Optimal Coverage Path Planning based on the Rotating Calipers Algorithm

J. Irving Vasquez-Gomez
CONACYT-Instituto Politécnico Nacional
México City, México
Email: jivasquezg@conacyt.mx

Magdalena Marciano-Melchor
Instituto Politécnico Nacional
CIDETEC, México City, México
Email: mmarciano@ipn.mx

Juan Carlos Herrera-Lozada
Instituto Politécnico Nacional
CIDETEC, México City, México
Email: jlozada@ipn.mx

***Abstract*—Field surveying by drones has become a successful activity with applications to mapping, precision agriculture, surveillance, etc. The current state of the art path planners for surveying rely on computing the path that minimizes the number of flight lines. However, such approach could be sub-optimal when the takeoff and landing points are included in the path. We propose a method that computes the optimal path, including the takeoff and landing points. Our method is based on the rotating calipers algorithm and it has a $O(n)$ complexity.**

I. Introduction

Field surveying by unmanned aerial vehicles (UAV or drones) has become a successful activity with applications to mapping, precision agriculture, surveillance, etc. A relevant problem is to plan a path for the vehicle so that when the path is followed the entire terrain (target area) is scanned. Such problem is known as coverage path planning [4]. The current state of the art coverage path planners rely on computing the path that minimizes the number of flight lines [5], [7], [8]. However, such approach could not be optimal if we consider the path to reach and exit from the target area. The issue is more evident in cases where the drone has to continue the surveying mission to another site. For example, the drone is looking for a person in a given area, however, once that the area has been covered it has to move to a point different from the take off point. In such cases, it is desirable that the path ends near to the next target. See as an example the path shown in Fig. 1.

We propose an efficient method, with complexity $O(n)$, to compute the optimal coverage path for a given target area but including the paths to reach and exit from the target area. The optimality is defined in terms of the time that the UAV needs to follow the path. The method is inspired on the computation of the antipodal points proposed by Shamos [6] which resembles a caliper that measures the width of a polygon. The proposed algorithm is a key piece in more complex tasks such as surveying of disjoint areas or building facade inspection, where each face of the building is a polygon that needs to be inspected.

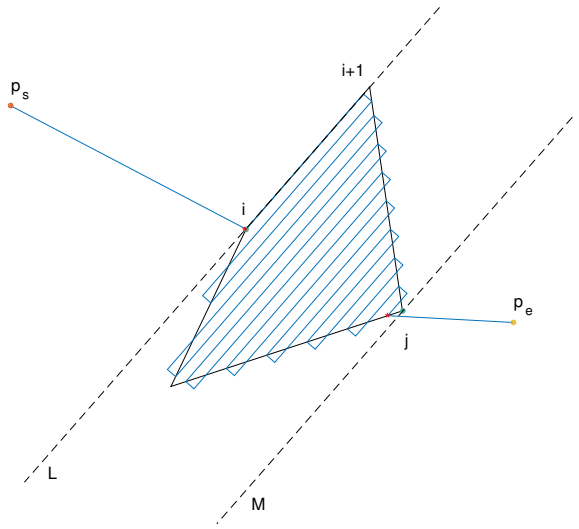


Figure 1. Example of an optimal coverage path. The path (drawn in blue) starts at the takeoff point p_s, follows a back and forth pattern inside the target area (polygon drawn in black) and moves to the landing point p_e. See the text for more details. Figure best seen in color.

II. Related Work

Coverage path planning has been extensively studied in the field of mobile robots [4]. The method of Boustrophedon cellular decomposition, proposed by Choset and Pignon [2], addresses the problem of covering polynomial wolds. That method performs a line sweep decomposition on the world; such decomposition divides the world into cells. Each cell is explored in a back and forth swept, however, the movement is always done in the same direction. Huang's method for coverage planning [5] was inspired on a robot that has to perform demining operations, so that, it is of vital importance that the robot guarantees that the area of interest will be totally covered. Huang proposes that the sweep direction of the back and forth movement of the robot must be done perpendicular to the minimum altitude (width) of the polygon. However, the method does not address the problem of traveling from one region to another.

Coverage path planning with drones has adopted several ideas from mobile robotics but it has included new features. Li et al. [7] propose a method to cover a convex or non-convex polygon. The method divides the non-convex regions into convex cells, then computes the optimal orientation of the flight lines by orientating the flight lines perpendicular to the minimum span (width) of the cell and finally optimizes the total path by selecting the best linking points (called joint points in Li's work). Unlike Li et al. approach where the flight lines are kept and only two linking points could be swapped, our method can change the orientation of the lines and therefore the linking points could change around the polygon. Torres et al. [8] propose a method based on the same coverage optimality for a polygon defined by Huang [5] but optimizes the path between target regions. A drawback of their method is that the calculation of the optimal orientation has a quadratic complexity, given that for each edge of the polygon, it has to compute the distance to each vertex. Balampanis et al. [1] propose a method for coverage with multiple UAVs which is based on the region growing algorithm.

Recent works have addressed the problem of minimizing the energy of coverage. Di Franco et al. [3] compute the optimal velocity for the flight but the method does not take into account the take off and landing points.

III. Path Planning

The problem addressed in this paper is to find the path that covers the target area in the minimum time taking into account the takeoff and landing points. Through this section we describe the proposed method. First, in subsection III-A we establish the assumptions and definitions of the problem. Then, in sections III-A-III-D we describe the method. Finally, in section III-E, we present an analysis of the method's computational complexity.

A. Assumptions and definitions

The area to be covered will be referred as the target area. We assume that the target area is planar and it is defined by a standard form convex polygon [6], $Q = \{V, E\}$, where $V = \{1, \ldots, n\}$ is a set of vertices ordered in clockwise direction and $E = \{(1,2), \ldots, (n-1, n), (n, 1)\}$ is the set of edges. The UAV is multirotor and we assume that during the surveying it follows two motion primitives, translation and rotation in site. The UAV takes off at point p_s and lands at point p_e, both points can be inside or outside the polygon. When the drone surveys an area, it begins at the starting point, moves to the target area, follows a search pattern and moves to the landing point. See Fig. 1.

A search pattern is a path that the drone follows to cover the target area. We use a back and forth pattern (BFP) where the drone traces straight lines inside the target area. The lines of the BFP, called flight lines, are connected by straight segments. The distance between flight lines depends on the camera and desired spatial resolution [3]. The advantage of the back and forth pattern is that it can be resumed into a series of waypoints, $\rho = \{p_0, \ldots, p_m\}$. For each waypoint of the path, the drone orientates itself to it and moves in straight line to reach it. If we include the takeoff and landing points, the path is written as $\tau = \{p_s, p_0 \ldots p_m, p_e\}$. The linking points are the waypoints where the drone starts or ends the BFP; they connect the path inside the target area to the takeoff or landing points. See red asterisks in Fig. 1.

Algorithm 1: Optimal coverage path planning.

Data: V, p_s, p_e
Result: τ

```
A ← computeAntipodalPairs(V);
c* ← ∞;
foreach (i, j) ∈ A do
    ρ ← BestPath(V,i,j);
    if cost({p_s, ρ, p_e}) < c* then
        τ ← {p_s, ρ, p_e};
        c* ← cost(τ);
```

B. Method overview

When a back and forth pattern is used to cover a target area each flight line is parallel to next one, so the initial and last flight lines are parallel. Let us replace the flight lines of the BFP by imaginary parallel lines of support L and M. See Fig. 1 as an example. Accordingly to [6], a line L is a line of support if it meets the boundary of a polygon and lies entirely in one side of it.

By definition, a support line can pass only trough a vertex or match with an edge of the polygon. In the case where one vertex is touch, such vertex will be considered as a linking point, given that it is in one of the extremes of the BFP. On the other hand, when the line matches an edge, the two vertices that form the edge could be linking points. Therefore, for any back and forth pattern the linking points can only match with the vertices that admit parallel lines of support. In Shamos thesis [6], such points are called antipodal pairs. For example, in Fig. 1 the vertices i and j are antipodal pairs and the BFP starts at i towards $i+1$ and ends in the antipodal of i, j. Note that for the same lines of support L and M, $i+1$ and j are also antipodal pairs, however, $i+1$ is not a linking point since it does not connect with the starting point.

Given that the linking points can only match with the antipodal pairs of a polygon, the idea of the proposed method is to compute all the antipodal pairs, find the best path for each antipodal pair and select the path that has the lowest cost in combination with the takeoff and landing points. See algorithm 1 which resumes the method.

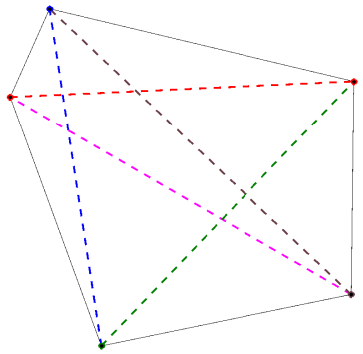


Figure 2. Example of antipodal pairs computed with Shamos Algorithm [6]. Dotted lines connect couples.

C. Antipodal pairs computation

The procedure `computeAntipodalPairs` computes all the antipodal pairs for a convex polygon. We use the algorithm proposed by Shamos [6]. It computes all the pairs in a $O(n)$ time, where n is the number of vertices of the polygon. It receives the set of vertices V and returns the pairs in the form $A = \{(i_1, j_1), \ldots, (i_n, j_n)\}$. This algorithm is known as the "rotating calipers algorithm" because it resembles to a dynamically adjustable caliper that is measuring the polygon as it rotates. Fig. 2 shows a set of antipodal pairs for the given polygon. Dotted lines relates each vertex with its antipodal pair.

D. Best path for each antipodal pair

In this procedure, we compute the optimal path given an antipodal pair. See algorithm 2. Lets us assume that i and j are an antipodal pair. The problem is to find the best BFP between i and j. Based on the principle that the best BFP is the one with the minimum number of flight lines. The problem is converted into finding a pair of parallel support lines that pass trough vertices i and j which have the minimum distance between them.

Li et al. [7] proved that the minimum width of a polygon can only occur between the distance of a vertex and an edge of the polygon (Li considers the edge-edge distance a special case of the vertex-edge case). So, the idea is: given an antipodal pair, rotate the caliper in clockwise direction until and edge is touch, then rotate the caliper in counter clockwise direction until a second edge is touch, finally measure both options in order to find the minimum width of the polygon.

In the following sections, we assume that all vertex indices are reduced modulo n (so that an index $n + 1 = 1$) and `angle`(m, n) is a function that computes the clock wise angle swept out by a line as it rotates from a position parallel to the edge $(m, m+1)$ to a position parallel to $(n, n+1)$[6].

Algorithm 2: Best path. The algorithm computes the best path for an antipodal pair. It receives a set of vertices, V, and the antipodal pair, (i, j).

Data: $V = \{1, \ldots, n\}$,(i, j)
Result: ρ

```
1  if angle(i,j) < angle(j,i) then
2      b ← j;
3      a ← i;
4  else
5      b ← i;
6      a ← j;
7  φ ← angle(b,a) − π;
8  γ_b ← angle(b-1,b);
9  γ_a ← angle(a-1,a) − φ;
10 if γ_b < γ_a then
11     b_2 ← b − 1;
12     a_2 ← a;
13 else
14     b_2 ← a − 1;
15     a_2 ← b;
16 if dist(b,a) < dist(b_2,a_2) then
17     ρ ← getPath(b, b + 1)
18 else
19     ρ ← getPath(b_2 + 1, b_2)
```

1) Rotate the caliper in clockwise direction until an edge, $(b, b+1)$, is found: For example, see Fig. 3 and note that if we rotate lines L and M over the antipodal points i and j the first edge of contact will be $(j, j+1)$, such edge will be called $(b, b+1)$ and it will be the base of one possible BFP.

One way to find $(b, b+1)$ is by finding the minimum angle between α_i and α_j, which are the angles from the support lines to the nearest edge of the polygon in clockwise direction. However, instead of directly measuring α_i and α_j, we measure the difference between the rotations from the line L' to M' and from M' to L'. See lines 1-6 of the algorithm 2. Once that the edge $(b, b+1)$ has been found, it is called a base edge because it resembles to the base of a triangle where the height is given by the antipodal pair of b, called a.

2) Find a second base edge by rotating the caliper in counter-clockwise direction: In this step, we find the other possible path, with base edge (b_2, b_2+1), by rotating the caliper in counter clockwise direction. However, instead of repeating the process the other way round, we measure the angles with respect of the line B, a line that matches $(b, b+1)$. See Fig. 4. The measured angles are γ_b, the angle with respect to B, and γ_a, the angle with respect to A, a line of support parallel to B that pass thought the antipodal pair of b. The angle ϕ complements γ_a to reach the edge $(a, a+1)$.

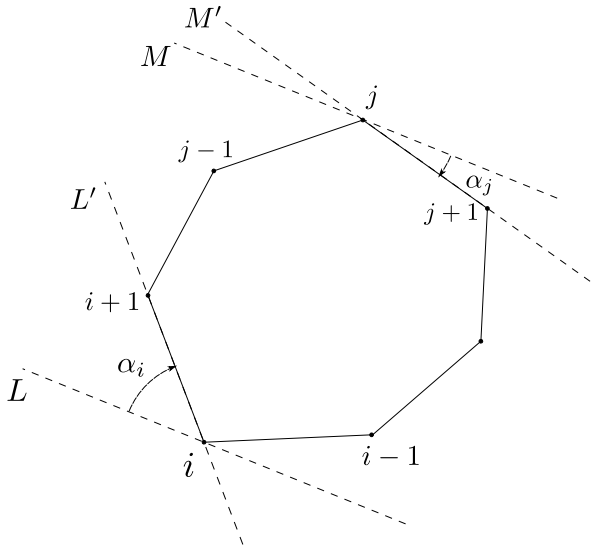


Figure 3. Illustration of the caliper rotation in clockwise direction. The lines L and M which represent the caliper are rotated until they touch the nearest edge.

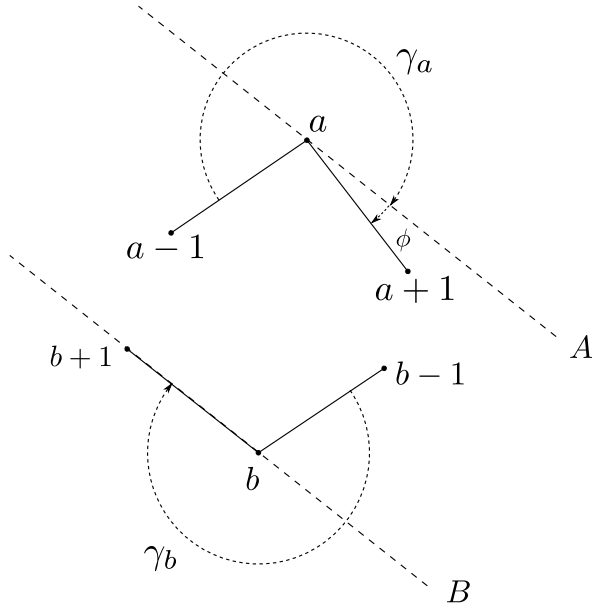


Figure 4. Illustration of the angles with respect of the found base edge. The minimum angle between γ_a and γ_a defines the nearest edge in counter-clockwise direction.

Notice that, from the previous step, $(b, b+1)$ was touch before $(a, a+1)$, therefore the angle ϕ is bigger than or equal to zero.

The minor of the angles γ_b and γ_a will tell us the nearest edge from the two possibilities: $(b-1, b)$ or $(a-1, a)$. Once that the second base edge, (b_2, b_2+1), has been found, the farthest vertex is the antipodal pair. See lines 7-15 from the algorithm 2.

3) Find the path that holds least flight lines: From the previews steps, we have two possible paths: i) path with base $(b, b+1)$ that ends at vertex a and ii) a path with base (b_2, b_2+1) that ends at vertex a_2. The path that holds least flight lines is the one with the smallest distance between the base and the farthest point. See lines 16-19 of the algorithm 2, where the function `dist`(b,a) computes the distance between the line that matches the edge $(b, b+1)$ and the vertex a, and the function `getPath`(b, c) computes a BFP with origin at vertex b with sweep direction perpendicular to (b, c).

E. Complexity

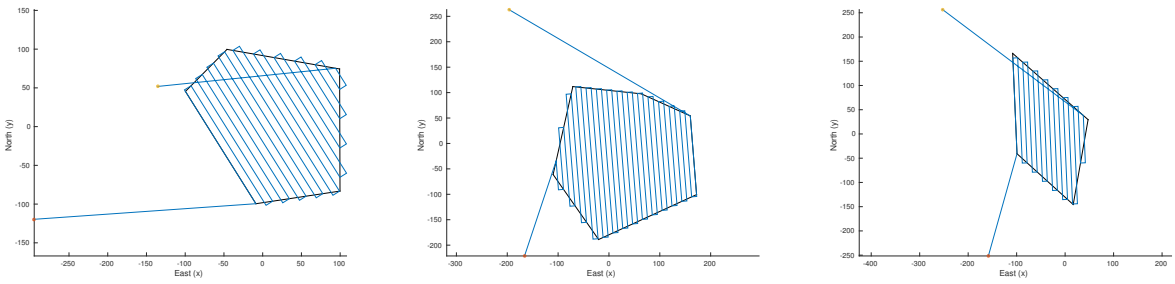
The computational complexity of the algorithm 1 is $O(n)$, where n is the number of vertices of the polygon. The algorithm requires a call to the antipodal pairs computation and a loop for each antipodal pair. The first step has been demonstrated to be $O(n)$ [6]. The second part calls the best path computation (algorithm 2) for each antipodal pair. Such algorithm performs a sequence of decisions and procedures that do not depend on the number of vertices, therefore its complexity is considered constant time, c. The important parameter to observe is how much antipodal pairs will be processed. In a more deep look of Shamos algorithm, it can produce at most 3 antipodal pairs per iteration, so, there is an upper bound of $3n$ pairs. Therefore, algorithm 1 performs in the worst case $3n \times c$ operations, simplifying, the complexity is $O(n)$.

IV. Experiments

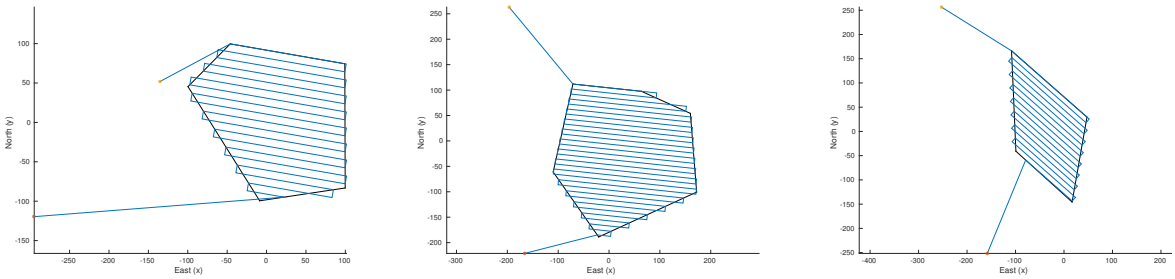
In this section, we compare the proposed method versus the state of the art approach of flight lines perpendicular to the minimum width (FLPMW). FLPMW approach was proposed by [5] for being implemented in mobile robots. Latter, FLPMW was adopted to UAV coverage in several works. In this experiment, we implement the algorithm described by Torres et al. [8] whose complexity is $O(n^2)$, given that it has to compute the distance from each edge to each vertex. In this experiment, we randomly generate convex polygons to survey and also we generate random starting and ending points. UAV forward speed was 10 m/s and heading rate was $\pi/4$ rad/s. Path cost is computed as the time that the UAV needs to complete the path.

Since, the solution provided by FLPMW is in the set of paths that are computed by our method (all antipodal pairs are analyzed, and the optimal path for each antipodal pair is obtained). The proposed method solution can be in the worst case equal to the solution provided by FLPMW, But, in many cases the proposed method provides a better solution. Fig. 5 shows some examples were the proposed method overcomes the FLPMW method. In our experiments, the time reduction was moderate, however it becomes a decisive factor in many tasks, such as search and rescue of injured people.

During the experiments, we have noticed that our method gets better results for compact polygons (where the relation



(a) FLPMW; $d(\tau) = 462$s (b) FLPMW; $d(\tau) = 842$s (c) FLPMW; $d(\tau) = 432$s



(d) Vasquez et al.; $d(\tau) = 450$s (e) Vasquez et al.; $d(\tau) = 821$s (f) Vasquez et al.; $d(\tau) = 415$s

Figure 5. Comparison of some of the solutions provided by the state of the art approach, flight lines perpendicular to the minimum width (FLPMW), versus the proposed method.

of the perimeter divided by the area is close to 4π). On the other hand, non compact polygons (for example elongated rectangles) are less prone to change the flight lines direction, since the distance to the starting and ending points is small with respect of the increment in the number of flight lines. A more depth analysis on the possible time reduction given the polygon shape is left for future work.

V. Conclusions and Future Work

We have presented a method for computing the optimal path that covers a convex target area. The proposed method is based on the rotating calipers algorithm and its complexity is $O(n)$. The proposed method, in the worst case, provides the same solution than state of the art approach where the flight lines are perpendicular to the minimum width of the polygon, but in the best case, the computed path is better. In a future work, we will apply the method to the coverage of disjoint areas and to the inspection of large 3D structures with unmanned aerial vehicles.

References

[1] Fotios Balampanis, Ivn Maza, and Anbal Ollero. Coastal areas division and coverage with multiple uavs for remote sensing. *Sensors*, 17(4), 2017.

[2] Howie Choset and Philippe Pignon. Coverage path planning: The boustrophedon cellular decomposition. In *Field and service robotics*, pages 203–209. Springer, 1998.

[3] Carmelo Di Franco and Giorgio Buttazzo. Coverage path planning for uavs photogrammetry with energy and resolution constraints. *Journal of Intelligent & Robotic Systems*, pages 1–18, 2016.

[4] Enric Galceran and Marc Carreras. A survey on coverage path planning for robotics. *Robotics and Autonomous Systems*, 61(12):1258–1276, 2013.

[5] Wesley H Huang. Optimal line-sweep-based decompositions for coverage algorithms. In *IEEE International Conference on Robotics and Automation (ICRA), 2001*, volume 1, pages 27–32. IEEE, 2001.

[6] Michael lail Shamos. *Computational Geometry*. PhD thesis, Yale University, 1978.

[7] Yan Li, Hai Chen, Meng Joo Er, and Xinmin Wang. Coverage path planning for uavs based on enhanced exact cellular decomposition method. *Mechatronics*, 21(5):876–885, 2011.

[8] Marina Torres, David A Pelta, José L Verdegay, and Juan C Torres. Coverage path planning with unmanned aerial vehicles for 3d terrain reconstruction. *Expert Systems with Applications*, 55:441–451, 2016.