REPORT DOCUMENTATION PAGE

Form Approved OMB No. 0704-0188

1. REPORT DATE *(DD-MM-YYYY)*	2. REPORT TYPE	3. DATES COVERED *(From – To)*
21-08-2006	Final Report	1 October 2003 - 01-May-06

4. TITLE AND SUBTITLE

Intelligent Control Management of Autonomous Air Vehicles

5a. CONTRACT NUMBER
FA8655-03-1-3064

5b. GRANT NUMBER

5c. PROGRAM ELEMENT NUMBER

6. AUTHOR(S)

Professor Mario Innocenti

5d. PROJECT NUMBER

5d. TASK NUMBER

5e. WORK UNIT NUMBER

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
University of Pisa
Via Diotisalvi 2
Pisa 56126
Italy

8. PERFORMING ORGANIZATION REPORT NUMBER

N/A

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)

EOARD
PSC 821 BOX 14
FPO 09421-0014

10. SPONSOR/MONITOR'S ACRONYM(S)

11. SPONSOR/MONITOR'S REPORT NUMBER(S)
Grant 03-3064

12. DISTRIBUTION/AVAILABILITY STATEMENT

Approved for public release; distribution is unlimited.

13. SUPPLEMENTARY NOTES

14. ABSTRACT

There are many issues in the general area of cooperative control of unmanned vehicles; one of particular interest is cooperative path planning and mission planning in a dynamic scenario with moving targets and moving obstacles. A dynamic scenario prevents usually the use of many algorithms due to their inherently high computational cost. The report briefly overviews some existing procedures used to solve both path planning and mission planning problems, and then proposes alternative algorithms which have a lower computational cost. In particular, we propose a path-planning procedure based on the Constrained Delaunay Triangulation, and the geometric properties of the in-centers of triangles. This procedure is not optimal from the analytical standpoint but it has several advantages for real-time applications because it allows slower sampling times and produces safer paths. The proposed path planning method takes into account areas of the scenario that may be more dangerous for the flight vehicle, by simply summing a term to the length of each sub-path depending of the dangerousness of the zone it crosses. The report presents also a sub-optimal mission planning algorithm based on a dynamic clustering of the targets in order to have a less myopic view of the entire scenario. The procedure is feasible in terms of total computational load, with respect to an optimal solution, which is known to be NP-hard and not achievable in polynomial time.

15. SUBJECT TERMS
UAV, unmanned aerial vehicle, cooperative control, autonomous control

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18, NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
a. REPORT	b. ABSTRACT	c. THIS PAGE			BARRETT A. FLAKE
UNCLAS	UNCLAS	UNCLAS	UL	29	**19b. TELEPHONE NUMBER** *(Include area code)* +44 (0)20 7514 4285

Intelligent Control Management of Autonomous Air Vehicles

Final Report

Grant: FA8655-03-1-3064

Submitted to:

EUROPEAN OFFICE OF AEROSPACE RESEARCH AND DEVELOPMENT

Submitted by:

Prof. Mario Innocenti

Date submitted:

July 2006

University of Pisa
Department of Electrical Systems and Automation
Via Diotisalvi 2, 56126 Pisa Italy

Summary

There are many issues in the general area of cooperative control of unmanned vehicles; one of particular interest is cooperative path planning and mission planning in a dynamic scenario with moving targets and moving obstacles. A dynamic scenario prevents usually the use of many algorithms due to their inherently high computational cost. The report briefly overviews some existing procedures used to solve both path planning and mission planning problems, and then proposes alternative algorithms which have a lower computational cost. In particular, we propose a path-planning procedure based on the Constrained Delaunay Triangulation, and the geometric properties of the in-centers of triangles. This procedure is not optimal from the analytical standpoint but it has several advantages for real-time applications because it allows slower sampling times and produces safer paths. The proposed path planning method takes into account areas of the scenario that may be more dangerous for the flight vehicle, by simply summing a term to the length of each sub-path depending of the dangerousness of the zone it crosses. The report presents also a sub-optimal mission planning algorithm based on a dynamic clustering of the targets in order to have a less myopic view of the entire scenario. The procedure is feasible in terms of total computational load, with respect to an optimal solution, which is known to be NP-hard and not achievable in polynomial time.

The activity described in the summary was performed during the last year of the grant. The topics studied within this year dealt primarily with path planning and mission planning issues in cooperative control. The principal investigator of the project was Prof. Innocenti. In addition, Dr. Lorenzo Pollini (assistant professor) and Mr. Andrea Bracci (Ph.D. student) participated to the research work.

1. Introduction

In the past few years cooperative control of a team of unmanned vehicles (UAVs) has become an area of increasing research interest. Teams of autonomous vehicles can be used for instance in military applications, search and rescue missions, fire protection, and other cases in which human presence can be endangered. Two very challenging problems encountered in cooperative control are the path planning and the mission planning for a team of UAVs in a dynamic scenario with moving and unknown targets and obstacles. Another important issue is the control of autonomous aerial vehicles in order to keep a predefined formation shape; this is known as the formation-flight problem.

The objective of the present report is to discuss and propose some strategies of cooperative control of UAV teams. The main idea is that cooperation can lead to better performance than non-cooperating vehicles; due to the fact that different vehicles can share different information about the environment depending on their own position and capabilities, and so a much larger view of the entire scenario can be obtained combining all this partial knowledge.

There are mainly two different coordination strategies: centralized and decentralized cooperative control. The former requires a central unit with great computational capabilities, which is able to plan the entire mission and then transmit the results to every vehicle. The latter requires many computational units (at the most one for each vehicle) with less computational capabilities, but with the added complication of vehicle-to-vehicle communications.

The report is organized as follows: first some results in the context of both path-planning and mission-planning will be reviewed, showing advantages and disadvantages of the different approaches. Then a new approach will be presented, dealing with some problems we encounter in cooperative control. More specifically, we introduce a fast and safer path-planning procedure that takes care of the presence of dangerous paths. This procedure is based on a dynamic version of the

Constrained Delaunay Triangulation (DCDT), and the geometric properties of the in-centers of adjacent triangles. The resulting path is suboptimal but has the advantage of a higher safety for real-time applications, especially in the case of very close obstacles, yielding a larger distance from the obstacles. The danger of certain zones is quantified by introducing an additional cost to every path section depending on the "risks" that a vehicle encounters traveling along that section. It will be shown that the additional cost does not affect significantly the overall computational load of the path-planning procedure, and can be used for online computation as well.

The proposed mission planning algorithm uses a clustering approach in order to obtain a less myopic and more scalable assignment procedure. The algorithm is based on the assumption that close targets are likely to be visited by the same vehicle, thus they can be grouped into the same cluster. Once the clusters are determined, each vehicle is initially assigned to a specific cluster. Two problems were encountered using the previous clustering approach; the first was the effect of the obstacles, and the second was the choice of the right number of clusters. The first problem was solved by using some pseudo-coordinates (the cost of the path linking a target to the others). The second was solved with a dynamic procedure capable of finding the adequate number of clusters depending on a design parameter related to the maximum size of each cluster.

2. Path Planning in a dynamic Environment

Path planning in a dynamic environment is very critical in the context of cooperative control due to the presence of moving obstacles and moving targets, among other factors. In the present context, the term "obstacles" is used to represent environmental entities such as mountains and buildings, as well as other elements of the scenario like no-fly zones with a potentially high level of dangerousness for the vehicles (military defenses, radar monitored areas, clouds of toxic material, zones with highly varying winds, temperature, atmospheric pressure, etc.).

In the case of no-fly zones, in particular, we may encounter moving obstacles, which prevent the use of pre-computed flight paths, and make the entire path planning process more complex. In the general case, in fact, such obstacles move/appear in an unknown and unpredictable fashion, and the path planning strategy must react preventing possible dangerous situations and re-creating a new feasible path. The concurrent presence of targets (in addition to obstacles) makes the path planning problem even more challenging, whether they are static, dynamic, and with dynamics known with different levels of accuracy.

The dynamicity of the scenario obviously makes off-line path planning inadequate, and fast path planning procedures are required to obtain better performance. In this section we present many different approaches to deal with a dynamic environment focusing on the advantages and the disadvantages of each. We always refer to a bi-dimensional scenario, and we consider that real obstacles are polygonal and enlarged by a factor such that we can neglect vehicles dimensions that, in turn, can be considered as point mass elements. Moreover we assume that targets be represented by points that must be visited.

2.1 Visibility Graph Procedure

This procedure is optimal, and it is based on a graph search of the minimum-cost path between two nodes. In the context of cooperative control the nodes represent typically the vehicles, the targets, and the vertices of the obstacles. The procedure can be summarized with the following three steps:

a. Consider the points defined by vehicles and targets positions in addition to the vertices of the obstacles.
b. Create the Visibility Graph (VG) linking each node with every other node and neglect non-admissible edges (edges intersecting at least one obstacle). The cost of each edge is equal to the length of the path linking the two points.

c. Denote with ID_{start} and ID_{end} the starting (vehicle), and the ending (target) node respectively and find the shortest path between ID_{start} and ID_{end} using the Dijkstra's algorithm.

The sequence resulting from Dijkstra's algorithm is the shortest path from a vehicle to a target and therefore we must run the algorithm for every pair vehicle-target. We point out that the VG must be found only once.

Now we look at the total computational cost of the entire procedure. In the following we denote with *n* the total number of nodes of VG which is given by the vertices of the obstacles and the two points relative to the vehicle and the target (supposing that there is only one vehicle and one target).

Construction of the visibility-graph.

To construct the visibility graph we need:

- Total number of edges (including non admissible edges) $N_E = \frac{n(n-1)}{2}$
- Total number of obstacle-edges $N_{OE} = n - 2$
- For every edge we must verify if it is admissible or not. This can be done verifying if the edge intersect or not an obstacle edge. The total cost of this operation is $C_A = N_E N_{OE} = \frac{[n(n-1)-(n-2)](n-2)}{2}$
- The total number of admissible edge is identified with N_{AE} and it depends on the scenario configuration.

Optimal path calculation.

The computation of the optimal path is a direct consequence of the value of the minimum cost:

- Once the visibility graph is found, the cost of the Dijkstra's algorithm with a Fibonacci heap is $C_{DA} = O(n \log n + N_{AE})$

- Summing up all the costs, the total complexity of the procedure is found to be $O(n^3)$.

We point out that the bottleneck of the entire procedure is the construction of the *visibility graph*. The advantage of this procedure is of course that it is able to produce the optimal path for every vehicle-target pair, but it becomes unfeasible for on-line implementation because of its high computational cost.

2.2 Fixed Tessellation Procedures

In order to achieve a faster path-planning procedure we can use a fixed tessellation of the scenario. The main idea is to perform the major part of the computation off-line, and use it for on-line implementation. We present different approaches based on a fixed tessellation and we will unfortunately conclude that we have no great advantages in using such fixed tessellations.

General properties

We first state some important results of path planning in tessellations. In the remainder of this sub-section we always consider the problem of finding the shortest path between two points in the plane, in presence of static obstacles. We use the following notation:

L_{VG}^{O} - The optimal path between the two points on the visibility graph.

L_{VG}^{NO} - A generic non-optimal path between the two points on the visibility graph.

L_{T}^{O} - The path on the tessellation corresponding to L_{VG}^{O}

L_{T}^{NO} - The path on the tessellation corresponding to L_{VG}^{NO}

We have assumed implicitly that if L_{T}^{O} and L_{T}^{NO} are reduced (with a path reduction procedure) we obtain L_{VG}^{O} and L_{VG}^{NO} respectively. Finally, we use the operator $|\cdot|$ to indicate the total length of a path. The main relations between the paths presented above are the following:

$$\left|L_{VG}^{O}\right| < \left|L_{VG}^{NO}\right|$$

$$\left|L_{VG}^{O}\right| \leq \left|L_{T}^{O}\right|$$

$$\left|L_{VG}^{NO}\right| \leq \left|L_{T}^{NO}\right|$$

Nothing can be said about the relationship between L_T^O and L_T^{NO}. T his is the main problem and it needs a more accurate study. The problem is due to the fact that we determine the path along the edges of the tessellation and then we reduce it finding the corresponding path on the visibility graph. Obviously we are searching for the optimal path on the visibility graph; so we hope that in every case we'll have $\left|L_T^O\right| < \left|L_T^{NO}\right|$ so that we could always recover the optimal path. Unfortunately this condition is not always verified, and there can be cases in which we select a path that, once reduced, is not the optimal one. We can state the following proposition based on the above which concepts:

Proposition 2.2.1 – Given L_{VG}^O, L_{VG}^{NO}, L_T^O, L_T^{NO} and a tessellation which produces a maximum relative error $e > 0$; and given $\left|L_{VG}^{NO}\right| < (1+e)\left|L_{VG}^{O}\right|$, then if we choose the path along the tessellation we are not sure that this corresponds to the optimal path on the visibility graph. In other terms there are not warranties that $\left|L_T^O\right| < \left|L_T^{NO}\right|$.

Proof: From the hypothesis we have the following relations:

$$\left|L_{VG}^{O}\right| < \left|L_{VG}^{NO}\right|$$

$$\left|L_{T}^{O}\right| \leq (1+e)\left|L_{VG}^{O}\right|$$

$$\left|L_{T}^{NO}\right| \leq (1+e)\left|L_{VG}^{NO}\right|$$

Then there can be cases in which we can find an e_1 such that $0 < e_1 < e$ and $\left|L_T^{NO}\right| = (1+e_1)\left|L_{VG}^{NO}\right|$ and $L_T^O > L_T^{NO}$. ■

We can verify the previous proposition in a simple case: the optimal path on the visibility graph is in the worst condition (the relative error is equal to e) and the non-optimal path on the visibility graph is in the best condition, which means that this path is completely covered by some edges of the tessellation ($e_1 = 0$). Obviously this case verifies the hypothesis of Proposition 2.1 and so this is verified. Conversely we also have the following result:

Proposition 2.2.2 - Given L_{VG}^O, L_{VG}^{NO}, L_T^O, L_T^{NO} and a tessellation which produces a maximum relative error $e > 0$; and given $\left|L_{VG}^{NO}\right| > (1+e)\left|L_{VG}^O\right|$ for every non-optimal path, then the following relation holds:

$$\left|L_T^O\right| < \left|L_T^{NO}\right|$$

Proof: From the hypothesis we have the following relations:

$$\left|L_T^O\right| \le (1+e)\left|L_{VG}^O\right| < \left|L_{VG}^{NO}\right| \le \left|L_T^{NO}\right|$$

which proves the proposition. ■

The last proposition says that if the relative error between every non-optimal path to the optimal path on the visibility graph is greater than e, then, moving on the tessellation we can always recover the optimal path. From the two previous propositions it is clear that we must find a tessellation that produces the least possible maximum relative error. Next we create a tessellation that assures that the relative error is limited by a known value.

Through a numerical procedure it has been found that the optimal shape of the polygons of a fixed tessellation is the equilateral triangle. The choice of a triangle is due to the fact that this is a convex polygon and every vertex is directly linked to the others, and there are not neglected internal paths. Moreover a tessellation mode of equilateral triangles allows the minimum value of the maximum relative error we commit by moving on the tessellation instead of a straight line. This error is about 15.5%. By allowing some "cuts" in the paths we can reduce this value to about 1.5% which is an admissible one.

The presence of the obstacles affects the regularity of the tessellation and the maximum error we obtain. There are substantially three ways to deal with obstacles:

a. Every obstacle-edge is subdivided in many parts with a given number of points that are forced to be part of the tessellation.
b. The non admissible triangles (which are triangles completely or partially intersecting an obstacle) are neglected and the remaining triangles form the resulting tessellation.
c. Insert the obstacle-vertices as node of the tessellation linking each of them to the vertices of the triangle containing it.

We next consider advantages and disadvantages of the three possibilities:

a. The resulting tessellation is refined in proximity of the obstacles but the triangles are not identical, so there are not warranties on the maximum relative error because it can only be found in the case of regular tessellation.
b. The resulting triangles are identical and if it is necessary the size of some triangles can be reduced in order to have a more precise bounding of the obstacles. The maximum relative error is bounded by the maximum relative error of the shape of the triangles.
c. The resulting triangles are all equilateral and the nodes of the visibility graph are all part of the tessellation. So this method allows larger tessellations and in some special cases it is capable of recovering directly the visibility graph-path without any reduction.

In all the three methods we find the path between two points in the scenario by recognizing which triangles enclose the initial and the final point and we run Dijkstra's algorithm between every vertex of the first triangle to every vertex of the final one. Then we add to every path the distance between the initial and the final points to the vertices of their respective triangle, and select the shortest path. The three methods require the initial tessellation, so we must grid the scenario with many points corresponding to the vertices of the equilateral triangles.

The procedure for creating a tessellation is as follows:

- Indicating with N_B and N_H the number of points in the base and in the height of the rectangle enclosing the scenario, the total number of triangles is $N_T = (2N_B - 3)(N_H - 1)$
- The total number of edges is approximately $N_{ET} = 3N_B N_H$.
- Once the edges have been found, we must verify which of them are entirely or partially inside an obstacle. So all the N_{ET} edges must be verified with the N_{OE} edges. Hence this procedure has a cost $C_{EV} = O(N_{ET} N_{OE}) = O(N_B N_H N_{OE})$.
- We note that the total cost of the verification procedure can be reduced to $C_{EV} = O((2N_B + 3N_H)N_{OE})$ by considering the intersections of the edges of the obstacles with the "super-edges" obtained by linking the aligned edges of the regular tessellation. By this way we can reduce the number of edges to be verified from $N_B N_H$ to $2N_B+3N_H$.

Now let us evaluate the total computational cost of the three methods:

a. The total cost depends strictly on the number of points we select on every edge. Suppose that we add N_{AP} points on every edge, the total number of nodes is $N_1 = N_B N_H + N_{AP} N_{OE}$. The cost of Dijkstra's algorithm is $9O(N_1 \log N_1 + E_1)$ where E_1 is the total number of admissible edges.
b. In this case the total cost depends on the desired refinement degree: if we use a mean refinement factor of two (that is: every triangle is divided into four triangles) we obtain a

total number of nodes equal to $N_2 = 4N_B N_H$ yielding a total number of edges $N_{E2} = 6N_B N_H + 3N_T$. It follows that the total cost of Dijkstra's procedure is $9O(N_2 \log N_2 + E_2)$ where E_2 is the number of admissible edges.

c. Using this procedure we add $n-2 = N_{OE}$ vertices to the tessellation and a maximum of $3(n-2)$ edges to the resulting graph. Hence the total number of nodes is $N_3 = N_B N_H + N_{OE}$ and as in the previous two cases the cost is $9O(N_3 \log N_3 + E_3)$.

Summing up all the costs we obtain:

$$C_a = C_{EV} + 9O(N_1 \log N_1 + E_1)$$

$$C_b = C_{EV} + 9O(N_2 \log N_2 + E_2)$$

$$C_c = C_{EV} + 9O(N_3 \log N_3 + E_3)$$

We focus on the fact that, independently from the refinement factor of the first two procedures, the following relationships hold:

$$N_3 \le N_1$$
$$N_3 \le N_2$$

Hence we can conclude that the third procedure is cheaper than the others. In order to have advantages (in terms of computational cost) in using procedure *c*, compared to the optimal procedure, we must have $C_c = O(n^k), k < 3$, so we must find ad adequate gridding of the scenario. In the worst case we can assume that $E_3 = N_{ET}$. If we choose $N_B = N_H = n$ (which is a reasonable choice compatible with the scenario) we obtain:

$$C_c = O(2n^2) + 9O(2n^2 \log n + n^2) = 9O(2n^2 \log n)$$

then if $18 \log n < n$ this procedure is cheaper than the optimal one. Solving $18 \log n < n$ for *n* positive integer we obtain $n > 16$.

To improve on the previous result, we can link the initial point and the final point to the vertices of the triangles enclosing them respectively and then run Dijkstra's algorithm once. The

total cost becomes $C_c = O(2n^2 \log n)$. Numerical simulations have shown that for n large the VG procedure is slower than the tessellation procedure, but for small n the former is faster because in the latter there are many terms depending on n^2. Since the computational cost of the VG procedure and the non-optimal one are very similar we will now consider alternate tessellation methods.

2.3 Euclidean Shortest Path

In 1999 Hershberger and Suri [1] found an optimal solution of the Euclidean Shortest Path in the plane with many obstacles with complexity O($n$log$n$) in time where n is the number of polygon vertices. Their algorithm is based on the so-called continuous Dijkstra's algorithm and uses the wave front propagation method. We refer [1] for the details of the algorithm.

2.4 CDT-based Path Planning

Although the result of Hershberger and Suri is very good in view of its low computational cost, we point out that if a vehicle follows an optimal path it is not sure that it will not collide with any obstacles. This is due to the fact that the optimal trajectories pass through obstacle edges and in real-time applications there will be always a selected sampling time, and so in the inter-sampling time there is the possibility of collision with a moving obstacle. To take care of this, non optimal paths must be considered as well.

2.4.1 Kallmann Procedure

In [2] Kallmann proposed a fully dynamic Constrained Delaunay Triangulation (CDT), which can be very effective in a dynamic environment because it only changes the moving edges (which are

the obstacles edges) yielding a lower computational cost. The procedure proposed by Kallmann can be summarized in the following steps:

a. Perform an initial CDT, which can be run in $O(n\log n)$ time (see [3]) and keep track of the constrained edges with the data structure defined in [2].
b. At every time step modify the actual CDT in order to consider moving obstacles (which means: moving constrained edges).
c. Define an initial point (vehicle) and a final point (target). Find the shortest adjacency chain between the two triangles enclosing the two points on the CDT without crossing any constrained edge. This procedure can be run in $O(n\log n)$ using a graph search and an efficient data structure [2].
d. Once the chain has been found, find the shortest path between the two points in the chain. This procedure can be run in $O(n)$ using a funnel algorithm [4].

As described in [2] the resulting path can be non-optimal because the funnel algorithm is ran in a possibly non-optimal chain. Though non-optimal in global sense, the resulting path is still too close to the obstacles because it is optimal in the considered chain. In the next section we present a modification to the present procedure in order to obtain a safer path among the obstacles.

2.4.2 Modified CDT Procedure

The main idea is based on the properties of the incenters of two adjacent triangles. Let us consider two adjacent triangles and their respective incenters (red points in the next figure). Since the incenter is the intersection point of the bisectors of the angles of the triangle and since every angle is smaller than 180° we are sure that the line linking the two incenters crosses the adjacent edge of two triangles. Consider the following figure:

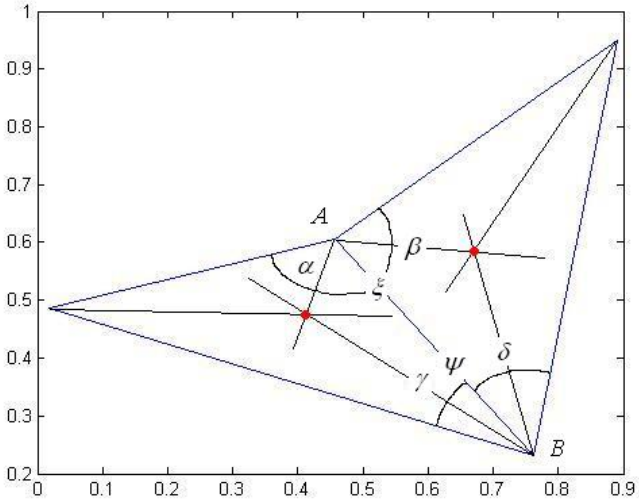

Figure 1: Computation of Incenters

The four angles $\alpha, \beta, \gamma, \delta$ are smaller than 180° each; hence the angles ξ, ψ are convex. This implies that the resulting quadrilateral is convex and hence the line linking the two incenters crosses *AB*.

The above properties are very useful in finding an obstacle-free path. Once we have defined the triangles using the CDT we know a set of points, which defines an admissible path. The procedure we propose uses the first three steps of Kallmann algorithm, while the last step is modified as follows:

Once the chain has been found, simply link the incenters of the adjacent triangles in order to obtain an admissible path and link the initial and the final point to their respective incenter.

The resulting path is surely non optimal but it has many advantages since it can be found in a smaller time than the Kallmann procedure because there is no need of the funnel algorithm; moreover it is more distant from the obstacles and therefore a safer path. This last property will be more clear with the following example.

Example 1 – The next figure shows a scenario with two obstacles, one vehicle (blue "x") and one target (green "o"). The optimal path between the two points is the green path while the red line is the union of the segments linking the incenters of the adjacent triangles.

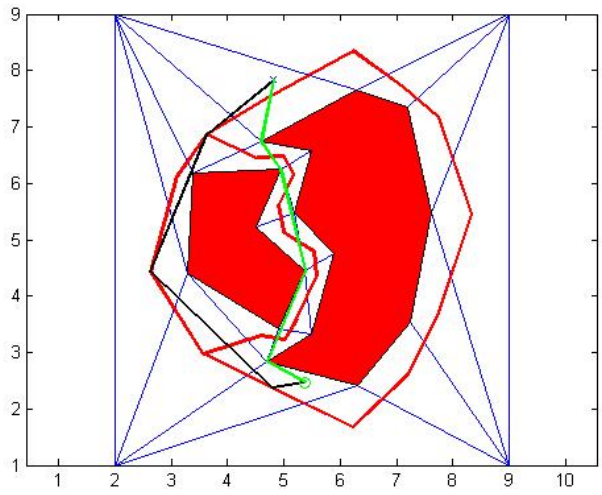

Figure 2: Path Computation for Example 1

The black path is the resulting path using a reduction procedure (see later). We note that the non-optimal path is far away from the optimal one (in this case the non-optimal path is about 27% greater than the optimal one) but it is worth to note that between the two obstacles the red path reveals safer than the optimal path because it is more distant from the edges of the obstacles.

We focus on the fact that the method of the incenters can be expanded noting that given a generic constrained triangulation of the plane and two points enclosed in two adjacent triangles, if the segment linking them crosses the adjacent edges, that segment is surely obstacle-free. This property is easily proven noting that the triangles are always convex polygons, and the segment can be divided in two inside one triangle each and then the resulting segment is completely inside the union of the two triangles. This implies that it does not cross any obstacle.

This last property can be used to reduce a path. Consider a path passing through many points inside the triangles of the CDT each, and try to delete one point (except the first and the last) in order to obtain a shorter path. If the new segment crosses all the adjacent edges of the triangles then it is an admissible path. More precisely:

a. Start from the first point of the obtained path and try to link directly to the fourth point of the path. If this segment intersects the two adjacent edges of the three triangles then the segment is surely admissible, otherwise it is non-admissible.
b. Indicate with P_i a generic node of the path, with T_i the triangle enclosing P_i, with S_i the adjacent edge between T_i and T_{i+1}.
c. Try to link P_i with P_{i+3}. If this segment intersects S_i, S_{i+1}, S_{i+2}, then this is an admissible one and we can delete P_{i+1} and P_{i+2} from the path. Otherwise try to link P_i with P_{i+2}, if this segment intersects S_i and S_{i+1} this is an admissible one and we can delete P_{i+1} from the path, otherwise it is a non-admissible segment.
d. Repeat step c until all nodes are visited.

Using this reduction procedure, the resulting paths are straighter and shorter, though it is necessary to run a simulation whose worst-case computational cost is $O(v)$ where v is the number of nodes in the initial path. We point out that both Kallmann procedure and our procedure are scalable with the number of vehicles and targets because the CDT depends on the obstacles only while the vehicles and the targets are considered in a second moment.

2.5 Concluding Remarks on Path-Planning

We have analysed path-planning procedures based on visibility graph, fixed tessellations, Euclidean path-planning and CDT-based path planning. Another common approach is to use mixed integer linear programming algorithms (MILP); this procedure is described in a later section, because it can be used to solve mission planning problems as well.

We have shown that the visibility graph approach leads to optimal trajectories but its computational cost is too large for real-time implementation. Fixed tessellations-based path-planning has a smaller

computational time, especially for large scenario but it is still too large in the view of real-time applications. Euclidean shortest path [2] is a fast procedure to find the optimal path between two points in the plane in presence of obstacles but it has the disadvantage that optimal paths are not necessarily safe in the view of real-time applications because they may be too near the obstacles (in theory they lie on the edges of the obstacles as much as necessary and possible). Kallmann procedure is based on the dynamic CDT and it has a low computational cost. However it has the same problem of the Euclidean shortest path because, though non optimal, the resulting path is still lying on the edges of the obstacles.

Our proposed method is a modification of the Kallmann procedure, and although it produces a non-optimal path, has a low computational cost and it allows safer trajectories especially when obstacles are close each other.

3. Dangerous Zones

In this section we deal with the presence of dangerous zones in the scenario, and how their presence may affect path planning procedures. These zones are in principle flyable but they may include a variety of risks for safe flight (environmental, adversary, etc.). We can include these zones by modifying the path cost by a factor depending on the probability of unsafe flight. To this end we refer to the search of the shortest path on a graph. We identify with E_{ij} the edge linking the i^{th} and the j^{th} nodes. Next we introduce the quantities P_{ij} for every edge such that:

$$0 \leq P_{ij} \leq 1$$

where $P_{ij} = 0$ means that the path is *safe* while $P_{ij} = 1$ means that if a vehicle travels along E_{ij} then it will be damaged.

The main question is the following: "*If an UAV travels along a path what is the total probability of being damaged ?*". Obviously the answer to this question depends on the dangerousness of each

edge of the path. Let us denote with P_P the probability of being damaged on the entire path; then we have the following relationship:

$$P_P = 1 - \prod_{k=1}^{L} (1 - P_k)$$

where L is the number of edges of the path and P_k is the probability of being damaged of the k^{th} edge of the path. It is trivial to note that, in order to obtain P_P, we can't directly sum all the P_k so we need another way to establish the total generalized length (total cost, W_P) of the path.

The first idea to deal with P_P is to consider it in the total cost in the following way:

$$W_P = \sum_{k=1}^{L} c_k + c_{MAX} P_P$$

where c_k is the cost of the k^{th} edge and c_{MAX} is a constant that must be tuned. This method represents the best way to consider P_P because it is exact, but it has the disadvantage that it can't be run with the on-line algorithms previously introduced because we take care of the dangerousness only after the path-planning procedure is done (in fact we can compute the total effects of the P_k only when we have the entire sequence of visits). Our goal is instead to deal with P_k during the path-planning procedure and so we must find another way to include P_P.

Consider then a modified cost $\hat{c}_k$ of every edge as follows:

$$\hat{c}_k = c_k + c_{MAX} f(P_k)$$

where f is a generic function of P_k. If a path contains only two edges E_1, E_2 then the total cost W_P will be:

$$W_P = \hat{c}_1 + \hat{c}_2 = c_1 + c_2 + [f(P_1) + f(P_2)] c_{MAX}$$

The goal is to find f such that the term into square brackets resembles P_P. Choose:

$$f(P_k) = \log\left(\frac{a}{1 - P_k}\right)$$

(where a is a parameter that can be tuned) which produces:

$$W_P = c_1 + c_2 + \left[\log\left(\frac{a}{1-P_1}\right) + \log\left(\frac{a}{1-P_2}\right) \right] c_{MAX} = c_1 + c_2 + c_{MAX} \log\left(\frac{a^2}{(1-P_1)(1-P_2)}\right)$$

we obtain:

$$W_P = c_1 + c_2 + c_{MAX} \left[2\log(a) - \log((1-P_1)(1-P_2)) \right]$$

Obviously log(a) is constant, and the second term in the square brackets is non-positive. We point out that, as we expected, if $P_1 = 0$, $P_2 = 0$ and $a = 1$ the modifying term is zero, which is the case of the non-dangerous paths. Moreover if an edge is dangerous ($P_k \rightarrow 1$) the modifying term is very effective in penalizing the respective edge.

Using the previous form for f we can update the procedure of finding the optimal assignment by simply summing $c_{MAX} f(P_k)$ to the cost of every edge. If a path contains L edges the total cost is given by the following expression:

$$W_P = \sum_{k=1}^{L} \hat{c}_k = \sum_{k=1}^{L} c_k + c_{MAX} \left[L\log(a) - \log\left(\prod_{k=1}^{L} (1-P_k) \right) \right] = \sum_{k=1}^{L} c_k + c_{MAX} \left[L\log(a) - \log(1-P_P) \right]$$

It is easy to see that if $P_P \rightarrow 1$ then W_P will be very large (depending on the value of c_{MAX}), while if $P_P \rightarrow 0$ then W_P will be given by the sum of the length of the edges (with $a = 1$).

Using this form of f allows we can include the dangerousness of an edge during the path-planning procedure. Moreover the total dangerousness of the path is directly calculated step by step and so there is no need to come back after the path-planning procedure in order to evaluate P_P.

Finally, we briefly show the effect of the parameter a. If $a = 1$ it does not affect the total cost. Only if $a \neq 1$, W_P is affected. We can use different values of a in the first step of the assignment procedure when we search the optimal shortest path between every vehicle to every target. We can set different values of a for the vehicles in order to deal with the differences between the vehicles themselves. For instance if a vehicle is more performing than the others then it may be favoured to be chosen to perform a task, so we can set a lower value of a for it. The operation of varying a at

the first step of the assignment procedure is not computationally expensive because we only need to sum the current value of log(a) to the cost of every edge. As an example, if a target T_1 was to be attacked and the vehicle U has not weapons, we can set a very large value of a when we search the optimal path between U and T. A different value of a for the same UAV can be set if a target T_2 requires a visit and not an attack.

We have proposed a procedure to deal with the dangerousness of some zones of the scenario. The exact effects of the probabilities of a vehicle being damaged along a specific segment of a path can be considered only at the end of the path-planning procedure and so it is unfeasible. We have therefore modified the cost of every edge with a term depending on the probability of damage in such a way that we can directly sum the modified costs in order to obtain the total cost of the path. This modification allows using one of the procedures shown before and, in the particular case of absence of dangerous zones, we recover the same paths of the unmodified costs. Moreover, we have an additional degree of freedom for dealing with the difference between vehicles.

4. Mission Planning

In this section we concentrate on the mission planning problem for a team of n UAVs. Many different situations may arise in this context: there can be targets that must be visited only, targets that, once reached, require a particular task such as mapping, identification, attack, verification, and so on. Moreover, some of the areas flown may have different levels of danger for the vehicle. Finally, we may have to consider vehicles with different characteristics, sensor suites, and capabilities.

4.1 Optimal Assignment

The objective of this section is to find a strategy of task assignment such that a team-cost is minimized. It is well known that finding the optimal solution to the assignment problem is a very hard problem in a general sense, because it involves the exact solution of the Travelling Salesman Problem (TSP), which is known to be NP-hard. Moreover the simple TSP involves one agent that must visit *m* cities, while in our case there are *n* agents, who must visit *m* cities minimizing a team-cost, hence the problem is much more complex. In order to find the optimal solution in the general case one must enumerate all the possible assignments and choose the cheapest one. Unfortunately the total number *N* of assignments is given by the following:

$$N = \frac{(n+m-1)!}{n!}$$

and then it is unfeasible even for small values of *n* and *m*.

Recently Rasmussen *et al*. [5] presented a way to compare heuristic solutions to the problem of task assignment. Obviously, in the case of very small values of *n* and *m* these strategies can be compared with the optimal one but for large values of *n* and *m*, heuristic procedures must be compared each other without even knowing the optimal solution.

4.2 Hungarian Algorithm

The Hungarian algorithm (Munkres' algorithm) is a fast procedure for determining a sub-optimal assignment. Define the matrix *C* of size *n* by *m* where the (*i*, *j*) entry represents the cost that the *i*-th vehicle have to go to the *j*-th target. The Hungarian algorithm finds the assignment that minimizes the sum of the cost of all the vehicles. More precisely it minimizes the following cost index

$$J = \sum_{i=1}^{n}\sum_{j=1}^{m} c_{i,j}\, p_{i,j}$$

where p_{ij} are the entries of a permutation matrix. The Hungarian algorithm works correctly for square matrices. In [6] we can find some modifications to the numerical procedure, to account for

the case of non-square matrices. The solution is sub-optimal in the sense that does not take care of the fact that after a vehicle has visited his target, it can be reassigned to another target and hence there is the possibility that a target receives multiple assignments for the same task. This may lead to unwanted behavior because a vehicle could move to a target and then to another one without visiting neither of them. This is obviously not desirable.

The main reason for the above sub-optimality of the Hungarian algorithm is that it performs a one-step optimization that is the obtained solution is optimal only if a vehicle stops its mission after completing the first task. In addition, if there are more tasks than vehicles, fictitious vehicles must be added in order to correctly run the algorithm.

4.3 MILP Approach

In this sub-section we briefly review a MILP (Mixed Integer Linear Programming) approach to the solution of both path-planning and mission planning. This method has the property of producing the optimal solution of both problems taking care of the vehicles' capabilities at the same time but, we will see, has the great disadvantages of a very large computational cost.

The path-planning and mission planning problem can be formulated as follow. Define a cost index J_T which must be minimized of the form

$$J_T = \sum_{k=1}^{m}\left(\sum_{i=1}^{N-1} q_k^{'}\left|s_{i,k}\right| + \sum_{i=0}^{N-1} r_k^{'}\left|u_{i,k}\right| + f\left(s_{N,k}\right)\right)$$

where p is the number of vehicles, $s_{i,k}$ are the state-space variables of the vehicles, $u_{i,k}$ are the input variables, q_k r_k are two cost vectors, $f(s_{N,k})$ are functions of the final states, N is the number of time steps. Define constraints in the following form:

$$Ax \le b$$

where the vector x contains all the $s_{i,k}$ and $u_{i,k}$. In order to take care of obstacle avoidance, collision avoidance, and target assignment many binary variables must be added, which enlarge the size of x.

The resulting optimization problem involves both real and binary variables, which can be solved using one of the many available solvers. The solution is the optimal to both the path planning and mission planning problems. This result stems from the fact that every problem we may have (such as shortest obstacle-free path, optimal assignment, minimal completion time, etc…) can be formulated as MILP. We note that, in order to run the MILP procedure we must define a sample time and the time horizon over which the optimization is run. Hence decreasing the sample time or increasing the time horizon leads to larger systems to be solved. For scenarios where there are many UAVs and many targets, the entire procedure becomes very computationally expensive. Moreover the unknown dynamics of the scenario (discussed in previous sections) combined with the high computational cost prevent the MILP to be applied in real-time applications. More details on MILP procedure can be found in [7].

4.4 Clustering approach

The exhaustive search of optimal assignment and the MILP approach are capable of finding the best solution to the problem in the view of minimization of a particular index, but these are both too computationally expensive and non-scalable. The Hungarian algorithm has instead a low computational cost, but the resulting assignment is only local optimal and global performance can be poor. In this section we present a new methodology for dealing with many vehicles and many targets in order to obtain a fast and less myopic assignment.

The main idea is based on the assumption that close targets are likely to be visited only by one vehicle rather than more vehicles. This assumption allows the introduction of targets clustering in order to obtain a lower order system, and to take advantage of the proximity of some targets. We initially deal with obstacle-free scenarios, and a single task (target visit), to show the capabilities of the proposed procedure. Consider the following figure where the blue circles are the targets:

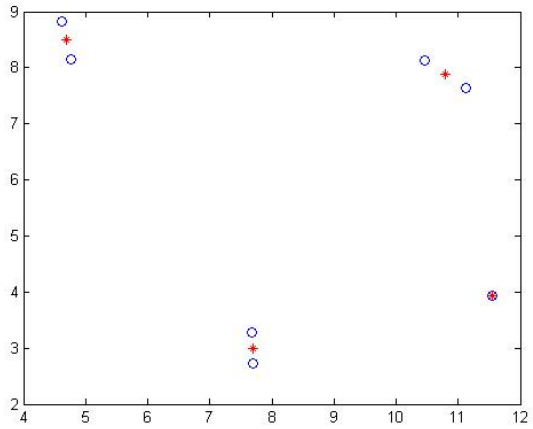

Figure 3: Initial Scenario for Clustering Approach Method

From the figure, many targets are close and then if a vehicle visits one of them it will probably visit the other. Hence, in this example, we can define four clusters which are marked as red stars.

The case of scenarios with obstacles is more complex because "close" targets can be very distant in terms of flyable trajectory. Consider the following figure:

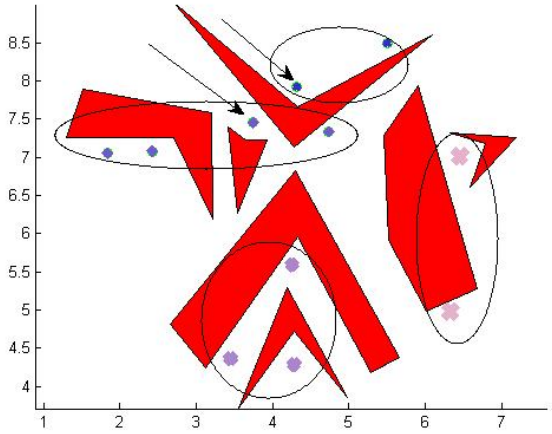

Figure 4: Initial Scenario with Obstacles for Clustering Approach Method

It is clear that the targets we have indicated with the two arrows are very near (the ideal straight line linking them is very short) but, due to the presence of the obstacles, these can not be clustered and perhaps a single vehicle is not sufficient.

The procedure to find clusters of targets uses the distances between the targets as pseudo-coordinates in a *m*-dimensional space. Initially we find the shortest path (or an admissible one, as

discussed before) linking every target to each other. In doing this we run the path-planning procedure only $m*(m-1)/2$ times because of the symmetry in going from target i to target j and vice versa. Once the paths are found we build the cost matrix (which is symmetric) with each path length, and we use its columns as the pseudo-coordinates of the targets.

After defining a desired number of clusters k we run a clustering algorithm (*k-means* or fuzzy *c-means* algorithm), which groups the targets in k clusters. In figure 4 the clusters were obtained using $k = 4$. The targets belonging to the same cluster are represented with the same symbol and the entire cluster is enclosed into an ellipse. It is worth noting that the clustering algorithm recognized correctly that the two targets we indicated in the figure with two arrows are far away in the scenario and they were assigned to different clusters.

The assignment is devised by finding the mean distance between every vehicle to the targets of every cluster and then by building a new cost matrix on which we can run an assignment algorithm such as the Hungarian technique. Once a vehicle has been assigned to a cluster it must optimise its path. This optimization step is the standard TSP, which can be solved approximately by heuristic procedures.

An open question to the approach suggested above is "Which is the right number *k*?". We must find a way to rigorously determine the optimal value of k. This is needed because there must be a compromise between a choice of too many clusters (with too few vehicles), and the opposite situation. The problem is well known in literature ([1], [8]) and we can find many approaches to solve it.

In the view of dynamic environment we must take care of the mobility of both targets and obstacles and so we need a dynamic procedure. We propose the following clustering procedure.

a. Fix a real positive r and choose a norm.
b. Start with only one cluster with centroid placed at the center of the m points.
c. Find the farthest point (in the sense of the chosen norm) named P_f from the center of each cluster and check if it is farther than r. If it is not then stop. Else go to step d.

d. Add a new cluster centred in P_f and run the *k*-means algorithm using as initial guess the available centres. Go to step c.

It is clear that the value of *r* affects very much the clustering result: choosing a large value will result in a larger amount of targets in every cluster. Setting a small value, results in a smaller number of targets in every cluster. If $r = 0$ there are as much clusters as targets since every cluster centre coincides with a target. The parameter *r* can be seen as a proximity factor and it can be tuned in order to obtain different clustering. The value of *r* may depend on the vehicles velocity.

The procedure we have presented is similar to the one described in [9], where the so-called *dispersion measure* is replaced by a norm. Moreover in our approach there is no "optimal" number of clusters, but the final number depends on the value of *r*.

In order to avoid sudden changes in clustering after a target has been visited, we weight each target with a value w_i such that $w_i \in [0 \quad 1]$; w_i must smoothly decrease from one to zero whenever the *i-th* target has been visited. In this way, the center of the respective cluster smoothly moves and churning effects can be avoided. We point out that by using the pseudo-coordinates we can deal with the effects of dangerous zones as well. This is due to the fact that the length of the shortest path we calculate with a path-planning algorithm becomes a more general path-cost summing up the effects of the dangerousness of certain zones.

5. Conclusions

Path planning and mission planning issues in cooperative control have been studied. Several known approaches for solving those problems were presented focusing on the advantages and the disadvantages of each. In the context of path planning we have shown that optimal algorithms produce paths that are not much applicable to real-time situations because the trajectories are too near to the obstacles. We proposed a dynamic procedure partially following the one proposed by Kallmann and based on the DCDT. Our procedure has a low computational cost and produces safer

paths in terms of danger to the integrity of the vehicles. Dangerous zones are taken into account by simply summing a term depending on the risk encountered in flying through a path. In the context of mission planning we proposed a dynamic clustering approach in order to obtain a less myopic and more scalable assignment procedure. Using this procedure the UAVs are assigned to a cluster instead of a single target and then each vehicle optimizes its path by approximately solving a TSP.

6. References

[1] J. Hershberger, S. Suri. An Optimal Algorithm For Euclidean Shortest Paths in the Plane. In *SIAM Journal on Computing* vol. 28, No. 6, pp. 2215-2256, 1999.

[2] M. Kallmann. Path Planning in Triangulations. In *Proceedings of the Workshop on Reasoning, Representation, and Learning in Computer Games*, International Joint Conference on Artificial Intelligence, Edinburgh, Scotland, July 31, 2005, 49-54.

[3] T. Schouwenaars, B. De Moor, E. Feron, J. How. Mixed Integer programming For Multi-Vehicle Path Planning. European Control Conference, Porto, Portugal, September 2001, pp. 2603-2608.

[4] B. Chazelle. A Theorem on Polygon Cutting with Applications. In Proceedings of the 23rd IEEE Symposium on Foundations of Computer Science, 339-349, 1892.

[5] S. Rasmussen, P. Chandler, J. Mitchell, C. Schumacher, A. Sparks. Optimal vs Heuristic Assignment of Cooperative Autonomous Unmanned Air Vehicles. In *Proceedings of AIAA Guidance Navigation and Control Conference and Exhibit*. 11-14 August 2003, Austin, Texas.

[6] D. Turra, L. Pollini, M. Innocenti. Fast Unmanned Vehicles Task Allocation With Moving Targets. In Proceedings of the 43rd IEEE Conference on Decision and Control, December 14-17, 2004 Atlantis, Paradise Island, Bahamas.

[7] T. Schouwenaars, B. De Moor, E. Feron, J. How. Mixed Integer programming For Multi-Vehicle Path Planning. European Control Conference, Porto, Portugal, September 2001, pp. 2603-2608.

[8] W. Lu, I. Traore. Determining the Optimal Number of Clusters Using a New Evolutionary Algorithm. In Proceedings of the 17th IEEE International Conference on Tools with Artificial Intelligence

[9] C. Chinrungrueng, C. H. Séquin. Optimal Adaptive k-Means Algorithm with Dynamic Adjustment of Learning Rate. In IEEE Transactions on Neural Networks, Vol. 6, No 1, January 1995.