A Decomposition Approach for the Complete Coverage Path Planning Problem*

Pedro Rocha, A.Miguel Gomes,

INESCPorto, Faculdade de Engenharia, Universidade do Porto,
Rua Dr. Roberto Frias s/n, 4200 – 465 Porto, Portugal
{pro10015, agomes}@fe.up.pt,

Abstract. In this paper, a complete coverage path planning problem is discussed, for application in real situations in the areas of agriculture, autonomous robotic house cleaning, and hydrographic survey. Since the approach to this problem is currently being developed, only the work developed so far is presented. In this problem one aims to find the most efficient path or circuit that completely covers a closed region. For the time being, no algorithms with universal real application capable of solving every variant of this problem exist. The proposed decomposition approach is based on dividing the initial region into convex sub-regions (which are easier to manipulate), followed by computing the full sequence of sub-regions to be traversed, and, finally, finding the full path inside each sub-region. This approach does not return an optimal solution to the complete coverage path problems, but its implementation define the basic steps for creating a software library that can be used to solve many geometrical based problems, in which these are included.

Keywords: Complete Coverage Path Planning, Autonomous Vehicle Routing, Algorithm Design, Computational Geometry.

1. Introduction

The Complete Coverage Path Planning problem (CCPPP) is a difficult problem to solve, where the main objective is to find the shortest path or circuit that covers an entire region. The degree of optimality of the solution has a direct effect on the energy consumption needed to completely cover a certain region, and also in the required amount of available time to do so. The resolution of these kind of problems have a broad range of real application situations, and can be widely used in tillage, planting, cultivating, spraying, among others. In many cases, specially in the agriculture applications [1], [2], the regions have forms bounded by natural obstacles like large stones, rivers, forests, and others. This reason causes the geometrical representation of the natural form to be an approximate of a complex geometric form, which can be considered irregular. Some other fields with some similarities with agriculture are autonomous robotic house cleaning [3], [4], [5], [6], in which a robot has to cover all the region of a surface with the goal of maximizing the cover but with the secondary objective of minimizing the overlapping of its path or circuit; and in the field of

* Partially supported by Fundação para a Ciência e Tecnologia (FCT) Project PTDC/EME-GIN/105163/2008 - EaGLeNest

hydrographic surveying [12], in which the vehicle makes the mapping of the underwater terrain and searching irregularities in the ocean. The particularity of the hydrographic survey problem is that, although it is limited by the same set of constraints, the area that is covered along its path is variable, since the visible area in the ocean floor depends on the field of view and distance to the same ocean floor, being greater with greater depths, smaller with lower depth, and the area monitored at one side of the path may be different from the area monitored on the other side when moving horizontally regarding a slope.

The proposed decomposition approach has also the objective of testing the capabilities, in performance as also in adaptability, of a geometric library focused in solving nesting problems, like two-dimensional irregular cutting-stock problems[13]. This library is being reconstructed to expand its functionality, performance and ease of use. Its development is progressed in a modular structure where each module has a specific application. With this kind of experiment, we hope to verify the current capability of the geometric library, to increase the diversification of its application areas and to further expand its functionality.

This paper is organized in 5 main sections. Starting with the Introduction, in Section 1. After it, in Section 2, it is presented an overview of the problem to solve, a general description of how it is usually solved, and also what tools are used to solve it. In Section 3, a detailed description of our proposed solution approach to the problem is given, with the methods and algorithms used, the requirements and choices made based on the constraints of the problem. Section 4 has the overview of the geometric library that is used as a tool to implement these algorithms, presenting some main functions used to build these algorithms, and its current state of development. In Section 5 the final comments and future work are presented. No results are given since this is a preliminary work, currently under development.

2. The Complete Coverage Path Planning Problem

The CCPPP includes many distinct cases of real application, in several distinct fields, one of them being the field of agriculture. It is usually represented by irregular geometrical forms, with a varied degree of complexity, which can include holes and curved segments. This complexity in its geometrical form causes some limitations in the possible ways that can be used to solve the CCPPP, so some approaches divide the initial geometrical form into more simpler forms to ease the problem resolution [1], [7]. Ideally, one would get the full path or circuit that would cover the entire region, without any kind of overlapping, with the minimum number of turns, if they represent and additional cost, in time or distance. Some methods divide the initial form in triangular, trapezoidal, rectangular, and other polygonal forms, to reduce a single complex problem into several simpler problems. After the division, the objective is to try to find the best path or circuit that covers all of the polygons, regarding the restrictions that increase or decrease the total cost.

Another methods proposed to deal with the CCPPP are based in rectangular cell decomposition [7], triangular cell decomposition [8], neural networks [9], genetic algorithms [10], but some of these methods have specific limitations. They need to have previous information about the area that they will have to go through, with the

size and location of the obstacles. That information may not be initially available. One example of an application with an initially unknown environment is [7], which makes the mapping of the terrain in real time, and adjusts its defined path when needed.
In the chosen solution, the approximate representation of the real form, for which is determined the maximum possible cover path or circuit with the minimum cost, is described by a set of regions which can have holes and disjoint regions. All the representations will initially be made with linear segments, although curved segments, circles, ellipses and splines are supported in the library. One example of an region divided into individual regions, and with the global path already determined is presented in the Fig.1.

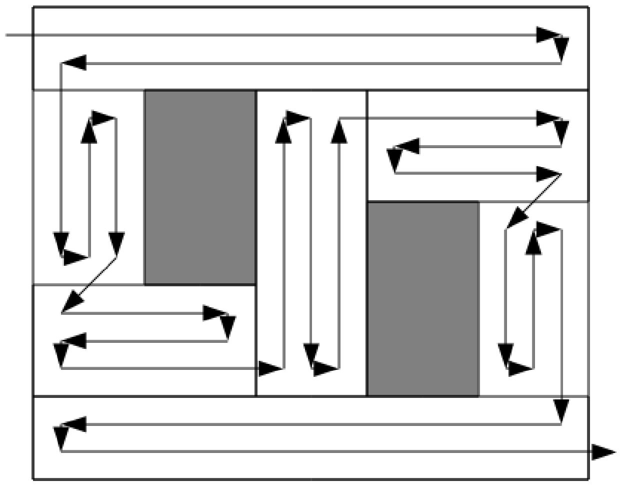

Fig.1. Global path example.

3. Solution Approach

In this work, the chosen method of resolution uses a decomposition approach to tackle the CCPPP, which uses some functions developed in the library. This decomposition approach has the advantage of being easier to implement, and to return results faster than other approaches. Other necessary functions that do not yet exist will be developed and inserted into an independent module on the EaGLeNest library when necessary.
Two solutions to solve two variants of the CCPPP are presented. The first one has the main objective of minimizing the number of turns in the complete circuit, while the second one, the main objective is to minimize the maximum length of the circuit, also achievable by minimizing the overlapping of the circuit path with itself.
Both solutions are divided in the same three phases. These three phases consist in decomposing the complex form into convex polygons, then finding the Hamiltonian circuit in the graph generated by connecting the centers of the convex polygons, and finally the computing of the complete cover path for all convex polygons. The two

solutions only differ in the third phase.

3.1. Convex Decomposition

In the first phase, the first step is to make the transformation of the real area into an approximate representation by connected linear segments, with the obstacles being defined as holes, but with a sequence of linear segments connected in inverse order compared to the outer layer. When a polygon is separated from the others (without any edge that connects it to the other polygons) it is treated independently. If some connection between invalid areas is allowed, we need to specify the extra cost to traverse it from one valid area to another. As such, as a first step, the holes are removed, by dividing the polygon in several parts, sectioned by an horizontal line positioned at the half of each hole.

In the end, we will get several parts of the initial form, without holes, with a maximum number equal to the number of holes plus one unit. Each of these elements will be further divided into triangular polygons. Each triangle is made from three sequential vertexes in a polygon, if they create a convex polygon in the inside of the same polygon. The outer vertex is ignored on the next step, and the process repeats itself until only 3 vertexes remain. The triangularization algorithm does not guarantee that the minimum number of triangles will be generated for any single piece. We can see that in Fig.2.

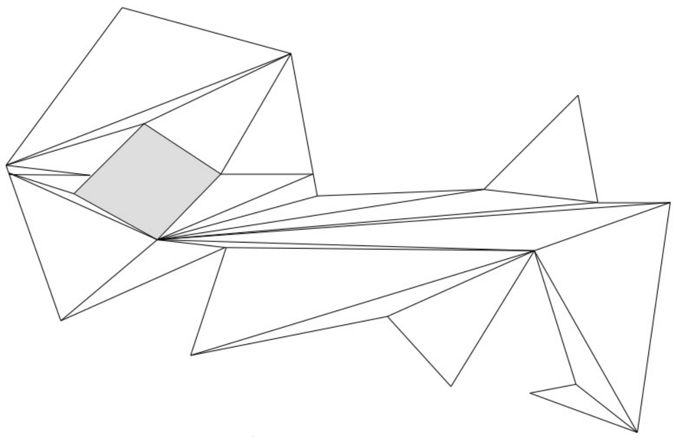

Fig.2. Complete triangularization of a complete form.

After the triangularization, all triangles are represented in a single mesh. This mesh allows that the transformation algorithm of triangular polygons into convex polygons tries to maximize the size of the convex polygons that may be build from a vertex that belongs to the outer layer, minimizing the total number of convex polygons generated. This transformation algorithm works by choosing an outer layer segment that has not yet been selected, and proceeding to the next segment that does not breaks the convexity criteria. In the event of not finding a next segment that complies with that restriction, it returns to the previously selected segment, marks the current segment as invalid, and continues the search from that point. In the worst case, the convex polygon generated is the same triangle from where the algorithm started. The effect of this can be seen in Fig.3.

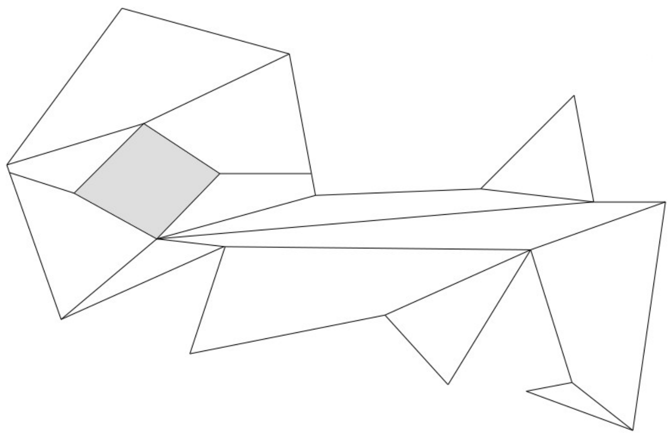

Fig.3. Convex polygons generated by the transformation algorithm.

3.2. Hamiltonian Circuit

The second phase starts by calculating the positions of the center of the convex polygons generated, and making the connections between them accordingly to shared segments between polygons. This step creates a graph with the connections between the center of each polygon connected to the other centers of polygons that share a segment. If they share just a vertex, that is not enough to make a connection.

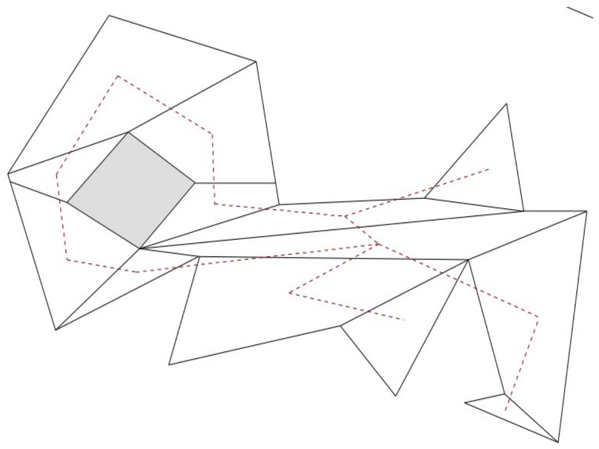

Fig.4. Equivalent graph generated from the center of the convex polygons and their contiguous polygons.

One needs to generate an unique complete circuit that reaches all the vertexes of the graph. This type of problem can be described as an equivalent to the Traveling

Salesman Problem, assuming that all the vertexes have a valid path to any other vertex. Since we cannot guarantee that a viable path exists between a given pair of vertexes, we compute the connections between any directly unconnected pair, with the minor cost between them, that goes through other intermediate vertexes. These virtual connections are added to the main graph, making the full Hamiltonian circuit[11] easy to compute, but however, it does not guarantee that the triangular inequality condition is satisfied.
To compute the full Hamiltonian circuit, we use a simple Traveling Salesman Problem heuristic, through a greedy algorithm that starts in any vertex, and proceeds to the next closest vertex, until it completes the path. At the end, it just connects to the starting vertex, and the full circuit is complete. Since all vertex have a connection to any other vertex, there is always a free connections to another vertex until all vertexes are included in the path.
The determination of the Hamiltonian circuit has the advantage of allowing to know which will be the edges of each convex polygons that will act as entry and exit points. Through these edges we can optimize the way the internal cover is made in each convex polygon, and also determine where the entry and exit point will be located in each entry and exit edge.

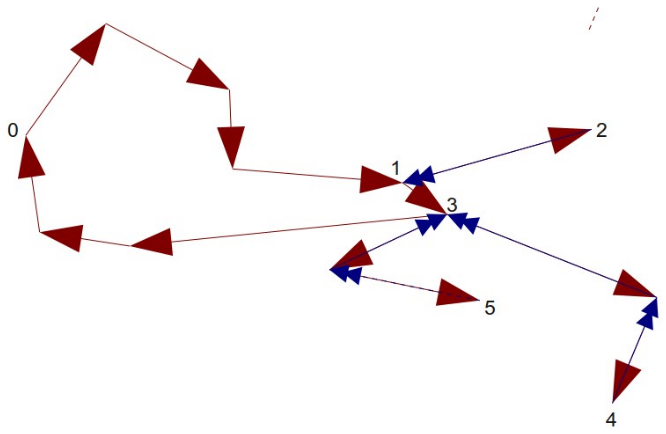

Fig.5. Hamiltonian Virtual Circuit in the graph.

The Fig.5. presents the virtual full Hamiltonian circuit that was computed in the graph. The connections represented by a single end arrow are normal connections. The double head arrows represent part of a virtual connection that is overlapping with the previously traversed path. The direction of navigation in the circuit of the graph is determined by the arrows. Since this circuit travels through the same vertexes more than one time, we cannot say that this is an Hamiltonian circuit, but it can be considered one if we convert the returning paths into alternative connections that connect only the end vertex of a real path, and a virtual free vertex that it returns to. The real circuit in the presented graphs starts in 0, proceeds to 1, goes to 2, returns to 1, advances to 3 and then 4, returns to 3, goes to 5, turns back to 3 and then finally ends with 0. However, the virtual Hamiltonian circuit does not return to previously traversed vertexes. The Hamiltonian circuit is 0, 1, 2, 3, 4, 5, 0.

3.3. Polygon Coverage

In the third phase, with the complete circuit already defined, and the entry and exit edges already selected, the complete circuit that covers the whole area can be built, with the goal of minimizing its cost, in two distinct ways, depending on the imposed conditions. If each turn implies an extended cost, a solution with the goal of minimizing the total number of turns might be preferred, even considering the increase of overlapping in the final circuit. In the opposite approach, if the turns do not add a significant cost, or if the cost of overlapping is minimal compared to it, the solution that minimizes the total overlapping is the favored resolution approach.

If we consider the option of minimizing the total number of turns, the proposed heuristic is described in the following paragraphs.

Ignoring the positioning of the entry and exit segments in every convex polygon, the coverage path of each polygon is achieved in a perpendicular pattern to the segment that connects the most distant pair of vertexes for every convex polygon. This minimizes the number of turns in the convex polygon coverage path [1].

Unfortunately, with this type of coverage path, the entry and exit points for each convex polygons are going to be modified. The added cost is the cost of the path overlapping necessary to connect one exit point from one convex polygon to the entry point of the next one. Even so, this type of solution might have an advantage in some instances of this kind of problem. The time that the path takes to be traversed is not taken into account, and so it is not considered as an added cost, but usually traveling a certain distance in a straight line is faster than traveling a same distance in a path with many turns.

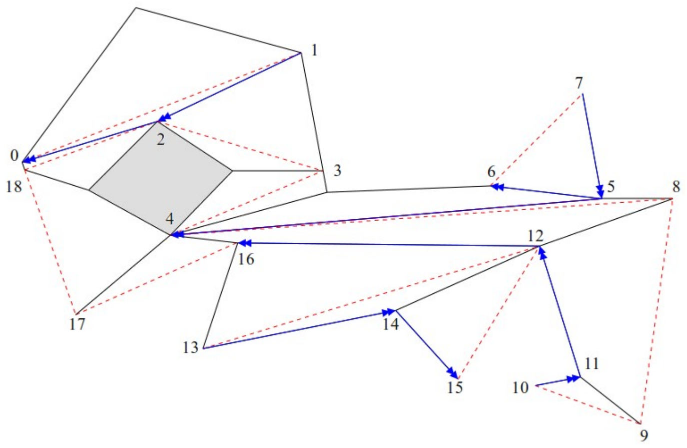

Fig.6. Complete path generated by the turns minimization heuristic.

In the previous image, the dashed lines indicate that the convex polygon which they are contained will be covered in a perpendicular pattern to that same line. The arrows with double end show the additional overlapping cost of connecting the entry and exit points between convex polygons. The full circuit, based on the solution from the previous step, follows the sequence defined in the image:

0, 1, 2, 3, 4, 5, 6, 7, 5, 8, 9, 10, 11, 12, 13, 14, 15, 12, 16, 17, 18, 2, 0.

When we consider the option of minimizing the overlapping of the circuit, we use an heuristic like the one proposed in the following paragraphs.

Taking into consideration the entry and exit edges of each convex polygon, a few particular cases might need some adjustments, so that the algorithm can generate a path without any overlapping.
When a convex polygon has several entry and exit points, to make a correct coverage the polygon needs to be divided accordingly to the number of the matched pairs of entry and exit points. This type of division divides a convex polygon with several entry and exit points, into several convex polygons with only a pair of entry and exit points, just like the example in Fig.7.

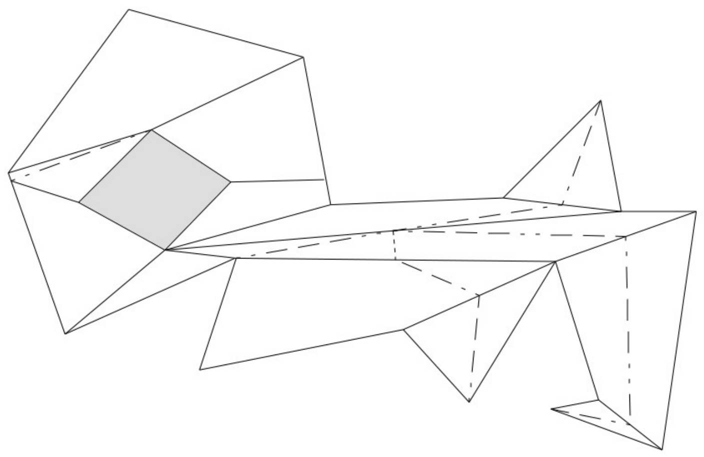

Fig.7. Convex polygon division into polygons with only one pair of entry and exit vertexes.

Still, another transformation might need to be done. We cannot always cover a convex polygon without any overlapping if we follow an exclusively perpendicular pattern to the line that connects the entry and exit vertexes, even when the vertexes are in contiguous edges.

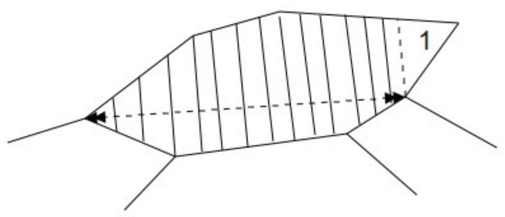

Fig.8. Invalid path coverage in a convex polygon.

As it is shown in Fig.8, the area marked by 1 cannot be covered if we follow the perpendicular pattern to the dashed line that connect the two most distant vertexes of the entry and exit segments.
As such, to solve this problem, the convex polygon is further divided, with a first division made from the common vertex, in the case of contiguous segments, or from

the last of the in-between outer layer segments that connect the vertex in the exiting edge up to the most distant vertex from it, that preferentially does not belong to the entry edge. One exception to this case, in which no division is made, is when the convex polygon is a triangle. In this exception, we cover the polygon with a parallel pattern relative to the entry edge. If the edges are not contiguous, we need to make an alternative kind of division, to make sure that the cover ends at the exiting edge. When an edge is simultaneously an entry and exit edge, that edge will be divided, connecting to an opposite free vertex or edge, also dividing the polygon in which it is contained. We can see an example in Fig.9.

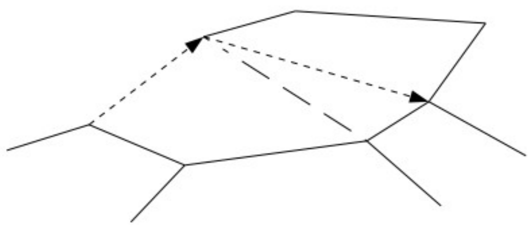

Fig.9. Main coverage lines after division.

This method allows to have a single path, without any overlaps, that minimizes the total distance traveled when comparing to the alternate solution. As a consequence, we now have a high number of turns, that can severely worsen the cost if they are taken into account, due to the further division of convex polygons, increasing their number, and creating the need to the individual coverage of every independent component.

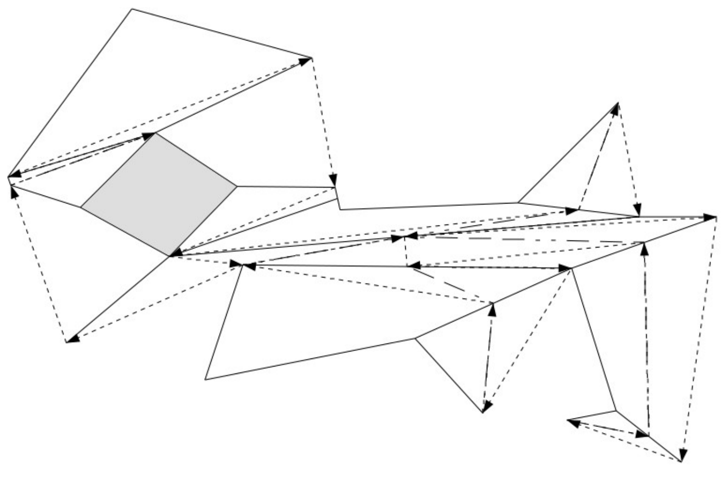

Fig.10. Coverage path created with the overlapping minimization heuristic.

The arrows in Fig.10 present the complete path without any overlapping, generated by the described heuristic. The coverage pattern is perpendicular to the correspondent line contained in each convex polygon.

4. Implementation

The implementation of these algorithms will rely in some existing functions from the EaGLeNest geometric library. This reassures the capabilities of this library to diversify the range of practical application, having the simultaneous advantage of testing and develop this tool to be applied in this kind of problems.
Some functions might need to be slightly modified, depending on the problem, but no heavy modification of functions will occur, since it is preferred to develop an independent function from scratch with a particular use in mind.
The data to any problem, be it vertexes, edges, polygons, and groups of polygons, and also algorithms to use in the resolution of the problem with their execution sequence, are contained in a XML file, which is the default file to use with the library. This format ensures backwards compatibility with the current version of the library, while also allowing the ease of reading, construction and manual modification of the contents of the file by any person. The main data structures are based on simpler one's, with identifiers. The most elementary unit is the vertex, or dot (with identification). With two vertexes we build a linear segment and its identification, and with an array of segments we build a polygon, also with identification. The higher hierarchical structure available is designed by geometrical shape, which can contain several polygons that might not be contiguous with any other, not be defined as holes, nor contained in another polygon's hole. These structures are also designed to support curves, with Bezier representation, including also ellipses and circles (and arc if an angle is specified), but the functions used to manipulate these forms are not yet implemented in the geometric library.
The most basic functions of the geometric library that were implemented to this moment are:
Reading data in XML file format, and loading the data structures accordingly; intersection detection between segments, that return the intersection points (be it linear segments or curved segments); computation of the minor angle between the intersection point generated with the intersection of two segments; the lesser distance of a vertex to a straight line; position of a vertex relative to a straight line segment (to the left, right or co-linear); the linear segment most to the left or to the right of a current selected segment; the nearest linear segment to the left or to the right of a current selected segment; determine if two vectors are oriented to the same quadrant; determination of the representation of a polygon (if it is a hole or an outer layer, according to the settings specified in the system); inversion of a polygon (transform a hole into an outer layer and vice-versa).
The functions composed by these basic functions of the geometrical library are:
Transformation of a polygon with holes into several polygons without holes; triangulation of irregular geometrical forms, construction of lists that contain every reachable vertex from a specified vertex (useful to determine which vertexes belong to a polygon); decomposition of a mesh into individual polygons; function that returns only the outer layer of a mesh and another that subtracts only the outer layer from a

mesh; computation of biggest convex polygons in a mesh (with exclusion for partition, and inclusion for coverage of polygons); construction of an adjacency matrix from a mesh; merging of geometrical forms, and finally the No-Fit Polygon construction from a pair of convex polygons.
The functions used in the geometric library, that can be used in the resolution of the problem discussed in this paper, are mostly the elementary functions, including a few of the more complex functions like the transformations of convex polygons in polygons without holes, the triangularization, and merging of polygons. The functions destined to the computation of the coverage path of the convex polygons are currently being implemented. The functions destined to the computation of Hamiltonian paths, and to the resolution of variants of Traveling Salesman Problems will be implemented later.

5. Final Comments

Since this is still a preliminary work, we cannot solve for every best solution possible with these heuristics, but we can use them to attempt to get solutions that achieve the minimum amount of turns or the minimum overlapping.
The geometric library has only some basic functionality, but already shows some good flexibility and support for generic applications, when considering usage of some of the implemented functions described in the previous section (4). It cannot be used extensively in any area, but can be used to start solving these problems in the fields of agriculture, hydrographic surveys, autonomous robotic house cleaning, and a few others.

As future work we expect to fully implement the modules presented in this work. We then proceed to collect the results from some variants of this proposed problem, and make some adjustments to improve performance. We also plan to improve the algorithms used in this paper, and continuously improve the geometric library to support more features. Most of the features have the possibility of being used in other kinds of problems that can have a representation based on geometrical forms.

References

1. G. Zuo, P. Zhang, and J. Qiao, “Path planning algorithm based on sub-region for agricultural robot,” in Proceedings of the 2nd international Asia conference on Informatics in control, automation and robotics - Volume 2, CAR’10, pp. 197–200, IEEE Press, 2010.
2. T. Oksanen and A. Visala, “Coverage path planning algorithms for agricultural field machines,” J. Field Robot., vol. 26, pp. 651–668, August 2009.
3. H. Choset, “Coverage for robotics – a survey of recent results,” Annals of Mathematics and Artificial Intelligence, vol. 31, pp. 113–126, 2001.
4. De Carvalho, R.N.; Vidal, H.A.; Vieira, P.; Ribeiro, M.I.; , "Complete coverage path planning and guidance for cleaning robots ," Industrial Electronics, 1997. ISIE '97., Proceedings of the IEEE International Symposium on , vol.2, no., pp.677-682 vol.2, 7-11 Jul 1997.
5. X. Wang and V. L. Syrmos, “Coverage path planning for multiple robotic agent-based inspection of an unknown 2d environment,” Mediterranean Conference on Control

and Automation, pp. 1295– 1300, 2009.

6. R. Mannadiar and I. Rekleitis, “Optimal coverage of a known arbitrary environment,” in Robotics and Automation (ICRA), 2010 IEEE International Conference on, pp. 5525 –5530, May 2010.
7. H. Choset and P. Pignon, “Coverage path planning: The boustrophedon cellular decomposition,” in In International Conference on Field and Service Robotics, 1997.
8. J. S. Oh, Y. H. Choi, J. B. Park, and Y. Zheng, “Complete coverage navigation of cleaning robots using triangular-cell-based map,” Industrial Electronics, IEEE Transactions on, vol. 51, pp. 718 – 726, June 2004.
9. S. Yang and C. Luo, “A neural network approach to complete coverage path planning,” Systems, Man, and Cybernetics, Part B: Cybernetics, IEEE Transactions on, vol. 34, pp. 718 – 724, February 2004.
10. A. Ryerson and Q. Zhang, “Vehicle path planning for complete field coverage using genetic algorithms,” Agricultural Engineering International: the CIGR Ejournal, vol. IX, July 2007.
11. Reinhard Diestel – “Graph Theory” - Springer - Verlag New York 1997,2000 – Page 214, theorem 10.1.1.
12. Cheng Fang; Anstee, S.; , "Coverage path planning for harbour seabed surveys using an autonomous underwater vehicle," *OCEANS 2010 IEEE - Sydney* , vol., no., pp.1-8, 24-27 May 2010.
13. Julia A. Bennell, Jose F. Oliveira, “The geometry of nesting problems: A tutorial”, European Journal of Operational Research, Volume 184, Issue 2, 16 January 2008, Pages 397-415.