TERRAIN FOLLOWING MINIMUM TIME TRAJECTORY DESIGN FOR UAV SWARMS USING MIXED INTEGER PROGRAMMING

J.W. Vervoorst, T. Kopfstedt
Diehl BGT Defence GmbH & Co. KG
Alte Nussdorfer Strasse 13, 88662 Ueberlingen
Germany

ABSTRACT

This publication discusses the calculation of time optimal mission scenarios for swarms of Unmanned Aerial Vehicles (UAV). A typical UAV mission consists of visiting a number of waypoints within a partially known environment and these tasks can be distributed between all members of the UAV swarms so that the complete mission can be accomplished in minimum time. In this particular case, a multi-level planning algorithm is presented that generates 2D or 3D trajectories for each member of the UAV swarm. A receding horizon controller approach is chosen for the calculation of the time optimal trajectories, therefore enabling adaptation to changes in the mission scenario or unknown obstacles.

Index Terms: Mixed Integer Linear Programming, task assignment, visibility graphs, receding horizon control

1. INTRODUCTION

Unmanned Aerial Vehicles are already used in a variety of fields where the deployment of humans is considered too dangerous or unnecessary. For example, UAVs serve as airborne communication stations, reconnaissance aircraft, or even in tactical operations. But all models in use today are operated only as pilotless vehicles, still requiring several human operators on the ground. In the future, UAVs will be able to plan their flight missions autonomously, without human intervention. Additionally, they will cooperate with other UAVs in large groups, or swarms. The complexity associated with managing swarms of UAVs can no longer be handled effectively by human operators. Thus, planning and control algorithms have to be implemented that guarantee safe cooperation and distribute workload evenly between all members of the swarms, allowing for much faster mission completion.

The planning algorithm presented in this publication is able to generate full three dimensional terrain following minimum time trajectories for each member of a UAV swarm. The terrain following aspect of trajectory generation is especially important when trying to avoid detection by enemy radar, as is necessary in tactical or reconnaissance operations in hostile airspace. Low flying and terrain following trajectories usually do not constitute the minimum time solution of the optimization problem, hence the two optimization criteria are inherently opposite. Thus, weighting factors that govern the importance of each criterion can be adjusted dynamically during the UAV mission in order to address possible changes in the mission profile.

A typical UAV mission generally can be described as visiting a number of waypoints distributed somewhere on a map. These can be points of interest for reconnaissance or maybe target locations. A schematic of the described scenario is depticed below in Figure 1.

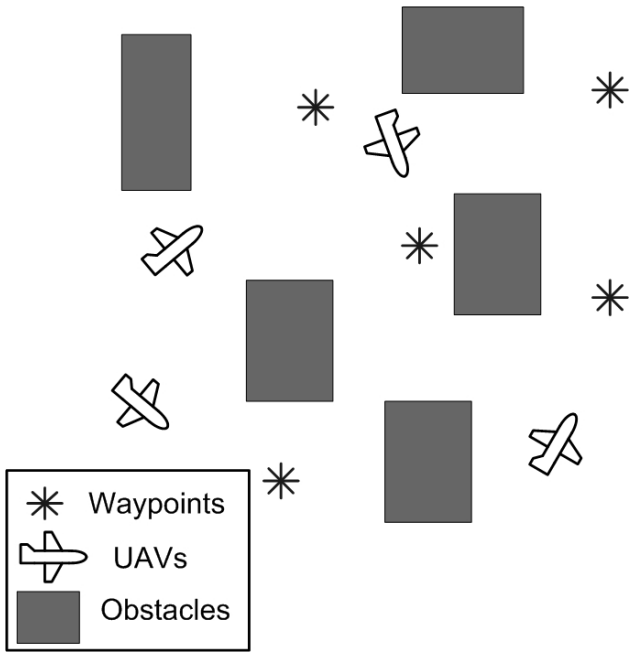

FIG 1. UAV mission scenario

The different levels of the complete mission planning algorithm are now explained in a little more detail.
In the first phase of the algorithm, the so called task assignment phase, tasks are distributed among all members of the UAV swarm. That means that each UAV receives an ordered list of waypoints that it has to visit during the duration of the mission. When all waypoints have been visited, the mission is completed. Waypoints are assigned to each UAV by means of a modified multidimensional traveling salesman problem (TSP). This problem is solved as a Mixed Integer Linear Programming (MILP) problem, as it is similarly outlined in [1].

In the next phase, the trajectory planning phase, detailed trajectories are generated for each UAV, also by MILP. In order to generate truly time-optimal trajectories, it is usually necessary to optimize the complete trajectory over a fixed horizon, from the starting point all the way to the end point. For long distances, this leads to a very complex optimization problem that cannot be solved in real time. Furthermore, unknown terrain or other changes in the mission along the way are not accounted for, and a change in the environment will render the previously calculated optimal trajectories obsolete. Thus, a receding horizon controller is implemented that calculates near-

optimal trajectories iteratively, previously shown for the single vehicle case in [2]. It allows navigation in uncertain or unknown environments as well as adapting to changes in the mission profile. A detailed kinematically constrained trajectory is only generated within a small planning horizon around the UAVs, while the rest of the trajectory is approximated.

Since the mathematical description of obstacles is crucial for both the task assignment and the trajectory planning, we will discuss it first in chapter 2. In chapter 3, the task assignment algorithm is described and examples for both 2D and 3D cases are given. Chapter 4 includes all details about the trajectory planning part of the mission planning tool. Chapter 5 combines task assignment and trajectory planning into a complete mission planning algorithm. Simulation results for various examples are discussed in chapter 6.

2. TERRAIN/OBSTACLE REPRESENTATION

One of the appeals of using a Mixed Integer Linear Programming is the possibility to declare obstacles very efficiently. Obstacles of almost arbitrary shape can be described by declaring constraints on the optimization problem.

2.1. Obstacles in 2D

The general approach to the declaration of obstacles shall be explained in detail for a very simple obstacle shape, a rectangle in the x-y plane.

In this case, obstacles can be described simply by giving their lower left and upper right corner $[x_{ll}, y_{ll}, x_{ur}, y_{ur}]^T$. The variable b_{object} is a binary, and the number M is an arbitrary large positive value.

$$(1)\quad \begin{aligned} x_{jk} &\le (x_{ll}) + M \cdot b_{object_{ijk1}} \\ y_{jk} &\le (x_{ll}) + M \cdot b_{object_{ijk2}} \\ x_{jk} &\ge (x_{ur}) - M \cdot b_{object_{ijk3}} \\ y_{jk} &\ge (x_{ur}) - M \cdot b_{object_{ijk4}} \end{aligned}$$

$$(2)\quad \sum_{l=1}^{4} b_{object_{ijkl}} \le 3$$

$$i = 1...N_O,\ j = 1...N_V,\ k = 1...T,\ l = 1...4$$

$[x_{jk}, y_{jk}]^T$ is the position of UAV j at time k. Index i describes the number of obstacles, j lists the number of UAVs in the swarm, k the number of time steps in the planning horizon, and l enumerates the four inequalities. As long as $\sum b_{object_{ijkl}} \le 3$ is fulfilled, the UAV is outside the obstacle.

2.2. Obstacles in 3D

By simply adding constraints for the z coordinate, rectangles can easily be modified into three dimensional obstacles.

$$(3)\quad \begin{aligned} z_{jk} &\le (z_{ll}) + M \cdot b_{object_{ijk5}} \\ z_{jk} &\ge (z_{ur}) - M \cdot b_{object_{ijk6}} \end{aligned}$$

$$(4)\quad \sum_{l=1}^{6} b_{object_{ijkl}} \le 5$$

The total number of constraints has risen to six and we now have a cuboid.

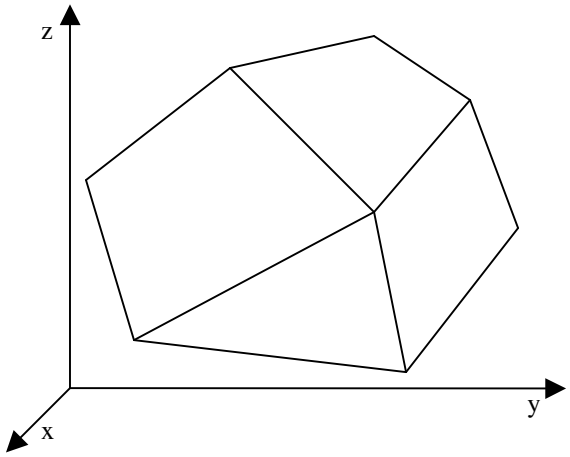

FIG 2. Polygon 3D obstacle

Similarly, we can describe any arbitrarily shaped convex obstacle with a number of constraint equations as stated above. The volume of the polygon can be described by

$$(5)\quad \vec{A} \cdot \begin{bmatrix} x \\ y \\ z \end{bmatrix} + \vec{b} \le \vec{0},$$

where $\vec{A}$ and $\vec{b}$ are linearly independent. The dimensions of $\vec{A}$ and $\vec{b}$ depend on the number of constraints necessary to describe the bounding surface of the obstacle.

3. TASK ASSIGNMENT

As already mentioned, the Task Assignment problem is very similar to the Traveling Salesman Problem (TSP). Just as in a TSP, there is a list of waypoints that need to be visited once. However, in this case there is not one single person, but rather there is a swarm of UAVs that split the task between them. Such a multi-dimensional TSP is explained in [10]. A similar description of the task assignment problem, applied to airplanes, can be found in [11] and [12].

3.1. Problem Description

In the case of the regular TSP, all of the waypoints have been assigned certain cost values. What is the meaning of these cost values? Typically, they describe the distances or travel times between waypoints. Therefore, in our case the cost values of the waypoints would be the flight times between waypoints. For simplicity, the distances between waypoints are approximated by straight line segments.

In the beginning we have a group of N_V UAVs, their

respective initial positions $\vec{x}_{init}$, as well as their maximum velocities $v_{\max}$. Furthermore, there's a number N_W of waypoints whose positions are known. Waypoint coordinates are combined in the matrix $\vec{W}$. There's also a number of N_O obstacles. Obstacle descriptions are also stored, depending on how the obstacles are defined (see chapter 2).

3.2. Visibility Graphs

A decisive problem for the path planning problem in mobile robotics is the detection of visibility. The standard robotics problem is the navigation of a system through an environment filled with obstacles. In our case this is one of several UAVs that is trying to find a free, unobstructed path to a waypoint.

So the goal should be to find the shortest route to the target despite the obstacles. In order to achieve this, we employ so called visibility graphs. A visibility graphs maps all possible straight line connections between points that are visible to each other, meaning that no obstacle is in the path of the connecting line segment. While there are several other methods of finding these connections, as outlined in [13] and [14], but visibility graphs are widely used and very efficient ([15]). Example visibility graphs for both 2D and 3D are shown in Figures 3 a) and b).

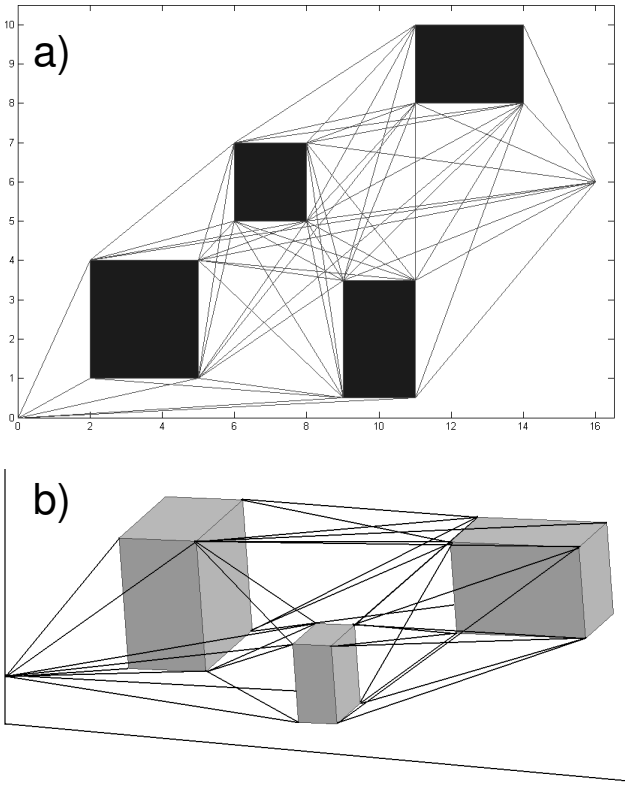

FIG 3. a) Example of a visibility graph in 2D
b) Visibility graph in 3D

3.3. Algorithm

The visibility graph finds all connections between waypoints, but not the shortest connections. In order to achieve this, the following algorithm is used:

- Calculation of the visibility graphs between all waypoints, the obstacles and the initial positions of the UAVs and storing all data in a cost table.
- Using the Dijkstra algorithm, the shortest distances between all waypoints are calculated from all combinations and stored in another table.
- All possible waypoint combinations per UAV are calculated and the flight distances are added up for all ordered permutations.
- The best permutation of each combination is chosen.
- Data is passed to the optimization to solve for the minimum time solution to the task assignment problem.

Data is passed on to the optimization through four matrices/vectors:

- The row vector $\vec{u}$ with size N_V defines what UAV is assigned what permutation
- The time matrix $\vec{T}$ with dimension $N_W x N_{perm}$ stores the times at which a waypoint is visited in a certain permutation.
- The visiting matrix $\vec{V}$, also with dimension $N_W x N_{perm}$, shows wether a waypoint is visited at all during a certain permutation.

3.4. Optimization

Goal of the optimization is to find such waypoint combinations that:

- all waypoints are visited once
- the whole mission is accomplished in a minimum amount of time

3.4.1. Choosing the Permutations

A binary decision vector $\vec{b}$ is introduced to allow choosing the correct permutations. $\vec{b}$ is of dimension $N_{perm} x 1$. $\vec{b}_i$ is 1 when permutation i is chosen and 0 when it isn't.

As already mentioned, every waypoint should be visited once, and also every UAV should only be assigned one permutation. This can be expressed as follows:

$$\sum_{j=1}^{N_{perm}} V_{ij} \cdot b_j = 1, \qquad i = 1,...,N_W \tag{6}$$

Equation (6) makes sure that when summing over every permutation, every waypoint is only chosen once

$$\sum_{j=N_p}^{N_{p+1}-1} b_j = 1, \qquad p = 1,...,N_V \tag{7}$$

The parameter N_p is used to enumerate the permutations. Permutations assigned to the p-th UAV are numbered from N_p to $N_{p+1}-1$. Thus, Equation (7) allows only one single permutation to be chosen for each UAV.

3.4.2. Objective Function

Formulation of the objective function has a large impact on the solution of the optimization. In the case described here, we want to visit all waypoints in minimum time. The

objective function for this is defined as:

$$(8)\quad Z = \left(\max_{i \in (1,\ldots,N_V)} t_{FD_i} \right) + \frac{\alpha}{N_V} \cdot \sum_{i=1}^{N_{perm}} c_i \cdot b_i$$

t_{FD_i} is the total flight duration for UAV i. The maximum total flight duration is therefore the time it takes for the longest permutation of the mission to be accomplished. That means that t_{FD_i} is also the overall mission time.

The second term is the averaged sum over the flight times of all UAVs multiplied by a weighting factor. We not only want the longest flight time to be minimal, we also want the other UAVs to reach their target as quickly as possible.
A task assignment solution for four UAVs and eight waypoints is shown in Figure 4.

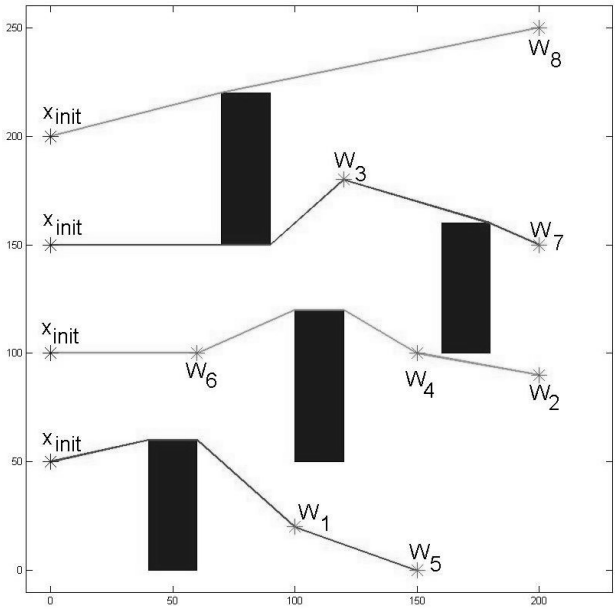

FIG 4. Task Assignment solution

4. TRAJECTORY PLANNING

In the previous chapter, the optimal mission scenario for a swarm of UAVs was found, with our main focus on calculating the optimal task assignment. The flight trajectories between waypoints were only approximated by straight line segments since a design of detailed flyable trajectories would have been far to complex at this point. Instead, detailed trajectory design is handled in this chapter. Using a discrete state space model for the UAV, flyable trajectories are generated with respect to velocity and acceleration constraints and obstacle and collision avoidance. Just like in the task assignment phase, Mixed Integer Linear Programming is used in the optimization of this planning problem. A basic example of a similar problem can be found in [3] and [4].

4.1. System Description

Detailed models of aircraft dynamics are typically strictly nonlinear and therefore trajectory optimization for these systems tends to be rather complex [5]. A basic, yet adequate model that can be used for the modelling of a UAV is a simple point mass. By using additional constraints on speed and turning radius, the behavior of an aircraft can be approximated. The discrete time state space representation for such a system is:

$$(9)\quad \begin{bmatrix} x \\ y \\ z \\ v_x \\ v_y \\ v_z \end{bmatrix}_{k+1} = \begin{bmatrix} 1 & 0 & 0 & \Delta t & 0 & 0 \\ 0 & 1 & 0 & 0 & \Delta t & 0 \\ 0 & 0 & 1 & 0 & 0 & \Delta t \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} x \\ y \\ z \\ v_x \\ v_y \\ v_z \end{bmatrix}_k + \begin{bmatrix} (\Delta t)^2/2 & 0 & 0 \\ 0 & (\Delta t)^2/2 & 0 \\ 0 & 0 & (\Delta t)^2/2 \\ \Delta t & 0 & 0 \\ 0 & \Delta t & 0 \\ 0 & 0 & \Delta t \end{bmatrix} \cdot \begin{bmatrix} a_x \\ a_y \\ a_z \end{bmatrix}_k$$

4.2. Dynamic Constraints

As described above, additional constraints are added to the system to make it behave like an airplane. These constraints are defined as follows.

4.2.1. Maximum Velocity

A realistic airplane certainly has a maximum velocity of flight, or rather a desired cruising speed at which it should fly. Such behavior can be accomplished by adding a constrain on the maximum possible velocity. Therefore, the following has to be true at all times: $|\vec{v}| \le v_{\max}$

Mixed Integer Linear Programming, as the name already implies, is limited to linear systems. However, taking the length of a vector is an inherently nonlinear operation, so another way of expressing maximum velocity has to be found. Instead, the velocity is mapped onto a finite number of unit vectors that are evenly spread out in space.

$$(10)\quad v^T \cdot i_{k,l} \le v_{\max} \qquad k,l = 1,\ldots,n_{v_{\max}}$$

$$(11)\quad i_{k,l} = \begin{bmatrix} \cos\left(\frac{2\pi k}{n_{v_{\max}}}\right) \cdot \cos\left(\frac{2\pi l}{n_{v_{\max}}}\right) \\ \cos\left(\frac{2\pi k}{n_{v_{\max}}}\right) \cdot \sin\left(\frac{2lk}{n_{v_{\max}}}\right) \\ \sin\left(\frac{2\pi k}{n_{v_{\max}}}\right) \end{bmatrix}$$

Figure 5 shows this method for the planar, two dimensional case.

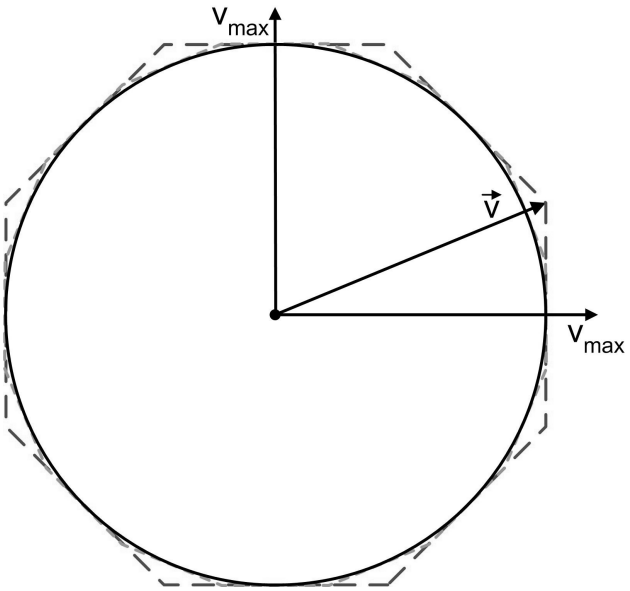

FIG 5. Mapping of velocity vector onto 8 and 16 unit vectors

4.2.2. Minimum Turning Radius

Analog to the description of maximum velocity constraints, the same setup can be used to limit the minimum turning radius of a UAV. In this case, we limit the maximum acceleration that the system can experience.

(12) $$a^T \cdot i_{k,l} \le a_{\max} \qquad k,l = 1,...,n_{a_{\max}}$$

(13) $$i_{k,l} = \begin{bmatrix} \cos\left(\frac{2\pi k}{n_{a_{\max}}}\right) \cdot \cos\left(\frac{2\pi l}{n_{a_{\max}}}\right) \\ \cos\left(\frac{2\pi k}{n_{a_{\max}}}\right) \cdot \sin\left(\frac{2lk}{n_{a_{\max}}}\right) \\ \sin\left(\frac{2\pi k}{n_{a_{\max}}}\right) \end{bmatrix}$$

4.3. Obstacle Avoidance

Obstacle avoidance constraints have already been discussed in chapter 2 when the general description of obstacles has been explained.

4.4. Trajectory Planning with Fixed Horizon

The constraints above can now be used to formulate a MILP optimization problem.

In this regard, trajectory planning with fixed horizon can be considered the traditional approach to finding time optimal trajectories. This means that the complete trajectory is calculated from beginning to end point, forming a large and complicated optimization that does not take into account changing environments. When obstacles are added or subtracted, the precalculated optimal solution essentially becomes worthless and a recalculation has to be performed.
It does however produce time optimal trajectories because the trajectory can be optimized as a whole, thus enabling the optimizer to find a ”closed loop solution”.

4.5. Trajectory Planning with Receding Horizon

As mentioned above, fixed horizon planning has its shortcomings especially in changing or unknown environments.

In order to achieve better results in this context, a different approach is chosen. Instead of optimizing the complete trajectory at once, a Model Predictive Control (MPC) setup is chosen. [6] lists the properties of Model Predictive Control, also called Receding Horizon Control:

- At time i and initial state $\vec{x}_i$ the optimization is performed for only the next N_P time steps. N_P is the so called planning horizon.
- The first N_E values of the optimal solution are used as inputs to the system. N_E is the execution horizon. N_E is usually set to 1.
- In the new initial state $\vec{x}_{i+N_E}$ the optimization is performed again, repeating the process, thus taking into account possible changes in the environment.

Similar implementations of MPC can also be found in [7] and [8].

To be able to use this approach, we need to overcome a problem. As seen in Figure 5, the trajectory is only optimized within a small area around the current state, the planning horizon. But the total trajectory consists of the optimized piece plus the remaining pieces from the planning horizon to the goal point. However, that piece is unknown and the total distance cannot be calculated. Therefore, the remaining trajectory is approximated with straight line segments connecting the planning horizon to the goal point.

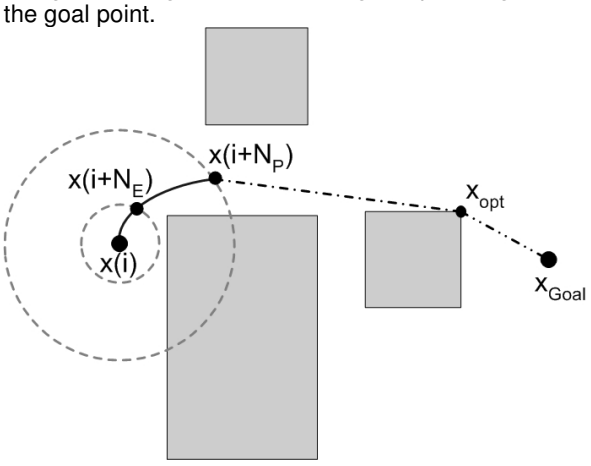

FIG 6. Details of Receding Horizon Control

4.5.1. Calculation of Cost Maps

In order to find the shortest distance to the goal point, a cost map of the complete environment is calculated, calculating and storing the distances of points to the goal point (Fig. 6). [6] shows that the shortest distance in an environment with convex obstacles is a combination of line segments between the points and the corners of the obstacles. Thus, the corners of the obstacles are cost

points in our cost map. The visibility graph between all obstacles and the goal point is calculated and by using the Dijkstra algorithm, as outlined in [9], the shortest connections between all the points and the goal point are calculated and stored as the cost of each point.

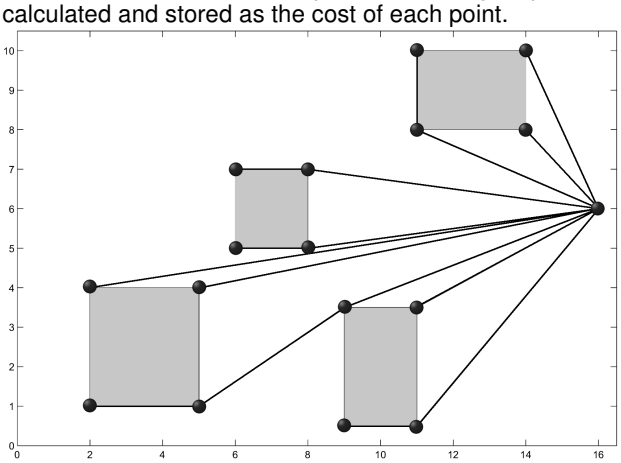

FIG 7. Calculation of a cost map in 2D

4.5.2. Cost Point Selection

The length of the trajectory outside the planning horizon can now be determined. As detailed in Fig. 1, it is:

- The distance from $\vec{x}_{i+N_P}$,the edge of the planning horizon, to a known cost point $\vec{x}_{opt}$ that is visible from that point.
- The distance from that cost point to the goal point. This value has already been calculated and is stored in the cost map.

The visibility constraint between $\vec{x}_{i+N_P}$ and $\vec{x}_{opt}$ is very important. Because of it, local optimization within the planning horizon does indeed have a global influence on the whole trajectory, therefore also minimizing the total trajectory length.

Now the planning problem is expressed in MILP form. Each UAV can select only one cost point during each optimization:

$$(14)\quad \sum_{i=1}^{N_{CP}} \sum_{j=1}^{N_{goal}} b_{CP_{ijk}} = 1 \ , \quad k = 1,...,N_V$$

with N_{CP} = number of cost points, N_V = number of UAVs, N_{goal} =2, the next two waypoints.

Equations (15) and (16) set the cost value and coordinates of the chosen cost point.

$$(15)\quad c_{opt_k} = \sum_{i=1}^{N_{CP}} \sum_{j=1}^{N_{goal}} c_i \cdot b_{CP_{ijk}} \ , \quad k = 1,...,N_V$$

$$(16)\quad \begin{bmatrix} x_{opt_k} \\ y_{opt_k} \\ z_{opt_k} \end{bmatrix} = \sum_{i=1}^{N_{CP}} \sum_{j=1}^{N_{goal}} \begin{bmatrix} x_{CP_k} \\ y_{CP_k} \\ z_{CP_k} \end{bmatrix} \cdot b_{CP_{ijk}} \ , \quad k = 1,...,N_V$$

The line connecting the edge of the planning horizon and the cost point is described by

$$(17)\quad \begin{bmatrix} x_{line_k} \\ y_{line_k} \\ z_{line_k} \end{bmatrix} = \begin{bmatrix} x_{opt_k} \\ y_{opt_k} \\ z_{opt_k} \end{bmatrix} - \begin{bmatrix} \left(x_{i+N_P}\right)_k \\ \left(y_{i+N_P}\right)_k \\ \left(z_{i+N_P}\right)_k \end{bmatrix}$$

In order to test visibility, this connection is divided into N_{test} parts:

$$(18)\quad \begin{bmatrix} x_{test_{km}} \\ y_{test_{km}} \\ z_{test_{km}} \end{bmatrix} = \begin{bmatrix} \left(x_{i+N_P}\right)_k \\ \left(y_{i+N_P}\right)_k \\ \left(z_{i+N_P}\right)_k \end{bmatrix} + \frac{m}{N_{test}} \cdot \begin{bmatrix} x_{line_k} \\ y_{line_k} \\ z_{line_k} \end{bmatrix} \quad m = 1...N_{test}$$

Each part is tested for interference with obstacles using a binary variable $b_{opt_{ikmn}}$, very much like in (2).

Equations (19) and (20) handle which goal point to use in the optimization. If a goal point is reached in the last step of the planning horizon or not at all, then only this point will be part of the optimization. If a goal point is reached within the planning horizon, then goal points are switched and the optimization directs the trajectory to the next goal point.

$$(19)\quad \sum_{i=1}^{N_{CP}} b_{CP_{i1k}} = \sum_{i=T}^{T+1} b_{goal_{ij}}$$

$$(20)\quad \sum_{i=1}^{N_{CP}} b_{CP_{i2k}} = \sum_{i=1}^{T-1} b_{goal_{ij}}$$

4.5.3. Cost Function

The cost function to be minimized is the total trajectory lengths of the UAVs, consisting of three parts:

- The part within the planning horizon, from $\vec{x}_i$ to $\vec{x}_{i+N_P}$
- The line between $\vec{x}_{i+N_P}$ and the selected cost point
- The distance between selected cost point and goal point.

The first part is described by the first part of Equation (21), which represents the number of time steps to the goal point times the distance traveled per time step.

$$(21)\quad Z = \sum_{j=1}^{N_V} \begin{pmatrix} \left(v_{j_{\max}} \cdot \Delta t\right) \cdot \sum_{k=1}^{T+1} k \cdot b_{goal_{kj}} + l_{line} \\ + c_{opt_j} + \sum_{m=1}^{T} \alpha \cdot z_{jm} \end{pmatrix}$$

l_{line} is the distance between $\vec{x}_{i+N_P}$ and the selected cost point. Since it constitutes the length of a vector, it is calculated just as in Equations (10) and (11).

The third part is simply the stored value of the cost point, meaning the distance to the goal point.

Additionally, the forth part of the objective function is added as a penalty term that can be used to influence UAV behavior. Large values of α will force the UAV to stay as low as possible during its mission, while simply setting α to zero will result in no penalty for changing altitudes.

5. SIMULATION RESULTS

First let's take a look at a general mission scenario. Figure 8 shows an example mission for 4 UAVs in a 2D environment with 4 obstacles. This example is also shown in Figure 4, where the task assignment solution for this particular UAV mission has been presented.

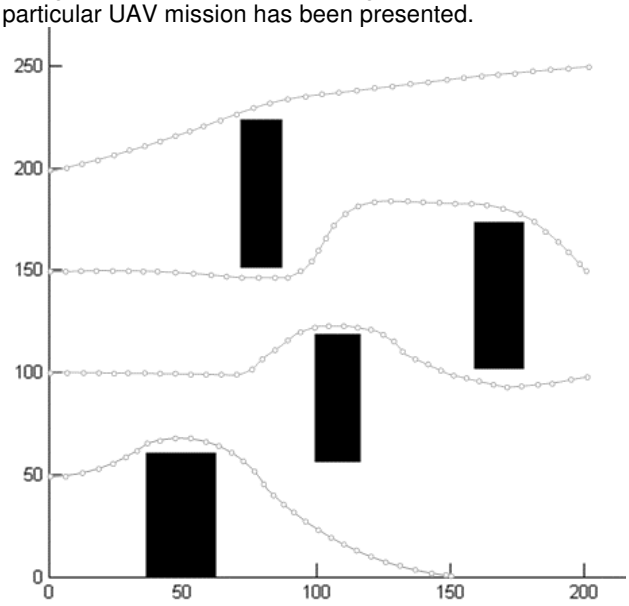

FIG 8. Mission example for UAV swarm

The optimal task assignment distributed the waypoints among all four UAVs in such a way that the individual flight times were almost identical, making for an ideal distribution of tasks.
Next we take a quick look at 3D trajectories. Two cases are being examined. In Figure 9a) the altitude penalty, seen in Equation (21), has been set to an extremely high level. Therefore, the optimization reaches a better value by planning the trajectory around the obstacle instead of over it.

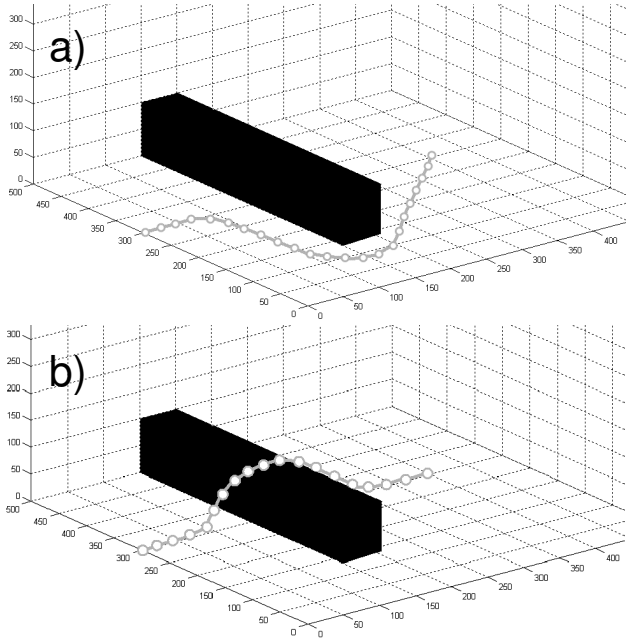

FIG 9. 3D trajectory behavior
a) with high penalty term for altitude
b) with low penalty term for altitude

In this case the primary concern of the UAV is to stay as low as possible, even if it means to have a longer flight path.
In Figure 9b) the altitude penalty has been set to a low level. Therefore altitude gain is not heavily penalized and the focus is on shortening the flight trajectory. The UAV will fly over the obstacle instead of around it.

6. CONCLUSION

This research paper gives a short overview about an approach to designing time optimal UAV missions by using Mixed Integer Linear Programming. The algorithm described here uses a form of receding horizon control to simplify the optimization problem and make it more capable of real-time calculations. Instead of planning the complete trajectory until the finish, only a small section is planned in detail iteratively and the rest is approximated by straight line segments. Therefore it foregoes the extremely complicated and time-consuming calculations typically encountered when calculating long trajectories with a regular fixed horizon approach.
Both the task assignment and trajectory planning algorithms are shown in detail with accompanying examples. Trajectories in both 2D and 3D perform as expected. By tuning the parameters of the objective function, the emphasis of UAV behaviour can be changed from pure minimum-time trajectories to primarily low altitude flight paths to avoid radar detection and the like.

REFERENCES

[1] C. Schumacher, P. Chandler, M. Pachter, "Constrained Optimization for UAV Task Assignment" AIAA Guidance, Navigation, and Control Conference, Providence, RI, USA, 16-19 August 2004
[2] A. Richards, J. How, "Model Predictive Control of Vehicle Maneuvers with Guaranteed Completion Time and Robust Feasibility", American Control Conference, Denver, CO, USA, 4-6 June 2003
[3] A. Chaudry, K. Misovec, R. D'Andrea, "Low Observability Path Planning for an Unmanned Air Vehicle Using Mixed Integer Linear Programming", Proceedings of the IEEE Conference on Decision and Control 2004. pp. 3823-3829
[4] M. Earl, R. D'Andrea, "Multi-Vehicle Cooperative Control Using Mixed Integer Linear Programming", IEEE Transactions on Robotics. Vol. 21, 2005, pp. 1158-1167
[5] R. Brockhaus, Flugregelung. Berlin. Springer-Verlag 2001
[6] J.-C. Latombe, "Robot Motion Planning". Boston, MA, USA: Kluwer Academic Publishers, 1991, pp. 153-200
[7] A. Richards, J. How, "Model Predictive Control of Vehicle Maneuvers with Guaranteed Completion Time and Robust Feasibility", American Control Conference, Denver, CO, USA, 4-6 June 2003
[8] E. Frew, "Receding Horizon Control Using Random Search for UAV Navigation with Passive, Non-cooperative Sensing", AIAA Guidance, Navigation, and Control Conference, San Francisco, CA, USA, August 2005
[9] T. Cormen, "Introduction to Algorithms". Cambridge, MA, USA: The MIT Press, 2001, pp. 595-601, 629-635

[10] M. Glocker, O. Stryk, „Gemischt ganzzahlig-kontinuierliche Optimalsteuerung: Methoden und Anwendungen“, DMV Jahrestagung. Heidelberg, September 2004
[11] D. Kingston, C. Schumacher: Time-Dependent Cooperative Assignment. American Control Conference, Portland, OR, 2005
[12] C. Schumacher P. Chandler, M. Pachter, “Constrained Optimization for UAV Task Assignment.”, AIAA Guidance, Navigation, and Control Conference, Providence, RI, August 2004
[13] M. Eichhorn, „Hindernisvermeidungssystem für ein Autonomes Unterwasserfahrzeug“, at, Vol. 11, pp. 514-525
[14] G. Dudek, M. Jenkin, “Computational Principles of Mobile Robotics”, Cambridge, UK: Cambridge University Press, 2000
[15] Baker, E.: Visibility Computation: Finding the Shortest Route for Motion Planning. Class Notes, University of North Carolina at Chapel Hill, Chapel Hill, NC, 2005