Coverage path planning with unmanned aerial vehicles for 3D terrain reconstruction

Article *in* Expert Systems with Applications · February 2016

DOI: 10.1016/j.eswa.2016.02.007

CITATIONS
46

READS
1,115

4 authors:

Marina Torres Anaya
University of Granada
9 PUBLICATIONS **49** CITATIONS
SEE PROFILE

David A. Pelta
University of Granada
214 PUBLICATIONS **1,815** CITATIONS
SEE PROFILE

Jose Luis Verdegay
University of Granada
303 PUBLICATIONS **9,461** CITATIONS
SEE PROFILE

Juan Carlos Torres
University of Granada
135 PUBLICATIONS **612** CITATIONS
SEE PROFILE

Expert Systems With Applications 55 (2016) 441–451

Contents lists available at ScienceDirect

Expert Systems With Applications

journal homepage: www.elsevier.com/locate/eswa

Coverage path planning with unmanned aerial vehicles for 3D terrain reconstruction

Marina Torres[a], David A. Pelta[a,*], José L. Verdegay[a], Juan C. Torres[b]

[a] Department of Computer Science and Artificial Intelligence, Models of Decision and Optimization Research Group, Universidad de Granada, Granada 18014, Spain

[b] Department of Systems and Languages, Virtual Reality Lab, Universidad de Granada, Granada 18014, Spain

ARTICLE INFO

Keywords:
Coverage path planning
Unmanned aerial vehicle
Heuristics
Non-convex areas
3D terrain reconstruction

ABSTRACT

Three-dimensional terrain reconstruction from 2D aerial images is a problem of utmost importance due its wide level of applications. It is relevant in the context of intelligent systems for disaster managements (for example to analyze a flooded area), soil analysis, earthquake crisis, civil engineering, urban planning, surveillance and defense research.

It is a two level problem, being the former the acquisition of the aerial images and the later, the 3D reconstruction. We focus here in the first problem, known as coverage path planning, and we consider the case where the camera is mounted on an unmanned aerial vehicle (UAV).

In contrast with the case when ground vehicles are used, coverage path planning for a UAV is a lesser studied problem. As the areas to cover become complex, there is a clear need for algorithms that will provide good enough solutions in affordable times, while taking into account certain specificities of the problem at hand. Our algorithm can deal with both convex and non-convex areas and their main aim is to obtain a path that reduces the battery consumption, through minimizing the number of turns.

We comment on line sweep calculation and propose improvements for the path generation and the polygon decomposition problems such as coverage alternatives and the interrupted path concept. Illustrative examples show the potential of our algorithm in two senses: ability to perform the coverage when complex regions are considered, and achievement of better solution than a published result (in terms of the number of turns used).

1. Introduction

Unmanned aerial vehicles (UAVs) have multiple uses at present besides obvious military applications. Search or exploration (Sujit, Sousa, & Pereira, 2009), impact analysis after an earthquake (Xu et al., 2014) and others like forest health monitoring, mine surveys or air quality monitoring (Watts, Ambrosia, & Hinkley, 2012) are some examples of civil applications of UAVs. Another interesting example is the 3D terrain reconstruction from 2D images taken from the UAV: the vehicle flies over an area taking overlapped images that will be used next to obtain a 3D reconstruction of the ground. The whole reconstruction problem is far from trivial and it can be decomposed into two different tasks: (1) the coverage path planning problem (CPP), and (2) 3D reconstruction from 2D images, which includes others like image alignment or features detection.

In this contribution we focus on CPP whose aim is to find a path for a vehicle in order to completely visit an area. The problem can be formalized when considering unnamed ground vehicles (UGVs; Choset, 2001; Choset & Pignon, 1998; Lee, Baek, Choi, & Oh, 2011), underwater robots (Bagnitckii, Inzartsev, & Senin, 2011; Hert, Tiwari, & Lumelsky, 1996) or UAVs but for the case of the 3D reconstruction of the terrain, a UAV is needed. CPP with UGV is a well studied problem in the context of robotics, see for example Galceran and Carreras (2013), where some considerations about its computational complexity are also stated. In this sense, several similar problems are considered as NP-hard.

UAVs are commonly used for the task allocation problem (Besada-Portas, De La Torre, Moreno, & Risco-Martín, 2013; Choset & Pignon, 1998) but, as long as we know, CPP using UAVs is a lesser studied problem sometimes simplified to deal only with

* Corresponding author. Tel.: +34958244216.
E-mail addresses: torresm@correo.ugr.es (M. Torres), dpelta@ugr.es, dpelta@decsai.ugr.es (D.A. Pelta), verdegay@ugr.es (J.L. Verdegay), jctorres@ugr.es (J.C. Torres).

http://dx.doi.org/10.1016/j.eswa.2016.02.007
0957-4174/

convex polygons as in Maza and Ollero (2007). When dealing with concave polygons, other researches propose methods focused on the path's construction regardless of the area's shape Valente, Cerro, Barrientos, and Sanz (2013) or Franco and Buttazzo (2015) where the UAV's path is made as if it was a convex polygon and turning back to cover a remaining sub-area caused by a concave zone. A more complex method presented in Huang (2001) is based on the subdivision of the area to simplify the surface into different areas covered by a determined motion. This simplified approach is similar to one proposed in Ji, Wang, Niu, and Shen (2015) where a concave polygon is decomposed into convex polygons.

In Li, Chen, Er, and Wang (2011), the authors propose a covering strategy where the beginning of the path is defined by the user but the endpoint is given by the algorithm (the user can not define it). As the UAV lands at the end of the path, it could happen that this point is unreachable for the user and the UAV could not be recovered.

In this context, the aims of this contribution are: firstly to propose and secondly, to evaluate a set of strategies to solve the CPP problem using a UAV, when considering both convex and non-convex areas.

From a practical point of view, and in contrast to Li et al. (2011), the take off and landing points for the UAV are chosen by the user. In real scenarios, the experts starts and finishes the UAV flight or mission at the same point, so from now on, we assume that the take off and landing point of the UAV are the same.

Also, the proposed strategies can be used with any rotorcraft UAV able to perform waypoint navigation and rotate around its own axis.

The paper is structured as follows. Section 2 presents an overview of the CPP problem for 3D terrain reconstruction, stating the inputs and output that an algorithm will need to solve the problem. Section 3 provides additional background information. Then, in Section 4 we propose a method to solve the coverage problem when the area is represented as a convex polygon. This is then expanded in Section 5 to address the cases of concave polygon or multiple connected convex polygons. We comment how to transform the problem of a concave polygon coverage into a multiple convex polygon coverage, stating when this transformation is really needed. Finally, in Section 6 we outline some considerations based on practical experiences, that allows us to speed up the solution of the problem.

Illustrative examples and results are shown in Section 7, while conclusions and further discussion are presented in Section 8.

2. Coverage path planning problem for 3D terrain reconstruction

As we stated before, we wish to design a path for the UAV that allows to obtain images fully covering the area of interest. As these images will be later used for terrain reconstruction, several considerations arise:

- *Overlapping:* Consecutive pictures should have a given percentage of overlapping. The greater the overlap is, the higher the accuracy of the 3D model will be.
- *Time contiguity:* The quality of the 3D texture will be higher when the pictures of contiguous areas of the terrain are taken at similar time. Otherwise, uncorrelated shadows or visual differences may appear, leading to a more difficult reconstruction and a less quality texture.
- *Orientation:* It is desired to have the pictures taken in the same orientation because it leads to a simplification in the 3D reconstruction phase (correlation among them are easier to find).

The coverage problem considered in this contribution needs as input:

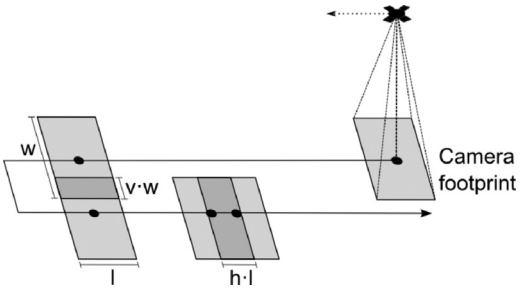

Fig. 1. The camera footprint on the terrain has length l and width w. Parameters h and v denote the horizontal and vertical overlapping percentage among images.

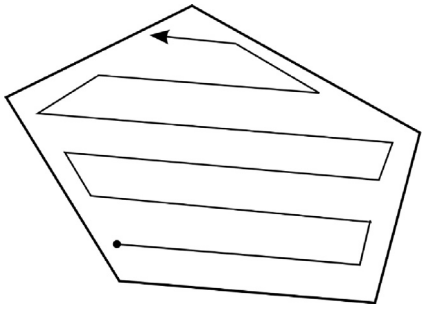

Fig. 2. Example of a CPP' solution. The path fully covers the polygon.

- The polygonal region.
- The start-end point: The start (take off) and the end (landing) points which are considered the same.
- The camera footprint: The length, l, and width, w, on the terrain taken by one image as shown in Fig. 1. These footprints are determined by the flight's height and the camera's features.
- The horizontal, h, and vertical, v, overlapping percentage among images, as depicted in Fig. 1.

A solution to the problem is a path for the UAV that allows a complete coverage of the region. An example is shown in Fig. 2.

3. Background information

The most critical point when using a UAV in this kind of problem is to minimize power consumption. As stated in Li et al. (2011) the fuel consumption can be reduced by decreasing the number of turns. For a fixed distance, the time is increased when the rotorcraft turns, because it has to completely stop before start moving into a different direction, wasting time while it slows down and accelerates once it has changed the direction. The path will be created with a zigzag motion (a.k.a a back and forth motion) trying to minimize the number of turns the rotorcraft must do along the coverage.

Rotorcrafts are able to move backwards in the same way as they can move forwards and sideways. Actually, they can perform movements on any direction. That simplifies the turns, because the rotorcraft can change the movement direction without changing the heading orientation.

Taking advantage of this feature and to reduce the time spent in every turn, the rotorcraft will not turn itself, instead the rotorcraft will be always heading to the same direction and will move sideways and backwards when it is required. As a side and necessary effect, the pictures' orientation will be the same for all pictures taken, facilitating the following reconstruction problem.

Having these ideas in mind, our algorithm will return a path that will be traversed as a zigzag or back and forth motion composed by longitudinal (the rows), transverse and possibly slightly diagonal moves.

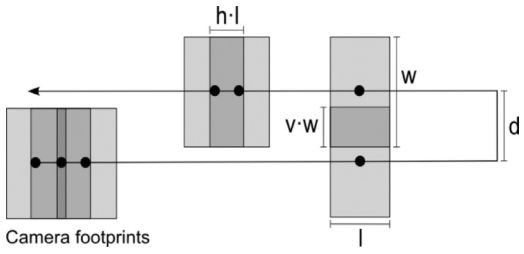

Fig. 3. The camera footprint on the terrain has length, l, and width, w. Parameters h and v, denote the horizontal and vertical overlapping percentage among images.

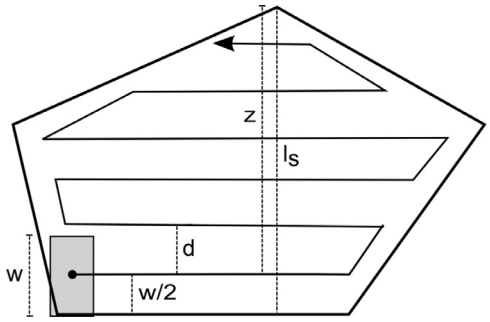

Fig. 4. The values needed to calculate the number of turns are shown. l_s is the length of the so called optimal line sweep direction.

In order to generate such back and forth motion, the distance between two rows is needed and its calculation depends on the defined vertical overlap and the camera footprint. A graphical scheme is shown in Fig. 3.

Let v be the vertical overlap and let w be the width of the camera footprint. The distance among rows, denoted as d, is the (vertical) distance between both footprints. Taking into account the vertical overlapping, d is calculated as follows:

$$d = w \cdot (1 - v)$$

The number of turns, n, to perform in a given polygon depends on the value d, w and l_s, where l_s is the length of the so called optimal line sweep direction (which is explained in Section 4.1).

Let us define an intermediate value $z = l_s - w/2$, where the term $w/2$ represents half of the size of the camera footprint width. Graphically, these parameters are shown in Fig. 4.

Let d define the distance separating rows. $\lceil z/d \rceil$ denotes the number of rows needed to perform the polygon coverage using a zigzag motion. Nevertheless, if the distance between the last row and the upper vertex ($z \bmod d$) is larger than $w/2$, the polygon is not fully covered by the last row and another row is required.

For each segment, two turning points are needed,[1] thus leading to a total number of turns given by the expression:

$$n = \begin{cases} 2 \cdot \lceil z/d \rceil & if\, z \bmod d \leq w/2 \\ 2 \cdot (\lceil z/d \rceil + 1) & if\, z \bmod d > w/2 \end{cases} \quad (1)$$

In short, it is clear that the number of turns depends on z because d is fixed at the problem's formulation according to the resolution required for the images. At the same time and for the same reason, z depends on the length l_s. The conclusion then is that the number of turns depends on l_s and to minimize the turns, the length l_s should be minimized. In other words, we need to determine the optimal line sweep direction.

[1] In some exceptional situations, just one turning point is needed. However, we do not consider such situation. In the worst case, our calculations will return a higher number of turning points.

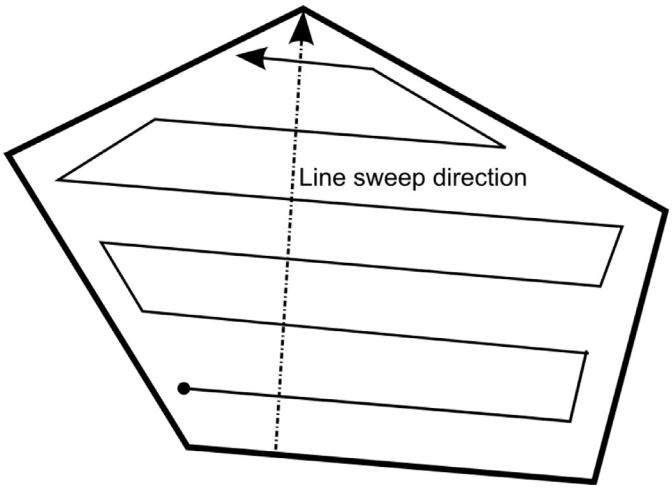

Fig. 5. Example of a line sweep direction.

4. A CPP Algorithm for single convex polygon coverage

In this section we deal with the problem for a convex polygon. The method includes a new concept named as "coverage alternative" that determines the direction in which the path is defined. This process is done after the number of turns calculation.

The first step of the method involves the calculation of the optimal line sweep direction, the one labeled with length l_s at Fig. 4 and then, the second step consists on constructing the coverage path using a back and forth motion as perpendicular rows to the optimal line sweep as shown in Fig. 5.

4.1. Optimal line sweep direction

As stated in Huang (2001), optimal line sweep direction's calculation is a well known and already solved problem. More than one line sweep may exist, but the optimal one allows to construct a path that minimizes the number of turns as commented in Section 2. A brief description of the calculations needed is included here for the sake of completeness.

When dealing with a convex polygon, the optimal line sweep direction is calculated as shown in Algorithm 1. Initially, a line from every edge of the polygon to the farthest vertex is "draw" and its length is measured. The line with the shortest distance is the optimal line sweep (the one that will generate less turns in

Algorithm 1 Line sweep direction.

```
distance(e, v): Euclidean distance between edge e and vertex v
for all edges in the polygon do
  max-dist-edge = 0
  for all vertex in the polygon do
    if distance(edge, vertex) > max-dist-edge then
      max-dist-edge = distance(edge, vertex)
      opposed-vertex = vertex
    end if
  end for
  if (max-dist-edge < optimal-dist) or (is first edge) then
    optimal-dist = max-dist-edge
    line-sweep = direction FROM edge TO opposed-vertex
  end if
end for
```

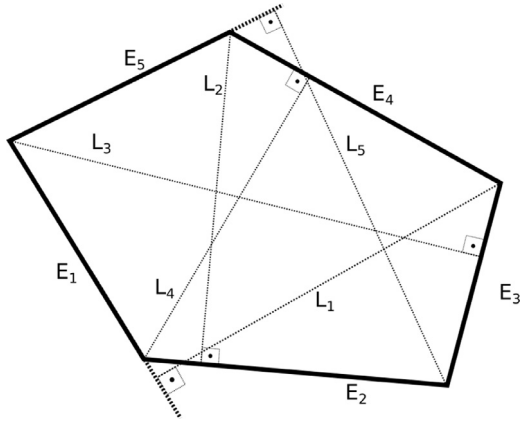

Fig. 6. Optimal line sweep direction's calculation for a convex area.

the coverage path). The line's direction points to the corresponding vertex.

Fig. 6 shows an example of the line sweep calculation method. For each edge E_i the line L_i to the farthest vertex is displayed. In the example, the shortest distance is the one associated to the edge E_4 so the optimal line sweep is the line L_4 pointing to the vertex connecting the edges E_1 and E_2 (to establish its direction). Fig. 7 shows an optimal (the one with minimal length) and a non-optimal line sweep direction.

4.2. Coverage alternatives

Once the optimal line sweep direction is obtained, we define four coverage alternatives to generate the back and forth motion. These are shown in Fig. 8.

A coverage alternative is defined according to two criteria. The first one is if the current direction of the line sweep or the opposite one is considered; and the second is the way the coverage path is constructed: clockwise (the first turn is made to the right) or counterclockwise (the first turn is made to the left). In short, considering:

```
coverage alternative = <CP><LSD>
CP = {clockwise (CW) | counterclockwise (CCW)}
LSD = {current | opposite}
```

where CP stands for coverage path and LSD for line sweep direction; the four alternatives are summarized in next table:

	CP	LSD
A_1	CCW	Current
A_2	CW	Current
A_3	CW	Opposite
A_4	CCW	Opposite

Now, given a specific pair P composed by a polygon p and a coverage form A_i, $P = (p, A_i)$, we denote the start and end points of the zigzag path, as a result of the alternative chosen, as $S(P), E(P)$

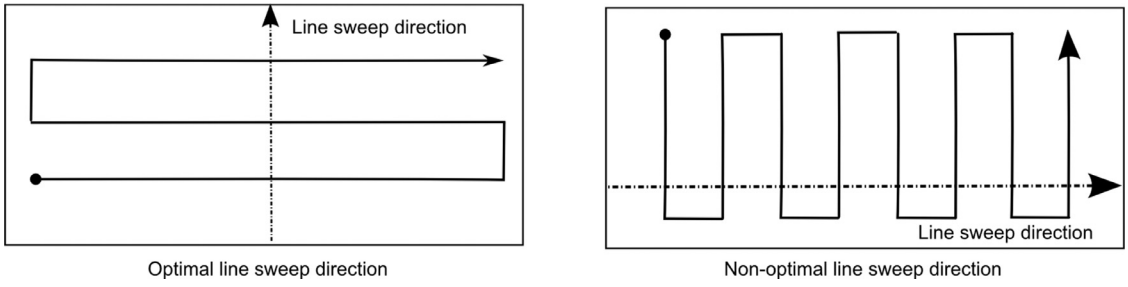

Fig. 7. Optimal and non-optimal line sweep direction.

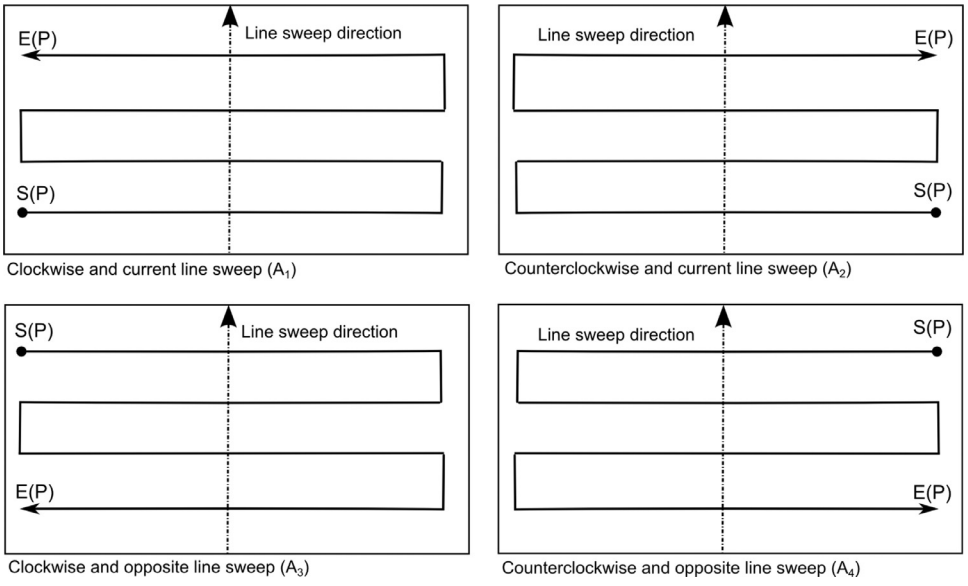

Fig. 8. Different alternatives to cover a region.

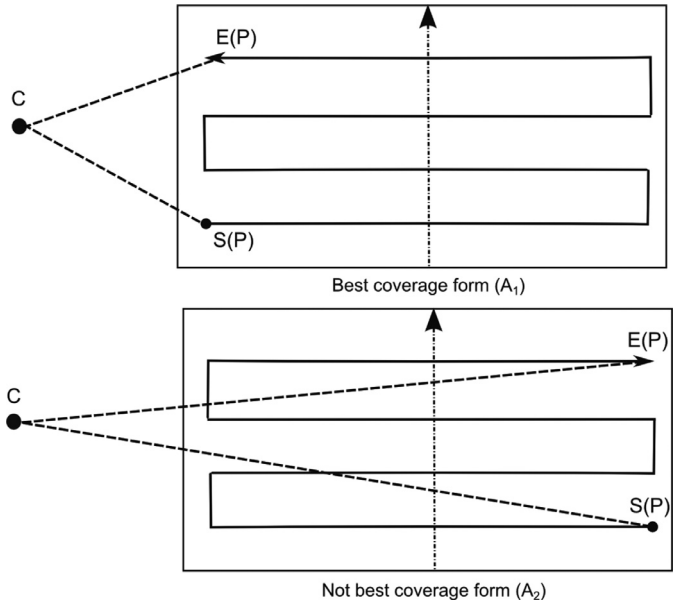

Fig. 9. The coverage alternative selected has impact when the connection to the ground control point is considered.

and the coverage distance for $P = (p, A_i)$ (the length of the path connecting $S(P)$ and $E(P)$) is denoted as $\mathrm{cd}(P)$.

All coverage alternatives contain the same and minimum number of turns because they are built based on the same line sweep.

Occasionally, they produce the same coverage distances as in Fig. 8. However, when considering polygons with other shapes, some differences between the coverage distances may appear. They will be caused by the variations found on transverse segments of the path generated by the zigzag motion when the coverage alternative is decided clockwise, or counterclockwise.

The problem we want to solve can be stated as finding the coverage alternative A_i of the pair $P = (p, A_i)$ that minimizes the total distance D:

$$D(P) = d(C, S(P)) + \mathrm{cd}(P) + d(E(P), C) \quad (2)$$

where $d(x, y)$ stands for the Euclidean distance between points x, y in the plane and C is the ground control point of the UAV (where the vehicle take off and land). C can be located anywhere in the terrain. The distances $d(C, S(P))$ and $d(E(P), C)$ are connections between the ground control point and the polygon. Such distances are called transition distances and they do not belong to the coverage path itself. These transitions are rectilinear movements so they do not affect the number of turns.

Fig. 9 shows the resulting paths for different coverage alternatives and the same ground control point C. The coverage alternative A_2 (clockwise) (bottom image) generates higher transition distances than the one at the top constructed by A_1, a counterclockwise alternative. As the point C is fixed, our aim is to find the A_i that minimizes $D(P)$. It should be noted how the starting and ending points of the coverage path changed their position as a function of the coverage alternative.

The strategy we propose to solve this problem is extremely simple, test every pair $P = (p, A_i), i = 1, \ldots, 4$, and keep the one that minimizes $D(P)$.

5. A CPP Algorithm for concave or multiple polygons coverage

The CPP can be also easily solved in certain concave areas. As this is not the usual case, we propose to decompose in a *set* of multiple polygons (being convex, concave or both). Before showing how we solve CPP in these cases, we need to comment first certain aspects about the optimal line sweep calculation in non-convex polygons that are needed to understand our proposal.

5.1. Optimal Line sweep direction calculation for concave polygons

When dealing with concave polygons, the procedure for calculating the optimal line sweep is somehow similar to the method explained in Section 4.1. In this case, we discard some edges following the next two steps procedure. First, calculate the convex hull of the set of vertices (the smallest convex set that contains all of them) and second, exclude those edges that are inside the convex hull. Then, the line sweep calculation is done as in the convex polygon case.

Fig. 10 shows an example of the procedure. On the top, the concave polygon appears. Then, in the middle, the convex hull is shown. The edges inside the convex hull, namely E_4, E_5 and E_6, are not employed. We just consider the edges E_1, E_2 and E_3 (bottom) and we conclude that the optimal line sweep is determined by L_2 (its direction points towards the vertex connecting edges E_3 and E_4).

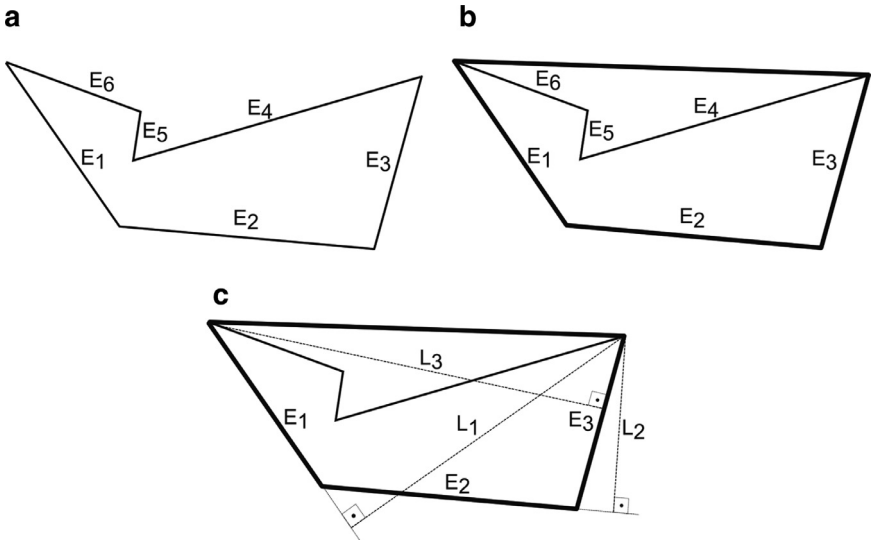

Fig. 10. Optimal line sweep direction calculation for a concave polygon. Edges inside the convex hull are excluded from the calculation.

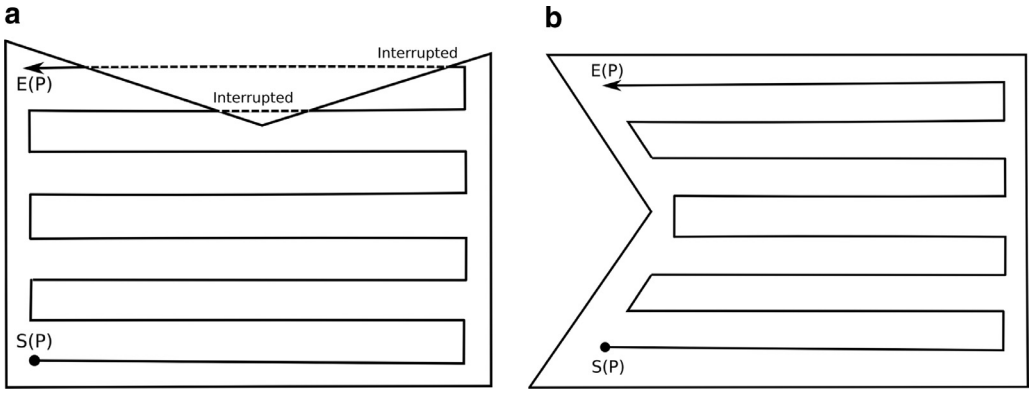

Fig. 11. Examples of coverage path interrupted (a) and not interrupted (b). In the latter case, the polygon is covered as a convex polygon.

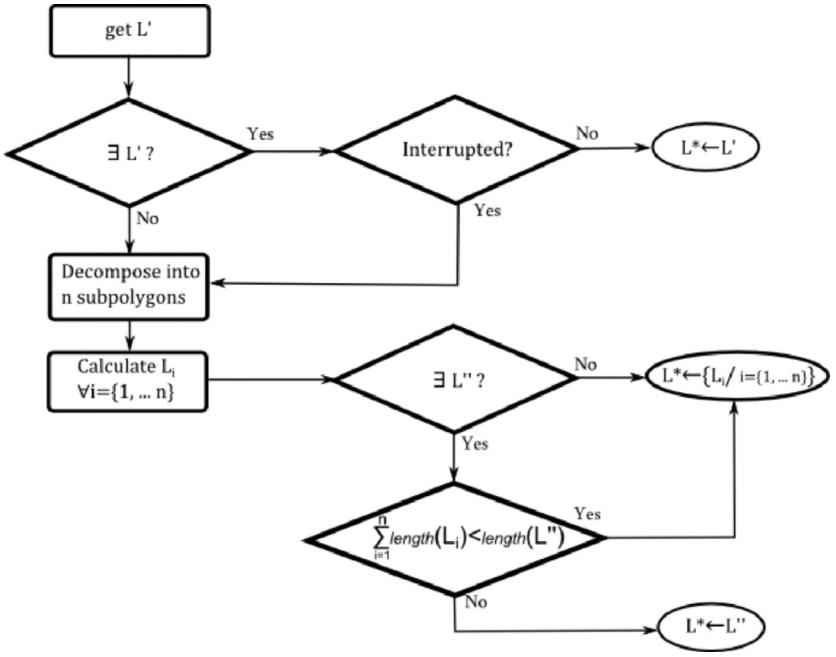

Fig. 12. Optimal line sweep calculation for concave polygons.

If the set is empty, the line sweep does not exist and the polygon should be decomposed in order to perform the coverage (as we will see in Section 5.2).

5.2. Concave polygon decomposition

If the polygon to cover is non-convex, we propose an algorithm that will allow us to perform the coverage as if we were dealing with multiple convex polygons.

The algorithm will require to detect first if the polygon can be covered as a convex polygon using the optimal line sweep calculated. A non-convex polygon can be covered using one line sweep direction (this is not true in general) if the path is not interrupted. We consider that the motion is interrupted when at least a row of the zigzag path crosses an edge of the polygon as Fig. 11 (a) illustrates.

If the UAV can carry out the back and forth motion on the whole polygon *without any interruption* caused by a concave vertex, then the polygonal area can be covered using such line sweep direction as reference. Fig. 11 (b) shows an example of this situation where, despite the concave vertex, the polygon can be covered as a convex polygon.

However, if the motion is interrupted the best of the following two alternatives is taken: (1) find a non-optimal line sweep and check if the coverage can be done and (2) decompose the polygon as a set of subpolygons.

This second task can be solved by any available algorithm in the literature. For example, the recursive greedy algorithm described in Levcopoulos and Krznaric (1998) or the heuristic one described in Levcopoulos and Lingas (1984). For a wider view of polygon decomposition algorithms, the reader is referred to Keil (2000). Also, in Fernández, Tóth, Cánovas, and Pelegrín (2008) the authors propose a way for decomposing a polygons with holes into convex polygons by diagonals. Our algorithm can also managed the output of such algorithm.

It is important to note that our method does not require the decomposition to be optimal or composed exclusively by convex polygons so a non-optimal but faster decomposition method will perfectly meets our needs.

The diagram shown in Fig. 12 describes our strategy. Given a polygon, the algorithm's output L^* could be a single line sweep or a set of line sweeps (if the decomposition of the polygon is a better strategy). Let L' be the optimal line sweep calculated as described in Section 5.1 and L'' the next best line sweep that does not interrupt the back and forth motion, if it exists. If L' generates

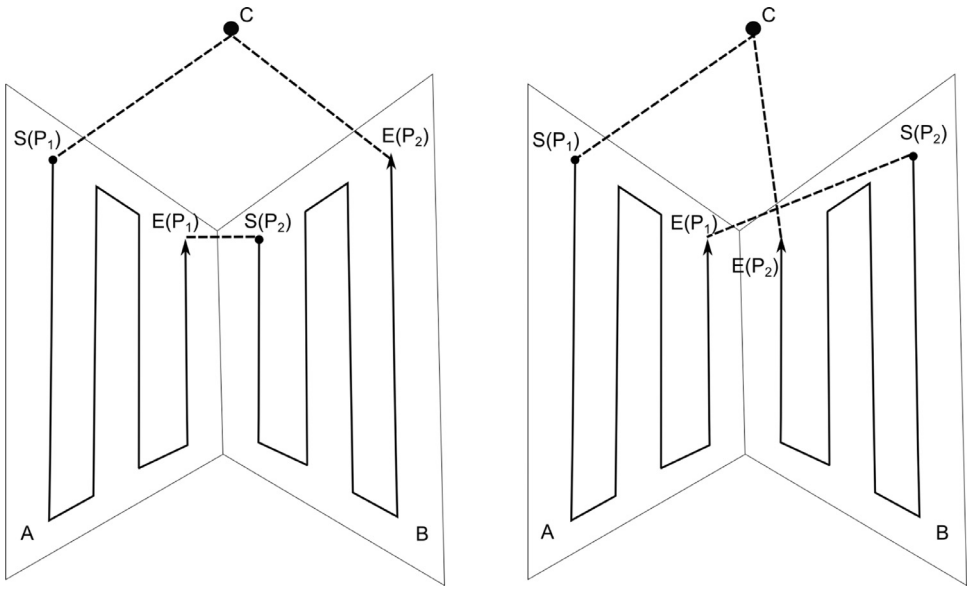

Fig. 13. Transition distances (connecting to point *C*) change as a function of the coverage alternative (note the different covering in polygon B).

an interrupted path, the polygon should be decomposed and the optimal line sweeps L_i for the n subpolygons are calculated using this algorithm recursively. Then the set of L_i is compared with the second best line sweep L'':

$$\left(\sum_{i=1}^{n}\text{length}(L_i)\right) < \text{length}(L'').$$

If the condition holds, the decomposition is the alternative chosen and the set of L_i is returned (every subpolygon will have its own line sweep direction). The same output is returned when L'' does not exist. Otherwise, the sub-optimal line sweep L'' is returned.

Using the output of the algorithm, the polygon (considered as a single one or as a set of subpolygons) will be covered as explained next.

5.3. Covering multiple polygons

In this section we analyze the situation where the UAV must cover more than one polygon. We assume the polygons have been previously decomposed and each polygon can be covered using one single line sweep. We need to determine the order in which the polygons are covered and, now, new transitions appear (e.g. a segment connecting two regions). Those transitions are measured from the end of a sub-region's path coverage to the starting point of the next sub-region to cover.

As every sub-region is a single polygon (only one line sweep is used), for each one we have the previously mentioned four alternatives $A_1, \ldots, A_4$ to decide the coverage direction. As we stated before, the way the convex polygon is covered will determine the starting and ending points of the coverage path for that polygon.

In case of multiple polygons, a solution is determined by the visiting order and the corresponding coverage alternatives of the polygons. We define $\varphi = (p_1, p_2, \ldots, p_n)$ as a permutation of n polygons that indicates the order in which they are visited and $\varphi' = (a_1, a_2, \ldots, a_n)$ where $a_j \in \{A_1, A_2, A_3, A_4\}$ indicates the coverage alternative in which the polygon φ_j is covered. Both permutations are referenced by the pair $\widetilde{P} = (\varphi, \varphi')$. Then, the coverage path for the polygon covered in position j is denoted by $P_j = (\varphi_j, \varphi'_j)$ and its coverage distance is $\text{cd}(P_j)$.

In case of multiple polygons we need to consider the distance of the transitions connecting the different polygons. The total transition distance, T, of a pair $\widetilde{P} = (\varphi, \varphi')$ given a ground control point C is defined as:

$$T(\widetilde{P}) = d(C, S(P_1)) + \cdots + d(E(P_j), S(P_{j+1})) + \cdots + d(E(P_n), C) \quad (3)$$

In other words, $T(\widetilde{P})$ is the sum of the length of the segments connecting the polygons plus the distance needed to reach the first polygon from ground point C and the distance from the last polygon back to C. The length of the transitions depends on the order in which the polygons are covered as well as on the alternatives selected to cover the polygons.

At the end, we need to determine the order in which the polygons are visited, φ, and the coverage alternative used for everyone, φ', with the objective of minimizing the total distance of the path, defined as:

$$\min f(\widetilde{P}) = \left\{ T(\widetilde{P}) + \sum_{j=1}^{n}\text{cd}(P_j) \right\} \quad (4)$$

where, as stated before, $\text{cd}(P_j)$ is the coverage distance for the polygon φ_j using the coverage alternative φ'_j.

Fig. 13 shows two examples for covering and connecting two polygons A and B. The solution on the left is $\widetilde{P}_1 = (\varphi = (A, B),\ \varphi' = (A_1, A_4))$ and the one on the right is $\widetilde{P}_2 = (\varphi = (A, B),\ \varphi' = (A_1, A_2))$. The polygon B is covered with a different coverage alternative on each solution. Those two alternatives exchange the start $S(P_2)$ and end point $E(P_2)$ of the polygon B resulting in different transitions between both polygons and the ground control point C. Although the coverage distance in B is the same under $\widetilde{P}_1$ and $\widetilde{P}_2$, we need to take into account the transition distances which are clearly shorter in the first case.

In order to minimize f, a basic algorithmic approach would be to generate all possible combinations of permutation of polygons and coverage alternatives, and then selecting the solution with minimum cost. For n polygons, there are $n!$ possible permutations without repetition. Only half of those represent a different path because the permutation algorithm must avoid generating the same path but on the opposite direction. Also, for each polygon there are four coverage alternatives, so we have $(n!/2) \cdot 4^n$ different

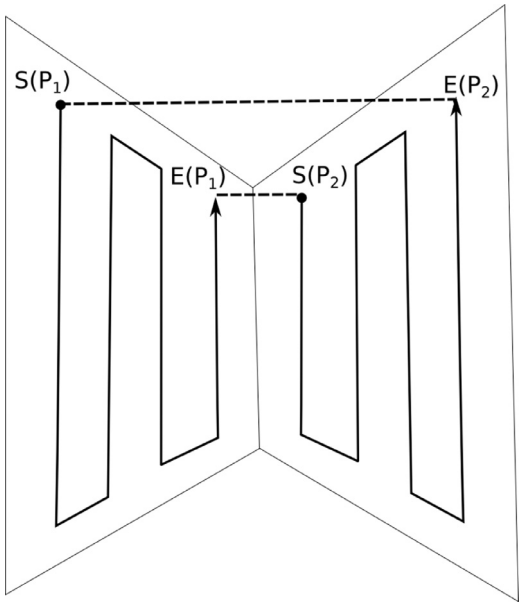

Fig. 14. Example of a closed path.

possible solutions. Clearly, this procedure is only affordable when n is "low".

In order to manage higher n values, we propose two criteria relying on two basics ideas that allow to reduce the number of permutations to explore: finding the best closed path and forcing some restrictions in the possible polygons orders.

6. Improvements from practical experience

When working in real world covering problems for 3D reconstruction, the experts "introduce" some peculiarities in the problem. Firstly, they manually define the polygon to be covered in a digital map. Because of that, in practice, the polygons tend to be relatively simple. Moreover, there is always a simpler polygon that contains the first one. Also, the method for the concave polygon decomposition described in Section 5.2 slightly reduces the number of subpolygons needed.

Secondly, it is common to plan the covering path "at the office", before knowing a possible location for the ground control point as occur with unreachable or unknown terrains.

Both behaviors allows us to improve the performance of our algorithm in two senses: we can consider that the number of subpolygons to cover is small; and we propose to solve the problem without the ground control point as input, which is left to be selected without restrictions at the end of the process.

6.1. Closed path

One way to reduce the number of permutations to explore, is to consider a slightly simpler problem associated with the so called closed path.

A closed path is a path that covers all polygons but does not consider the ground control point. The last covered polygon is connected with the first one using a straight line, thus adding a new transition between points $E(P_n)$ and $S(P_1)$ as Fig. 14 shows. This approach has perfect sense in real cases because the ground control point C is usually located close to the region to cover.

We define the best closed path as the one that generates the minimum total distance.

Once the best closed path is known, the ground control point C will be connected to the end $E(P_i)$ and the start $S(P_{i+1})$ of one of the transitions segments. The aim is to select the points $E(P_i)$ and $S(P_{i+1})$ that minimizes $d(E(P_i), C) + d(C, S(P_{i+1}))$.

The main idea of the algorithm is to find the minimum distance as in Section 5 but without considering the ground control point C. The transitions sum is then calculated as:

$$T(\widetilde{P}) = \left(\sum_{j=1}^{n-1} d(E(P_j), S(P_{j+1}))\right) + d(E(P_n), S(P_1))$$

We also fix the initial polygon φ_1 to avoid generating repeatedly the same path (any rotation of φ represents the same permutation) but not its coverage alternative. The other conditions are kept: all coverage alternatives for each polygon are considered and the opposite orders are not generated. The problem then is slightly simplified and now $((n-1)!/2) \cdot 4^n$ permutations exists. Although it does not look very impressive, when $n = 6$, we reduce from 1.474.560 to 245.760 the number of permutations to be considered.

6.2. Considering the spatial adjacency of the polygons

As the reader may observe, the differences in the total distance for different closed paths depend more heavily on the transition distances than on the coverage distances. Using this idea, we propose a strategy to avoid some permutations that will cause long transitions, for example, when considering as consecutive in the permutation two sub-regions that are not spatially adjacent. We consider two regions as adjacent if they have at least one vertex in common.

Thus the method can limit the number of transitions between non-adjacent sub-regions, forcing to connect a sub-region to an adjacent one, if possible. The idea is to process only those permutations that include at least a specific number of transitions between adjacent sub-regions. That implies that the set of permutations considered forcing at least i adjacencies includes all permutations for $i+1$ adjacencies. Let us denote as adj_i the set of permutations with at least i adjacencies, then

$$\text{adj}_{i+1} \subseteq \text{adj}_i \quad \forall i, 1 \leq i < n$$

and because of that, the best solution found when forcing i adjacencies is always better or equal than the one found when $i+1$ adjacencies are forced.

As the number of adjacencies forced increases, the number of permutations satisfying the adjacency criterion decreases. Given a feasible permutation, the corresponding path construction and evaluation is the most time consuming process of the method. These feasible permutations are the only ones that are generated and evaluated, thus the computational time to provide a solution is diminished.

7. Illustrative examples

In order to show our proposal, we will develop next two illustrative examples. In the first one we will show the path coverage over a complex non-convex polygon and in the second example, we will provide an alternative and more efficient path over the example problem presented in Li et al. (2011).

7.1. First example: Covering a complex area

Here we will show the output of our algorithm for the complex polygon shown in Fig. 15 (a). As the polygon is not convex, we need to apply first the algorithm described in Section 5.

Starting the procedure, the line sweep directions are found as explained in Section 5.1. The optimal line sweep direction for the whole polygon L' is the one calculated for the edge 9 but it generates an interrupted path. As there is no second best line sweep

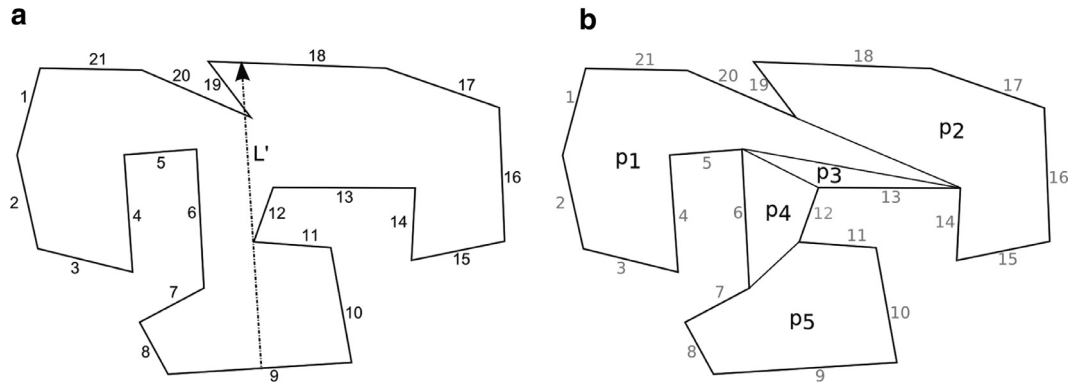

Fig. 15. Example of a complex area (a) that needs decomposition. In (b), the suggested decomposition.

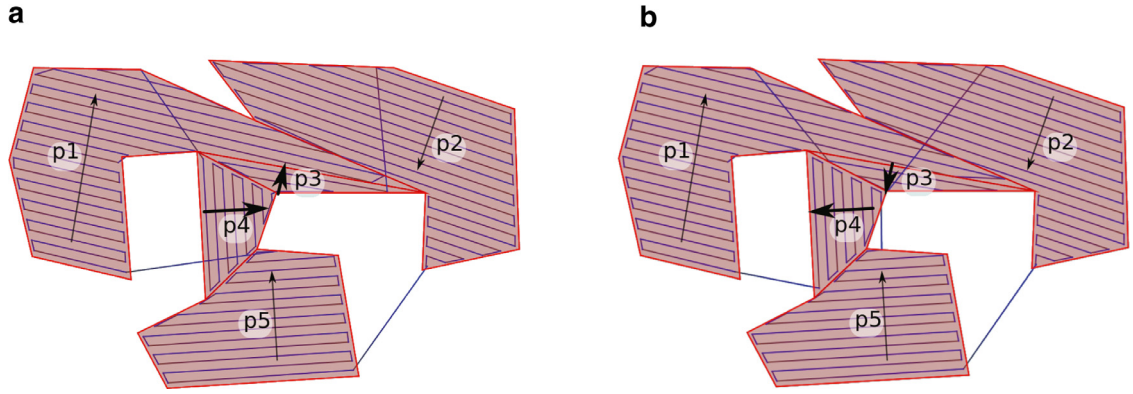

Fig. 16. Solutions obtained after solving the closed path. In (a), no adjacencies forced, $\widetilde{P}_1 = ((p_1, p_4, p_3, p_2, p_5), (A_2, A_1, A_3, A_2, A_2))$. In (b), the adjacencies were forced to 4 $\widetilde{P}_2 = ((p_1, p_3, p_2, p_5, p_4), (A_2, A_2, A_2, A_2, A_4))$.

L'' able to generate a non-interrupted path, the polygon has to be decomposed.

The result of a possible decomposition chosen for this example[2] is shown in Fig. 15 (b). The polygon is decomposed into five subpolygons. The reader may notice that the subpolygons p_1, p_2 and p_5 are not convex polygons. However, their corresponding line sweeps generate non-interrupted paths so they can be covered without further processing.

Now that the line sweep direction for each subpolygon has been determined, we need to solve the coverage problem on multiple polygons (using the closed path idea): the connection order φ has to be found and the coverage alternative for each polygon φ' should be determined. Recall that the algorithm studies every possible combination of order and coverage alternative, that is $\widetilde{P} = (\varphi, \varphi')$, and select the one that minimizes the sum of distances $f(\widetilde{P})$ (see Eq. (4)).

There are $\frac{(5-1)!}{2} \cdot 4^5 = 12288$ different permutations (considering the closed path idea) of order and coverage alternative for that polygon.

The algorithm's output is $\widetilde{P}_1 = ((p_1, p_4, p_3, p_2, p_5), (A_2, A_1, A_3, A_2, A_2))$ with total distance associated $f(P_1) = 274.722$ and 129 turns. The coverage path is shown in Fig. 16 (a).

Although in the example provided, the number of possible permutations of ordering and coverage is not very high, our algorithm can reduce the number of permutations generated forcing a number of adjacent connections (as explained in Section 6.2).

In the example, forcing 1 or 2 adjacencies led to a set of permutations that is equal to the one generated when no adjacencies were forced. In other words, in this particular example any permutation considered verifies that at least two adjacent polygons are consecutive in φ. Please note that this may not be true in general.

So, as the set of permutations is the same when considering 0, 1 or 2 adjacencies, the best corresponding solution $\widetilde{P} = (\varphi, \varphi')$ is also the same.

When three adjacencies are forced, just 8192 permutations are evaluated and within them, the same solution previously found appears (because the solution itself has three adjacencies). When

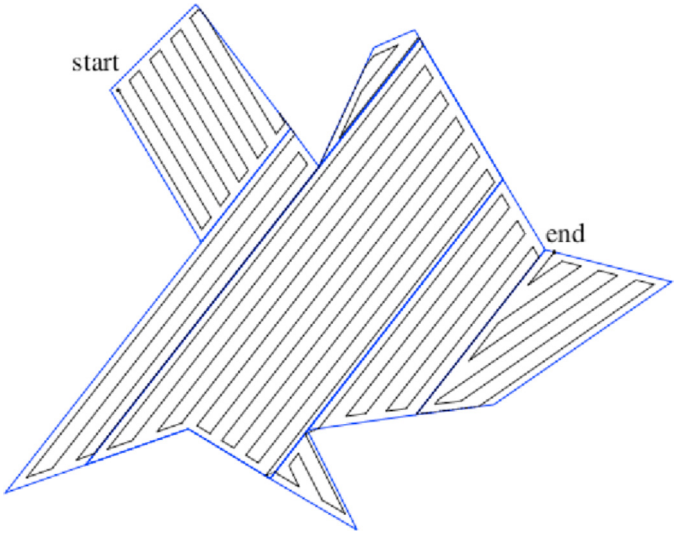

Fig. 17. Area and coverage example, taken from Li et al. (2011).

[2] The decomposition for this example is made trying to connect two concave vertex while minimizing the sum of the lengths of the line sweep of the subpolygons.

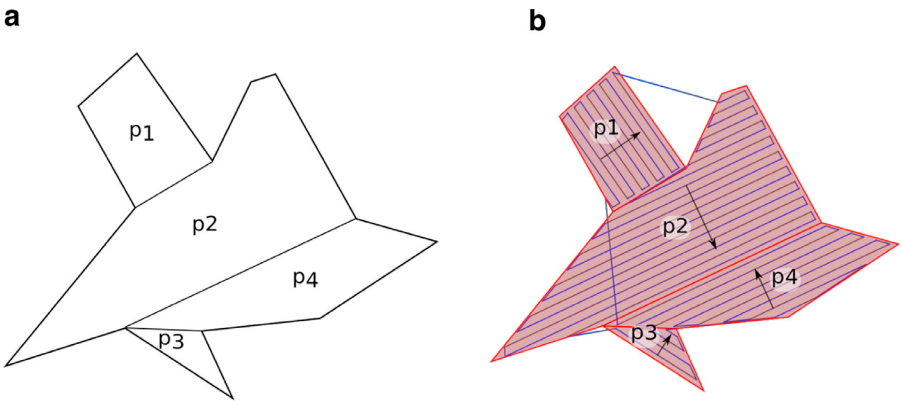

Fig. 18. Polygon's decomposition (a) and coverage path (b).

considering four adjacencies, 4096 permutations are explored and the previous best solution is not considered. The new best one is $\widetilde{P}_2 = ((p_1, p_3, p_2, p_5, p_4), (A_2, A_2, A_2, A_2, A_4))$ giving the coverage path shown in Fig. 16 (b). The distance associated is 275.627 and it has the same number of turns. In other words, checking 50% of the permutations (4096 vs. 8192) allowed us to obtain a new solution with just a 0.33% of increment in the total distance.

Regarding the computational effort required to solve this problem, the next table shows the variation on time and number of permutations for each number of adjacencies forced and the best solution's total distance.

Permutations		Time (s)	$f(\widetilde{P})$
$\lvert adj_1 \rvert =$	12288	21.69	274.72
$\lvert adj_2 \rvert =$	12288	21.97	274.72
$\lvert adj_3 \rvert =$	8192	14.95	274.72
$\lvert adj_4 \rvert =$	4096	6.14	275.62

7.2. Second example: Comparison against a published result

In this second example, we take the same area proposed in Li et al. (2011) and we compare our solution against the one published. It should be noticed that in the reference paper, the problem solved is slightly different as their algorithm also return the initial and final control ground points which are chosen after the path generation.

The solution proposed in Fig. 17, taken from Li et al. (2011) has 87 turns.

Departing from such example, Fig. 18 (a) shows the result of the decomposition. Then, our algorithm is able to obtain a solution with 80 turns as shown in Fig. 18 (b). Moreover, in our case, the ground control point can be located everywhere close to the path, not affecting the total number of turns.

Because n is low, no adjacency restrictions were considered. The output of the algorithm, Fig. 18 (b), is the order $\varphi = (p_1, p_2, p_3, p_4)$ and the coverage alternatives $\varphi' = (A_4, A_3, A_1, A_4)$ from a number of $\frac{(4-1)!}{2} \cdot 4^4 = 768$ permutations generated.

8. Conclusions and further comments

Three-dimensional terrain reconstruction from 2D aerial images is a problem of utmost importance due its wide level of applications. Here, we focused in solving the coverage path problem using a UAV for images capturing. The algorithm's main aim is to optimize the battery consumption, through minimizing the number of turns and the flight total distance. In contrast to other approaches our method can deal with both convex and non-convex regions. The proposed four coverage alternatives meet all requirements for the 3D reconstruction problem and achieve an improvement over the method that use only one coverage alternative, while creating shorter transitions between areas and/or the ground control point. In addition, we introduced the interrupted line sweep concept and its use to decide whether a non-convex polygon decomposition should be done or not.

Two illustrative examples showed the potential of our algorithm in two senses: ability to perform the coverage when complex regions are considered, and obtain better solutions than published results (in terms of the number of turns used).

Although it was not shown here, our algorithm returns together with the path a list of GPS coordinates of the turning points. In this sense, such output could be used by any UAV supporting waypoint navigation. The algorithm was successfully tested on a real environment where the output of our method was imported into the UAV's own flight control software, thus allowing the verification of other vehicle specific security constraints. It is important to remark that the software is available upon request.

Nowadays, the use of UAVs for image capturing is relevant in the context of intelligent and expert systems for disaster managements (for example to analyze a flooded area), soil analysis, earthquake crisis, civil engineering, urban planning, surveillance and defense research. Although the process of decision making from the captured images is far from trivial, the capture of such images verifying the overlapping constraints, time contiguity and so on, is crucial for the rest of the tasks. It is expected that the proposed algorithm, may have impact in several application areas due to the relevance of the coverage path planning problem.

Some concerns may arise about the algorithm regarding the computational complexity or the "combinatorial explosion" as the problem becomes bigger. It is necessary to recall that our objective is to provide a method for practical situations where the UAV's operator manually defines the area to cover. That means that the area is an approximation of the terrain and is usually simplified by using a simpler polygon that contains the complex terrain. Here, the "problem size" is associated with the number n of polygons (after decomposition) to cover. As we need to consider the possible permutations, in the worst case, our algorithm is $\mathcal{O}(n!)$. However, we provide two additional heuristics to restrict the number of permutations to consider. This allowed for example, when $n = 6$, to reduce the number of permutations from 1.474.560 to 245.760.

As further research, we envisage at least three clear points to address. First, is the consideration of UAV's autonomy in the planning. It may happen that the area needs more than one flight to be covered, thus there would be a need to detect the best "return to ground control" points along the path. Second, and as long as we know, research on coverage path planning using several flights is not usual although from a practical point of view is clearly

relevant. Finally, another important improvement for the method is to discuss the situation when the UAV can not fly outside of the area forcing more complicated transition between subpolygons.

Acknowledgments

D. Pelta and J.L. Verdegay acknowledge support from projects P11-TIC-8001 (Consejería de Economía, Innovación y Ciencia, Junta de Andalucía) and TIN2014-55024-P (Spanish Ministry of Economy and Competitiveness). Both projects include FEDER funds from the European Union.

M. Torres enjoys a Ph.D. research training staff grant associated with the project TIN2014-55024-P and co-funded by the European Social Fund.

J.C. Torres acknowledges support from projects TIN2014-60956-R (Spanish Ministry of Economy and Competitiveness) and G-GI3000/IDIC (Consejería de Obra Pública de la Junta de Andalucía). Both projects include FEDER funds from the European Union.

References

Bagnitckii, A., Inzartsev, A., & Senin, R. (2011). Facilities of AUV search missions planning. In *OCEANS'11* (pp. 1–7). IEEE Press.

Besada-Portas, E., De La Torre, L., Moreno, A., & Risco-Martín, J. L. (2013). On the performance comparison of multi-objective evolutionary UAV path planners. *Information Sciences, 238*, 111–125.

Choset, H. (2001). Coverage for robotics—A survey of recent results. *Annals of Mathematics and Artificial Intelligence, 31*, 113–126.

Choset, H., & Pignon, P. (1998). Coverage path planning: The boustrophedon cellular decomposition. In *Field and service robotics* (pp. 203–209). Springer.

Fernández, J., Tóth, B., Cánovas, L., & Pelegrín, B. (2008). A practical algorithm for decomposing polygonal domains into convex polygons by diagonals. *Top, 16*(2), 367–387.

Franco, C. D., & Buttazzo, G. (2015). Energy-aware coverage path planning of uavs. In *2015 IEEE international conference on autonomous robot systems and competitions (ICARSC)* (pp. 111–117). IEEE.

Galceran, E., & Carreras, M. (2013). A survey on coverage path planning for robotics. *Robotics and Autonomous Systems, 61*(12), 1258–1276.

Hert, S., Tiwari, S., & Lumelsky, V. (1996). A terrain-covering algorithm for an AUV. *Autonomous Robots, 3*, 91–119.

Huang, W. H. (2001). Optimal line-sweep-based decompositions for coverage algorithms. In *Proceedings of the IEEE international conference on robotics and automation, 2001 (ICRA'01): vol. 1* (pp. 27–32). IEEE.

Ji, X., Wang, X., Niu, Y., & Shen, L. (2015). Cooperative search by multiple unmanned aerial vehicles in a nonconvex environment. *Mathematical Problems in Engineering, 2015*, Article ID 196730, 19 pages.

Keil, J. M. (2000). Polygon decomposition. In *Handbook of computational geometry: vol. 2* (pp. 491–518). Elsevier.

Lee, T.-K., Baek, S.-H., Choi, Y.-H., & Oh, S.-Y. (2011). Smooth coverage path planning and control of mobile robots based on high-resolution grid map representation. *Robotics and Autonomous Systems, 59*(10), 801–812.

Levcopoulos, C., & Krznaric, D. (1998). Quasi-greedy triangulations approximating the minimum weight triangulation. *Journal of Algorithms, 27*(2), 303–338.

Levcopoulos, C., & Lingas, A. (1984). Bounds on the length of convex partitions of polygons. In *Foundations of software technology and theoretical computer science* (pp. 279–295). Springer.

Li, Y., Chen, H., Er, M. J., & Wang, X. (2011). Coverage path planning for UAVs based on enhanced exact cellular decomposition method. *Mechatronics, 21*(5), 876–885.

Maza, I., & Ollero, A. (2007). Multiple uav cooperative searching operation using polygon area decomposition and efficient coverage algorithms. *Distributed Autonomous Robotic Systems, 6*, 221–230.

Sujit, P., Sousa, J., & Pereira, F. (2009). UAV and AUVs coordination for ocean exploration. In *Oceans 2009-Europe* (pp. 1–7). IEEE.

Valente, J., Cerro, J. D., Barrientos, A., & Sanz, D. (2013). Aerial coverage optimization in precision agriculture management: A musical harmony inspired approach. *Computers and Electronics in Agriculture, 99*, 153–159.

Watts, A. C., Ambrosia, V. G., & Hinkley, E. A. (2012). Unmanned aircraft systems in remote sensing and scientific research: Classification and considerations of use. *Remote Sensing, 4*(6), 1671–1692.

Xu, Z., Yang, J., Peng, C., Wu, Y., Jiang, X., Li, R., et al. (2014). Development of an UAS for post-earthquake disaster surveying and its application in Ms7. 0 Lushan Earthquake, Sichuan, China. *Computers & Geosciences, 68*, 22–30.