithm to minimize the number of turns based on a greedy recursive method. The process recursively decomposes the field into two subfields until no concave field remains. In each decomposition step, the criterion of optimization is the minimum width sum of two subfields. The final tracks in each subfield are perpendicular to the width of that subfield. The algorithm is proven to have polynomial time complexity.

In this paper, the authors also propose a polynomial time algorithm to minimize the number of turns in coverage path planning, and the time complexity is greatly improved comparing to the existing algorithms. The remainder of this paper is organized as follows. In Section II, the problem of coverage path planning is transformed into width calculation of the coverage field and the problem statement is formally introduced. A linear time algorithm for convex fields is described in Section III. A polynomial time algorithm for non-convex fields is designed in Section IV. In Section V, the proposed algorithm is compared with existing algorithms and a practical experiment with real field data is also conducted. Section VI summarizes the key results in this paper.

II. Problem Statement

The goal of this paper is to find a path to completely cover a field by a vehicle and the travel distance of the vehicle must be minimized. Since boustrophedon method is the most common method to cover a simple field, this paper will adopt parallel tracks that can be used by boustrophedon method (or other similar methods) to cover the field. Based on this assumption, the travel distance of the vehicle consists of the distance along tracks and the distance to turn at the end of tracks.

Before giving further problem statement, the authors firstly introduce the concept of altitude and width of convex polygons as follows:

Definition 2.1: Given a convex polygons P, a line of support L is a line intersecting P and such that the interior of P lies to one side of L.

Definition 2.2: The altitude A of a convex polygon P is the shortest distance between a pair of parallel lines of support $(L_1,\ L_2)$.

Definition 2.3: The width W of a convex polygon is the minimum altitude of that polygon.

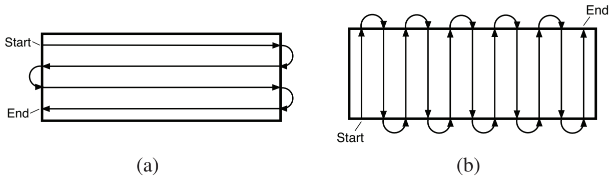


Fig. 1. Different track directions for convex fields. [17]

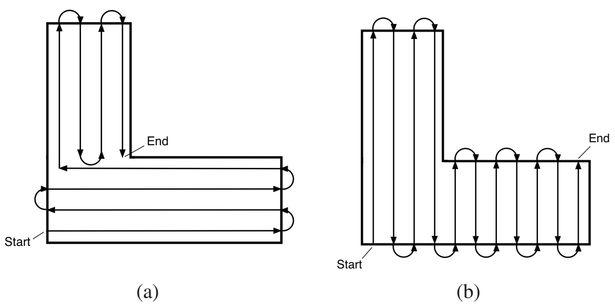


Fig. 2. Different track directions for non-convex fields. [21]

If the field is convex and does not contain any obstacles, the coverage planning with parallel tracks is quite simple. The main task is to find the optimal orientation of the parallel tracks. By deeper analysis, the problem can be further reduced. As illustrated in Fig. 1, covering the field in different directions can produce nearly the same distance along the tracks, but can produce a large difference in number of turns, which means a large difference in distance on turns. Therefore, the total travel distance mainly depends on the number of turns, i.e. the total distance will decrease as the number of turns decreasing. Furthermore, the number of turns in a given track direction is also proportional to the altitude of the convex polygonal field in that direction. Therefore, the problem can be reduced to search the minimum altitude (width) of the field and its corresponding direction. The parallel tracks can be generated along the vertical direction of width.

If the field is non-convex (concave or with obstacles), finding the optimal solution is hard. One possible strategy is to decompose the complex field into convex subfields. Each subfield can be covered by parallel tracks in a different direction. Fig. 2 shows an example that assigns each subfield a different track direction results in a better solution than applying only one track direction to the whole fields. Furthermore, the minimum number of turns in each subfield can be determined by the width of the subfield. To obtain an optimal solution for the non-convex field, the total sum of widths must be minimized.

The coverage problem becomes to find a convex decomposition of the non-convex field that has the minimum sum of widths. The authors refer such a decomposition as the Minimum Sum of Widths (MSW) Decomposition. Let $\mathcal{P}$ be the polygon that represents the non-convex field with n vertices. In a convex decomposition $\mathcal{D}$, the polygon $\mathcal{P}$ is decomposed into $m\ (m \leq n)$ convex polygons $\mathcal{P}_1, \mathcal{P}_2, \ldots, \mathcal{P}_m$, whose widths

Algorithm 1 Width of a Convex Polygon

```
Input: Vertex list of the given convex polygon in counter-clockwise order
Output: Width of the polygon and direction of the corresponding parallel lines of support
 1: Delete middle vertices of any collinear sequence of three vertices
 2: $V_a$ ←Vertex with minimum y-coordinate
 3: $V_b$ ←Vertex with maximum y-coordinate
 4: $RotatedAngle \leftarrow 0$
 5: $Angle \leftarrow 0$
 6: $Width \leftarrow \infty$
 7: $Caliper_a$ ←Unit vector along positive x-axis
 8: $Caliper_b$ ←Unit vector along negative x-axis
 9: while $RotatedAngle < \pi$ do
10:     $E_a$ ←Edge from $V_a$ to its next adjacent vertex
11:     $E_b$ ←Edge from $V_b$ to its next adjacent vertex
12:     $A_a$ ←Angle between $Caliper_a$ and $E_a$
13:     $A_b$ ←Angle between $Caliper_b$ and $E_b$
14:     $Altitude \leftarrow 0$
15:     if $A_a < A_b$ then
16:         Rotate $Caliper_a$ by $A_a$
17:         Rotate $Caliper_b$ by $A_a$
18:         $V_a$ ←The next adjacent vertex of $V_a$
19:         $Altitude$ ←Distance from $vertex_b$ to $Caliper_a$
20:         $RotatedAngle \leftarrow RotatedAngle + A_a$
21:     else
22:         Rotate $Caliper_a$ by $A_b$
23:         Rotate $Caliper_b$ by $A_b$
24:         $V_b$ ←The next adjacent vertex of $vertex_b$
25:         $Altitude$ ←Distance from $vertex_a$ to $caliper_b$
26:         $RotatedAngle \leftarrow RotatedAngle + A_b$
27:     end if
28:     if $Altitude < Width$ then
29:         $Width \leftarrow Altitude$
30:         $Angle \leftarrow RotatedAngle$
31:     end if
32: end while
33: return $Width$ and $Angle$
```

are $\mathcal{W}_1, \mathcal{W}_2, \ldots, \mathcal{W}_m$ respectively. Let $S(\mathcal{D})$ be the sum of width of $\mathcal{D}$. Then the problem can be stated more formally:

Problem 2.4 (MSW Decomposition):

$$\begin{aligned} \underset{\mathcal{D}}{\text{minimize}} \quad & S(\mathcal{D}) = \sum_{i=1}^{m} \mathcal{W}_i \\ \text{subject to} \quad & \mathcal{P}_i \in \text{convex polygon} \end{aligned}$$

Note that the optimal coverage of a convex field can be seen as a degenerate of the above problem statement where $m = 1$.

III. Coverage of Convex field

As described in the previous section, the optimal coverage of a convex field can be determined by the width of that polygon. However, a convex polygon admits parallel lines of support in any direction, and for each direction the altitude is usually different. Fortunately, not all directions need to be examined to determine the width. Suppose a convex polygon is given, along with two parallel lines of support. If neither of these lines coincides with an edge, it is always possible to rotate them to decrease the distance between them. Therefore, the width of polygon can be determined by examining only the edge orientations. A formal proof of this result can be found in [17].

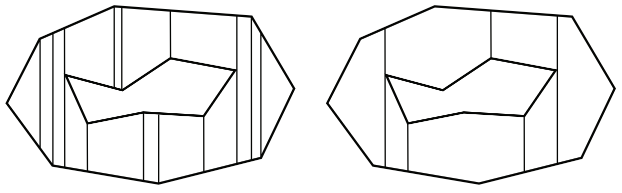


Fig. 3. (a) Trapezoidal decomposition. (b) The proposed convex decomposition.

Based on this result, a linear time complexity algorithm is given in [22]. The algorithm adopts rotating calipers, which is a method used to construct efficient algorithms for a number of computational geometry problems. The method is analogous to a vernier caliper that rotates around the outside of a convex polygon. Every time one blade of the caliper lies flat against an edge of the polygon, it forms an antipodal pair with the point or edge touching the opposite blade. The complete rotation of the caliper around the polygon detects all antipodal pairs and can be carried out in $O(n)$ time. The process is described in Algorithm 1. The unit of angle is radian.

After the width is calculated, the minimum number of turns N_{turn} can be determined by

$$N_{turn} = \lceil \frac{W}{d} \rceil \tag{1}$$

where W is width of the given convex polygon, d is the space between two adjacent tracks and ceiling symbol $\lceil * \rceil$ is the smallest integer not less than $*$. [21] Then the N_{turn} parallel tracks can be generated in the direction $Angle$ that is returned by the algorithm.

IV. Coverage of Non-convex field

Firstly a new convex decomposition method is designed based on the line sweeping method in one sweeping direction (IV-A) and the optimal coverage for each convex subfield is searched (IV-B). Then, for each edge orientation (including edges of the polygon and edges of the obstacle), the authors apply the designed convex decomposition and obtain the sum of widths. After that, the convex decomposition with minimum sum of widths is selected (IV-C). To avoid unnecessary tracks to cover the subfields, the resulting decomposition are finally refined by merging adjacent similar convex polygons (IV-D).

A. Convex Decomposition

The proposed convex decomposition method is an enhancement of the trapezoidal decomposition [15] and is designed to reduce unnecessary cells. As illustrated in Fig. 3a, the trapezoidal decomposition comprises cells that are shaped like trapezoids or triangles (which can be seen as degenerate trapezoids). To improve time complexity, it adopts a sweep line method and treats each vertex as an event. To form the decomposition, a vertical line is swept from left to right through the polygon field. When an event is encountered, it extends rays upward and downward through the free space of the polygon field until an edge that lies immediately above and below the event is hit. Many events will have either just an upward ray or a downward ray. Trapezoidal cells are formed at the event depending on the event type. Once the sweep line finishes the rightmost event, a trapezoidal decomposition results. However, the drawback of trapezoidal decomposition is that it produces too many redundant convex cells. Some adjacent small convex cells can be merged into a larger convex cells in the sweep line process, as shown in Fig. 3b. What follows is an improvement of trapezoidal decomposition that reduces the redundant convex cells.

1) Events: In trapezoidal decomposition, all vertices are classified into five types of events: *OPEN, CLOSE, SPLIT, MERGE* and *INFLECTION*. These types of events are defined as follows:

Definition 4.1: A vertex v is an *OPEN* event if its two neighbor vertices lie on the right side of the sweep line and the interior angle at v is less than π; if the interior angle is greater than π, then v is a *SPLIT* event. A vertex is a *CLOSE* event if its two neighbor vertices lie on the left side of the sweep line and the interior angle at v is less than π; if the interior angle is greater than π, then v is a *MERGE* event. A vertex is an *INFLECTION* event if its two neighbor vertices lie on opposite sides of the sweep line.

In the proposed convex decomposition, the *INFLECTION* event is replaced with four more types of events: *FLOOR_CONVEX, FLOOR_CONCAVE, CEIL_CONVEX, CEIL_CONCAVE* which are defined as follows:

Definition 4.2: A vertex v is a *FLOOR_CONVEX* event if its previous neighbor vertex v_{prev} lies on left side of the sweep line while its next neighbor v_{next} lies on right side of the sweep line, and the interior angle at v is less than π; if the interior angle is greater than π, then v is a *FLOOR_CONCAVE* event. On the contrary, a vertex v is a *CEIL_CONVEX* event if its previous neighbor vertex v_{prev} lies on right side of the sweep line while its next neighbor v_{next} lies on left side of the sweep line, and the interior angle at v is less than π; if the interior angle is greater than π, then v is a *CEIL_CONCAVE* event.

These definitions are based on the assumption that vertices of the polygon are listed in counter-clockwise order and vertices of holes (obstacles) are in clockwise order. Then the interior of the polygon is always to the left of each edge when following the order. Examples of eight event types are illustrated in Fig. 4.

2) Sweep Line Algorithm: Next, the sweep line algorithm is applied to form the decomposition. The events are firstly sorted based on their x-coordinates in an ascending order. During the sweeping process, a balanced binary search tree L is used to maintain the "current" edges that the sweep line intersects. A cell in the algorithm can be represented by two lists: ceiling list and floor list, both of which bound the cell. The sweep line algorithm starts from left to right, and visits each event in order. When the sweep line encounters an event, different operations are made depending on the type of event:

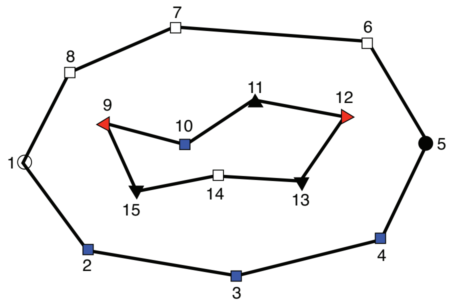


Fig. 4. Eight event types: *OPEN* (1), *CLOSE* (5), *SPLIT* (9), *MERGE* (12), *FLOOR_CONVEX* (2, 3, 4, 10), *FLOOR_CONCAVE* (11), *CEIL_CONVEX* (6, 7, 8, 14) and *CEIL_CONCAVE* (13, 15). The sweep line is horizontally swept from left to right.

OPEN event: Two incident edges of this event are inserted into L. A new cell is opened and the vertex of this event is added to floor list of the cell.

SPLIT event: The edges immediately above and below this event are searched in L. Then the intersection of the sweep line and the above edge, and the intersection of the sweep line and the below edge are determined. Find the cell to which this event belongs. Add the above and below intersection points into ceiling list and floor list of the cell respectively. Now the current cell is considered to be closed. After that, two new cells are opened. Add vertex of this event into floor list of the top new cell and ceiling list of the bottom new cell respectively. Add the above intersection point into ceiling list of the top new cell and the below intersection point into floor list of the bottom new cell. After the cell operations, two incident edges of this event are inserted into L.

FLOOR_CONVEX event: Find the cell to which this event belongs. Add the vertex of this event into floor list of the current cell. Delete the left incident edge of this event from L and insert the right incident edge of this event into L.

CEIL_CONVEX event: Find the cell to which this event belongs. Add the vertex of this event into ceiling list of the current cell. Delete the left incident edge of this event from L and insert the right incident edge of this event into L.

FLOOR_CONCAVE event: Delete the left incident edge of this event from L. Search the edge that is immediately above this event in L, and determine the intersection point of the sweep line and the above edge. Then add the right incident edge of this event into L. Find the cell to which this event belongs. Add the vertex of this event into floor list of the current cell and the intersection point into ceiling list of the current cell respectively. Now the current cell is considered to be closed. After that, a new cell is opened. Add the vertex of this event into floor list of the new cell and the intersection point into ceiling list of the new cell respectively.

CEIL_CONCAVE event: Delete the left incident edge of this event from L. Search the edge that is immediately below this event in L, and determine the intersection point of the sweep line and the below edge. Then add the right incident edge of this event into L. Find the cell to which this event belongs. Add the vertex of this event into ceiling list of the current cell and the intersection point into floor list of the current cell respectively. Now the current cell is considered to be closed. After that, a new cell is opened. Add the vertex of this event into ceiling list of the new cell and the intersection point into floor list of the new cell respectively.

MERGE event: Two incident edges of this event are deleted from L. The edges immediately above and below this event are searched in L. Then the intersection of the sweep line and the above edge, and the intersection of the sweep line and the below edge are determined. Find the two cells to which this event belongs. Add the vertex of this event into floor list of the top cell and ceiling list of the bottom cell respectively. Add the above intersection point into ceiling list of the top cell and the below intersection point into floor list of the bottom cell respectively. Now the two cells are considered to be closed. After that, a new cell is opened. Add the above intersection point into ceiling list of the new cell and the below intersection point into floor list of the new cell respectively.

CLOSE event: Two incident edges of this event are deleted from L. Find the cell to which this event belongs. The vertex of this event is added to floor list of the current cell and the current cell is considered to be closed.

After all events are visited, the polygonal field is decomposed into a list of convex cells. The adjacency graph of these cells can also be determined in the process. The main difference between the proposed convex decomposition and trapezoidal decomposition is at the *FLOOR_CONVEX* and *CEIL_CONVEX* events. At these two events, the proposed decomposition doesn't open or close a cell, but rather just updates the current cell. Note that the above description of sweep line algorithm assumes that the x-coordinates of all events are distinct. For general cases, the assumption can be achieved by rotating the coordinate system in a sufficiently small amount.

B. Optimal Coverage for Each Convex Polygon

In one sweeping direction, the field is decomposed into convex sub-polygons by applying the proposed convex decomposition. The width of each sub-polygon can then be determined independently by applying the method described in Sec. III. Also, the optimal orientation of parallel tracks in each sub-polygon can be determined independently. Then the sum of width of these polygons can be calculated.

C. Sweep Direction

Until this section, the convex decomposition is done in a particular sweeping direction. However, the best sweep direction is not known and has to be searched. In [17], the author shows that the best sweep direction is perpendicular to one of the boundary or obstacle edges, if all tracks are perpendicular to the sweep direction. In this paper, the authors also assume that the best sweep direction is perpendicular to one of these edges. So only sweep directions that are perpendicular to the boundary or obstacle edges are examined. For each sweep direction, the field is decomposed into convex sub-polygons and the width of each sub-polygon can be determined independently. Among all the possible sweep directions, the convex decomposition with minimum sum of width is selected. By now, the minimum sum of widths decomposition is done.

D. Merging Adjacent Polygons

Since each sub-polygon is covered independently, it may produce redundant tracks to cover two adjacent sub-polygons, when parallel tracks of these two adjacent sub-polygons have the same track orientation [16]. To avoid the redundant tracks, two convex sub-polygons are merged if they have the same track orientation and are entirely adjacent to each other [21]. The definition of adjacency and entire adjacency of two polygons are described as follows:

Definition 4.3 ([21]): Consider two polygons P_1 with n vertices and P_2 with m vertices. If an edge of P_1, $v_{1i}v_{1(i+1)}$ $(i \in [1,n])$, coincides with an edge of P_2, $v_{2j}v_{2(j+1)}$ $(j \in [1,m])$, P_1 is adjacent to P_2.

Definition 4.4 ([21]): P_1 and P_2 are adjacent to each other, whose coincidence edges are $v_{1i}v_{1(i+1)}$ and $v_{2j}v_{2(j+1)}$ respectively. If $v_{1i} = v_{2j}$ (or $v_{1i} = v_{2(j+1)}$) and $v_{1(i+1)} = v_{2(j+1)}$ (or $v_{1(i+1)} = v_{2j}$), P_1 is entirely adjacent to P_2.

With the decomposition and adjacency graph, the planner determines a merge process into four steps. First, assign each sub-polygon an unique group number. Secondly, iterate all these sub-polygons. For each sub-polygon, test it with all its neighbors to see whether these two polygons satisfy the merging condition. If the neighbor sub-polygon satisfies, change the group number of the neighbor sub-polygon to that of the current sub-polygon. Thirdly, sort the sub-polygons according to their group numbers. Finally, iterate the ordered sub-polygons and merge the sub-polygons that have the same group number.

Then parallel tracks can be generated in each sub-polygon according to their track orientations.

E. Time Complexity Analysis

Let n be the number of vertices of the input polygonal field. In the proposed convex decomposition (IV-A), the first step is to sort the events based on the x-coordinates. This takes $O(n \log n)$ time. Then in the sweep line process, for each event, determining an edge that is immediately above or below the event takes $O(\log n)$ time if the edges are stored in a balanced binary search tree. So for all events, the sweep line process takes $O(n \log n)$ time. Since each sub-polygon takes linear time to calculate the width and the total number of vertices of these sub-polygons is linear to n, it takes $O(n)$ time to determine the sum of width of these sub-polygons (IV-B). As long as the sweep line process takes $O(n \log n)$

in one sweeping direction, examining sweep directions that perpendicular to all edges of the polygon takes $O(n^2 \log n)$ time (IV-C). In the merging process (IV-D), the number of sub-polygons and the number of edges in the adjacency graph are both linear to the number of vertices n. The first step takes $O(n)$ time. In the second step, examining the entire adjacency of two polygons can be taken in $O(n \log n)$. To test the entire adjacency for all sub-polygons, edges in the adjacency graph will be visited twice which is linear to n. So the second step requires $O(n^2 \log n)$. Sorting in the third step requires $O(n \log n)$ time. In the last step, merging of two sub-polygons takes $O(n \log n)$ time. To merge all sub-polygons that have the same group number, it requires $O(n^2 \log n)$ time. So the whole merging process takes $O(n^2 \log n)$ time.

Therefore, the total time complexity of the proposed algorithm in this paper is $O(n^2 \log n)$.

V. Test Results

The algorithm is implemented in C++ and tested on a computer with 1.3 GHz CPU and 4 GB RAM. To test the performance, the result of the proposed algorithm is compared with two former researchers' results. Both former algorithms use sum of widths as cost function to search the optimal decomposition. Fig. 5a is an example given by Huang [17], who adopts dynamic programming technique to search the optimal decomposition. Since the dynamic programming in [17] searches all possible decompositions, the solution in Fig. 5a can be seen optimal in such case. Fig. 5b shows the solution generated by the proposed algorithm. By comparing the total width of these two solutions, the total width of the proposed algorithm is 3.5% greater than that of the optimal solution. Note that the method in Huang's work [17] needs exponential time to obtain the optimal solution. However, the proposed algorithm requires only $O(n^2 \log n)$ time, which is much faster to obtain the nearly optimal result. The computational time for the proposed solution is 0.10 second as an average of 10 running times.

Fig. 6a is an example given by Li et al. [21], who design a greedy recursive method for the decomposition. Fig. 6b shows the solution generated by the proposed algorithm. The total width of the proposed algorithm is 3.6% less than that of the greedy recursive method in [21]. Also the proposed algorithm requires less time to obtain the result, since the greedy recursive algorithm in [21] requires $O(n^4)$ time. The computational time for the proposed solution is 0.09 second as an average of 10 running times.

Another experiment field [32°35′29.5″N 85°29′31.3″W], shown in Fig. 7a, has an area of 6.65 acres (or 26896.73 m^2) and has one obstacle area. The solution of the proposed algorithm is shown in Fig. 7b. The space between two adjacent parallel tracks is set to 2 meters. The solution shows that the optimal sweeping direction is 87.40° and the field is decomposed into 8 sub-polygons. The computational time for the proposed solution is 0.25 second as an average of 10 running times.

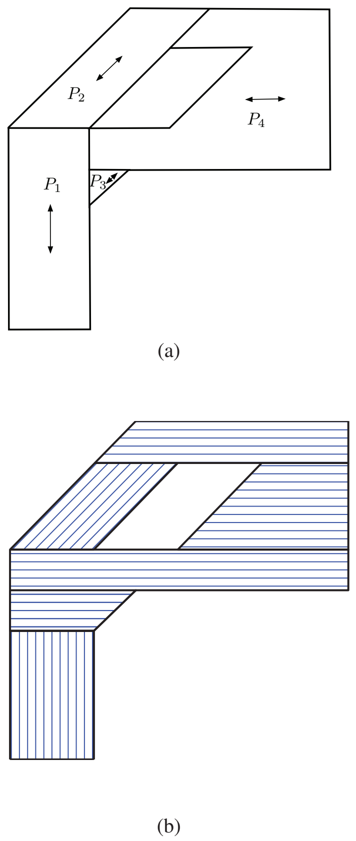


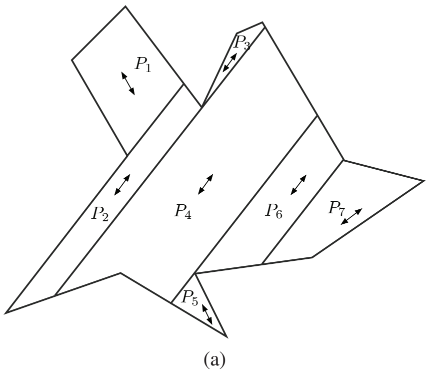
Fig. 5. (a) Solution of Huang's algorithm [17]. Arrows indicate the track directions. (b) Solution of the proposed algorithm.

VI. Conclusion

In this paper, the authors try to minimize the number of turns in coverage path planning. The problem is reduced to find the optimal decomposition of a complex field into convex subfields. The criterion of optimization is the sum of width of these decomposed subfields. Firstly, a new convex decomposition algorithm in one sweeping direction is designed based on sweep line method. Then the convex decomposition is performed in all directions that are perpendicular to the edges, and the convex decomposition with minimum sum of width is selected. To avoid unnecessary tracks to cover the subfields, the resulting decomposition is finally refined by merging adjacent similar convex sub-polygons. The time complexity of this algorithm is $O(n^2 \log n)$. The proposed algorithm is compared with two state-of-the-art algorithms. The results show that the proposed algorithm can produce a nearly optimal solution with much greater efficiency. Another experiment, based on real field data, shows that the proposed algorithm can produce feasible and effective solution for a large area field containing obstacles.

References

[1] D. W. Hodo, D. M. Bevly, J. Y. Hung, S. Millhouse, and B. Selfridge, "Optimal path planning with obstacle avoidance for autonomous surveying," in *Proc. Annu. Conf. IEEE Ind. Electron. Soc.*, Glendale, AZ, Nov. 2010, pp. 1577–1583.



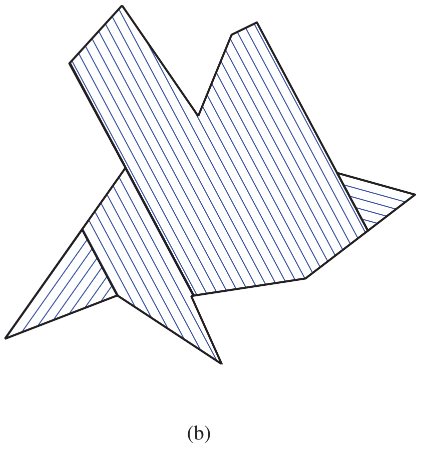
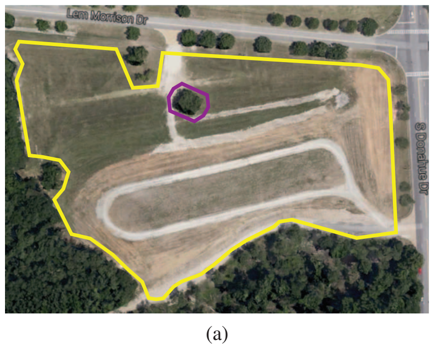


Fig. 6. (a) Solution of Li's algorithm [21]. Arrows indicate the track directions. (b) Solution of the proposed algorithm.



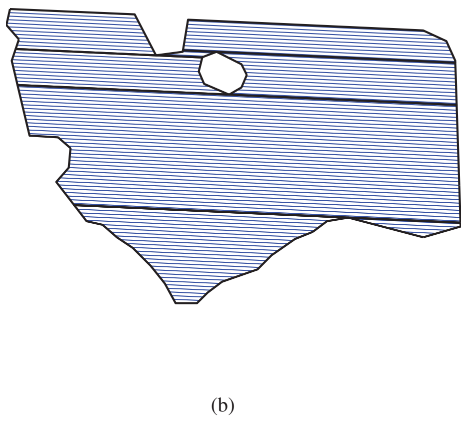


Fig. 7. Test field near Auburn University and solution of the proposed algorithm.

[2] B. Shiu and C. Lin, "Design of an autonomous lawn mower with optimal route planning," in *Proc. IEEE Int. Conf. Ind. Technology*, Chengdu, China, Apr. 2008, pp. 1–6.

[3] J. S. Oh, Y.-H. Choi, J.-B. Park, and Y. Zheng, "Complete coverage navigation of cleaning robots using triangular-cell-based map," *IEEE Trans. Ind. Electron.*, vol. 51, no. 3, pp. 718–726, June 2004.

[4] S. X. Yang and C. Luo, "A neural network approach to complete coverage path planning," *IEEE Trans. Syst., Man, Cybern. B*, vol. 34, no. 1, pp. 718–724, Feb. 2004.

[5] G. Kim and W. Chung, "Tripodal schematic control architecture for integration of multi-functional indoor service robots," *IEEE Trans. Ind. Electron.*, vol. 53, no. 5, pp. 1723–1736, Oct 2006.

[6] C. Luo and S. X. Yang, "A bioinspired neural network for real-time concurrent map building and complete coverage robot navigation in unknown environments," *IEEE Trans. Neural Netw.*, vol. 19, no. 7, pp. 1279–1298, July 2008.

[7] C.-H. Kuo, H.-C. Chou, and S.-Y. Tasi, "Pneumatic sensor: A complete coverage improvement approach for robotic cleaners," *IEEE Trans. Instrum. Meas.*, vol. 60, no. 4, pp. 1237–1256, Apr. 2011.

[8] E. Acar, H. Choset, and J. Lee, "Sensor-based coverage with extended range detectors," *IEEE Trans. Robot.*, vol. 22, no. 1, pp. 189–198, Feb. 2006.

[9] A. Yazici, G. Kirlik, O. Parlaktuna, and A. Sipahioglu, "A dynamic path planning approach for multirobot sensor-based coverage considering energy constraints," *IEEE Trans. Cybern.*, vol. 44, no. 3, pp. 305–314, Mar. 2014.

[10] A. Das, M. Diu, N. Mathew, C. Scharfenberger, J. Servos, A. Wong, J. S. Zelek, D. A. Clausi, and S. L. Waslander, "Mapping, planning, and sample detection strategies for autonomous exploration," *J. Field Robotics*, vol. 31, no. 1, pp. 75–106, Jan. 2014.

[11] L. Paull, S. Saeedi, M. Seto, and H. Li, "Sensor-driven online coverage planning for autonomous underwater vehicles," *IEEE/ASME Trans. Mechatronics*, vol. 18, no. 6, pp. 1827–1838, Dec. 2013.

[12] T. Oksanen and A. Visala, "Coverage path planning algorithms for agricultural field machines," *J. Field Robotics*, vol. 26, no. 8, pp. 651–668, Aug. 2009.

[13] H. Choset, "Coverage for robotics–a survey of recent results," *Ann. Math. Artificial Intell.*, vol. 31, no. 1, pp. 113–126, 2001.

[14] E. Galceran and M. Carreras, "A survey on coverage path planning for robotics," *Robot. Auton. Syst.*, vol. 61, no. 12, pp. 1258–1276, 2013.

[15] J. Latombe, *Robot Motion Planning*. Boston, MA: Kluwer Academic Publishers, 1991.

[16] H. Choset and P. Pignon, "Coverage path planning: The boustrophedon cellular decomposition," in *Field and Service Robotics*. London, United Kingdom: Springer, 1998, pp. 203–209.

[17] W. Huang, "Optimal line-sweep-based decompositions for coverage algorithms," in *Proc. IEEE Int. Conf. Robotics and Automation*, vol. 1, Seoul, Korea, May 2001, pp. 27–32.

[18] C. Fang and S. Anstee, "Coverage path planning for harbour seabed surveys using an autonomous underwater vehicle," in *Proc. IEEE OCEANS*, Sydney, Australia, May 2010, pp. 1–8.

[19] M. de Berg, O. Cheong, M. van Kreveld, and M. Overmars, "Voronoi diagrams: The post office problem," in *Computational Geometry: Algorithms and Applications*, 3rd ed. Berlin, Germany: Springer-Verlag, 2008, ch. 7, pp. 147–172.

[20] J. Jin and L. Tang, "Optimal coverage path planning for arable farming on 2D surfaces," *Trans. Amer. Soc. Agricultural and Biosystem Eng.*, vol. 53, no. 1, pp. 283–295, 2010.

[21] Y. Li, H. Chen, M. Joo Er, and X. Wang, "Coverage path planning for UAVs based on enhanced exact cellular decomposition method," *Mechatronics*, vol. 21, no. 5, pp. 876–885, Aug. 2011.

[22] M. Houle and G. Toussaint, "Computing the width of a set," *IEEE Trans. Pattern Anal. Mach. Intell*, vol. 10, no. 5, pp. 761–765, Sept. 1988.