UKACC International Conference on Control 2012
Cardiff, UK, 3-5 September 2012

Towards a Fully Autonomous Swarm of Unmanned Aerial Vehicles

Jeremie Leonard
Cranfield University
Cranfield, MK430AL
Email: j.leonard@cranfield.ac.uk

Dr. Al Savvaris
Cranfield University
Cranfield, MK430AL
Email: a.savvaris@cranfield.ac.uk

Prof. Antonios Tsourdos
Cranfield University
Shrivenham, Swindon, SN6 8LA
Email: a.tsourdos@cranfield.ac.uk

***Abstract*—With advances in UAS technologies the quadrotor was given a special interest for its manoeuvrability and payload capacity. These assets are amplified when more of them are deployed simultaneously in order to improve the situational awareness over areas of interest. As the number of agents operating in the same environment grows, a common intelligence is needed to optimize their cooperation and ensure their safety throughout the completion of the missions. This paper presents the results of experiments conducted to demonstrate a set of algorithms on a surveillance system employing a swarm of quadrotor UAVs to track detected targets. It was initially assumed that the UAV paths are generated at constant altitude to replace the complicated quadrotor dynamics by ones of a point mass entity. The system is then extended to the third dimension to allow for a more complex guidance and navigation scheme. Several simulations were performed under various circumstances to validate the accuracy and robustness of the system.**

Keywords: Autonomous, swarm, UAV, quadrotor, control.

I. Introduction

Recent events have highlighted the ever-changing face of modern warfare and the need for strategically urban battle techniques to be developed and improved upon. The increase in asymmetric warfare shows that the current forces have particular weaknesses exploited by the enemy, due to an increased knowledge of the urban battlespace, putting lives in greater danger. With advances in UAV and UGV technologies together with the improvements in sensor capabilities there is a definite domain that allied forces can use to their advantage. Covert operations, disposable resources and getting closer to the enemy are all assets that these technologies carry on. Using multiple platforms to detect and inspect targets in regions of interest could provide precious information of the surrounding urban area, paving safer corridors, increasing the situational awareness. A thorough analysis of the data gathered by the swarm would help mapping safer and more effective missions. Ultimately, an on-board processing of these information would enable the system to function in real time applications to execute such recognition missions.

The state of the art in swarm behaviour has changed in the past few years and an increasing number of teams focus their research on formation flying and multi-agent control [1][2]. However, most of their work is based on off-the-shelf platforms modified to grab certain objects, land on a given surface or fly alongside its pears in an ideal environment. Resource optimisation, conflict resolution and failure diagnostic are still challenging fields of study to draw the need of human interaction further back.

II. Two dimensional simulation

A. The simplified model

Initially, the 3D environment is reduced to a two dimensional space and the 6 Degrees of Freedom (DoF) quadrotor is replaced by a 3 DoF entity. The simplified kinematics enables the testing of the following algorithms in real time and in more complex scenarios involving more units and several types of targets. In order to match the behaviour of its flying counterpart, the two dimensional model must be non-holonomic. For bench testing purposes, the point mass replacement was based on differential drive vehicle for which the user can easily control the rotation speeds ω_L and ω_R of the left and right wheels to direct the unit. This gives the unit's kinematics the advantage of being linear:

$$\begin{aligned} V_L &= R_{wheel}\,\omega_L \\ V_R &= R_{wheel}\,\omega_R \end{aligned} \tag{1}$$

$$\begin{aligned} V &= \tfrac{V_R+V_L}{2} \\ \omega &= \tfrac{V_R-V_L}{d} \end{aligned} \tag{2}$$

Thanks to this linearity, ω_L and ω_R can be monitored by a simple PI controller, easy to implement and computationally quick. The emphasis will be put on individual guidance and swarm cooperation.

B. Guidance and navigation

In mobile robotics, it is important that the environment in which the vehicles will be driving is eciently converted to a road map that the system can use to optimize the units movements. Even if the map is only partially known or uncertain, the conversion to a C-space representation needs to be simple enough to run in real-time, but still precise enough to guarantee the optimality of the calculated path. Looking forward a little bit, the main application of the system developed in this research would be a permanent surveillance setup in a dense urban environment. In this context, a cell decomposition coupled with an A* algorithm

978-1-4673-1558-6/12/$31.00 ©2012 IEEE

would guarantee a quick and easy way to find the shortest path between two points separated by several obstacles. Its computational lightness also makes it the most fitting solution to be implemented on the on-board processor. The A* algorithm is based on a step-by-step effort to minimize the path [3]. The idea is very intuitive: at each step of the way, it favours the solutions directly closer to the targets and leaves the other cells on the side. Those unfit solutions can be temporarily ignored but not deleted because there is no way to be sure that a path is going to be the right one. They are stored in a list of unexplored possibilities that the algorithm can search later on if the previously chosen path leads to a dead end. The path can then be cut down in the middle and restarted from a point previously left aside.

Developing an algorithm capable of finding the optimal path to a given point is essential to make the system intelligent. To make it autonomous, the quadrotor would then have to reach the goal point on its own by following the path allocated to it. Once again we want to implement on-board the algorithm in charge of guiding the quadrotor along its path. The pure pursuit (or carrot following) method answers the simplicity requirement and presents three additional assets.

- The guidance algorithm itself does not need to consider the dynamics of the vehicle.
- The method generates a virtual point, called carrot, on the path in front of the vehicle it wants to move which then has to determine the sequence of commands in speed and heading to stay at a look-ahead distance away from the virtual point while permanently aiming towards it (Fig.1). Since the point has no reality it can be created, deleted or moved with ease and the quadrotor will always follow. An appropriate control of the carrots behaviour could prove to be very beneficial for the upcoming conflict detection and resolution in 3D.

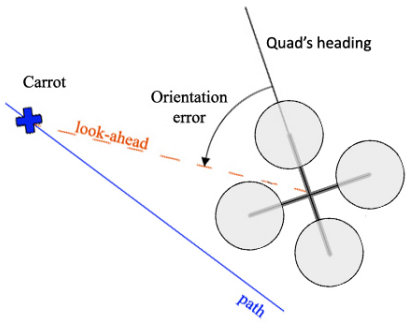

Fig. 1. Pure pursuit principle

- Though the A* generated path is made of straight segments, tracking the carrot leads the vehicle to cut the corner resulting in a much smoother trajectory and an even shorter trajectory. To avoid bumping into one of the obstacles while cutting a corner, the map of the environment is pre-processed to widen every obstacle. The path generated by the A* will therefore have a bigger clearance distance with the obstacles in the environment to ensure the safety of the quadrotor throughout the mission.

It is assumed that the vehicles have a relatively accurate knowledge of their environment. The number of unexpected obstacles should therefore be limited and a brute-force A* replaner is sufficient to guarantee an optimized avoidance. Once the system is aware of the threat, it adds the obstacle on the map and runs the A* once again. The next commands sent to the quadrotor will take the presence of the new obstacle into account and avoid it.

C. Cooperation

To match the competences of their manned competitors, automated vehicles need to be developed for numerous applications which require a larger range of action and long-term possibilities. By increasing the number of units deployed, one can also increase the systems capabilities. In the scope of this work, the augmented number of agents widens the surveillance area and enables the system to reach more targets in less time [4]. Cooperative protocols are then implemented for versatility, so that the system could guarantee an optimized assignment of the quadrotors regardless of the number and types of the detected targets.

1) Single fixed target: When a target is spotted in the environment, the system attempts to send an agent to the targets location as fast as possible. If only one vehicle is deployed, it is automatically sent to any target that might appear in the area. The path is planned using the A* algorithm in order to reach the target with the shortest path possible. Although if several agents are used simultaneously, the system calculates the path of all the units to the target and only the closest one is assigned to the target. The others can ignore the assignment and focus on something else as shown in Fig.2.

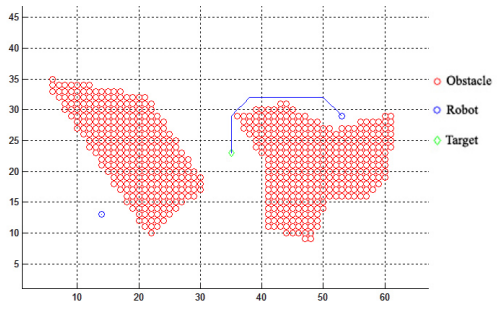

Fig. 2. Single assignment

This assignment method assumes that a target can be covered by a single unit. It would however be easy to imagine scenarios where several agents need to rendezvous at the target's location (Fig.3). For that purpose, the pure pursuit algorithm was modified to create a carrot for unit i of variable speed V_{C_i}. Hence if we need all the vehicles to reach the target at the exact same time, each V_{C_i} can be set to travel the path of unit i in a given time τ. The travel time τ is fixed by the central intelligence based on the kinematic limitations of the

slowest unit. It could also be possible to establish a hierarchy within the vehicles forcing them to reach the target in a certain order. V_{C_i} would then vary depending on the priority level of unit i as well as its distance to the target.

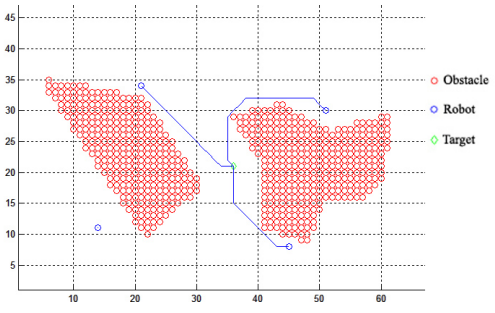

Fig. 3. Multiple assignment

2) Multiple fixed targets: To complete the target tracking protocols it is important to take into account a multi-target assignment. If the number of targets is smaller than the number of quarotors, the objective is to minimize the overall travelled distance. Rather than treating each target individually as they appear, the system tries to optimize the movements of the entire swarm. Every time a new target is detected, it is added to a list of previously detected targets and the allocation process is reiterated. This way each assignment is coherent with the rest of the system. To do so, the user has to deal with all the possible combinations unit/target and when the numbers increase, the optimal allocation quickly stops to be evident. An auction algorithm is therefore implemented to deal with this type of situations.

The Auction algorithm is a method used to solve classical assignment problems and is very well suited for parallel computation [5]. In the basic auction problem, there are n people and n objects (quadrotors and targets respectively) that have to be paired. Each combination person i - object j is associated to a benefit a_{ij} so that the user will try and maximize the total benefit. Mathematically, the aim is to construct n pairs person-object $\{(1, j_1), (2, j_2), ..., (n, j_n)\}$, where all the objects j_1,...,j_n are distinct, that maximizes the total benefit $\Sigma_{i=1}^{n} a_{ij_i}$. The basic algorithm is modified to account for a higher number of vehicles than targets. The aim of the algorithm remains the same but in order to optimize the assignment it is capable of leaving agents out of it.

On the other hand if the number of targets becomes greater than the number of quadrotors, these targets are no longer seen as individual missions but rather as checkpoints (Fig 4). The units have to find the optimal path going through all the targets as quickly as possible. Since the number of quadrotors/targets is not limited, the solution for that problem needs to be versatile enough to answer any type of configuration. To meet this requirement, it was decided to use a genetic algorithm to handle the allocation of the targets.

Even though the method is largely based on the resolution of the multiple Travelling Salesman Problem (mTSP), parts of the method were modified to fit the overall system. In the original Traveling Salesman Problem, the salesman is supposed to come back to the initial location after visiting all the cities thereby creating a looped path. It is also common for a mTSP to fix a single point of origin for all the salesmen and a single end point. This final point can sometimes even be the same as the initial location. For the target-tracking application, each unit has a different initial position and is not required to terminate its path on the exact same node. Once again the priority is given to a minimum time constraint.

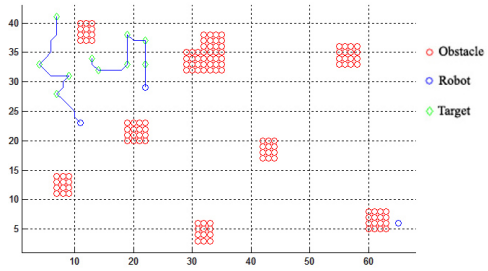

Fig. 4. Targets as checkpoints

To calculate the paths, the Genetic algorithm needs the number of targets $n_{targets}$, the number of quadrotors n_{quads}, and the initial positions of all of the above to create a list of all the targets and generate an array of random break points with $(n_{quads} - 1)$ elements. When applied to the target list, these break points will divide the list into n_{quads} sub-lists representing the n_{quads} paths travelled by the vehicles. The algorithm then needs to evaluate the fitness of the solutions and mutate the fittest genes to converge towards an optimal solution. Similarly to the Auction process, the genetic algorithm was modified so that it can choose to keep units out of the assignment if necessary.

3) Maneuvering target: The next step was to introduce moving targets (also called free-agents) in the environments and have the quadrotors react to it. Here, the objective is not to intercept the target anymore but rather to follow it from a given distance. To avoid predictability, the targets drive through the environment in random arcs and at variable speeds. Since the units are only moving according to the targets whereabouts, it is impossible for the system to plan a collision-free path ahead. An obstacle avoidance scheme based on the potential field method was therefore added to the system to safely follow the free-agents without hitting its surroundings (obstacles or other agents) [6].

To begin the pursuit, the user needs to fix a trailing distance d_{trail} that the vehicle H needs to keep at all time with the target. Depending on the distance D to the target, this unit adapts its speed to keep up with it which is done by creating a distance-to-the-target dependent component K_{H_d} to control its speed. To follow a target moving at V_{free} this component would be:

$$\begin{aligned} K_{H_d} &= V_{free} + k_{H_d} \times (D - d_{trail}) \quad if\ D \geq d_{trail} \\ K_{H_d} &= V_{free} \times k_{H_d} \times (D - d_{trail}) \quad if\ D < d_{trail} \end{aligned} \tag{3}$$

To keep the unit aiming at the target, the vehicle has to maintain its heading aligned with the quad-to-target axis to keep the orientation error e_{H_o} to zero. The term in Eq.4 is added to correct the orientation:

$$K_{H_\phi} = k_{H_\phi} \times e_{H_o} \tag{4}$$

The potential field obstacle avoidance is a method that converts the map known by the system into areas of attractive and repulsive potentials. The attractive field is usually generated by the target we are trying to reach but in our case, the vehicle wants to keep a certain distance with the free agent so the attractive force is excluded. The repulsive forces are generated by the known threats in order to push the vehicle away from them. Their magnitude decreases as the vehicle gets further away from them until a vicinity distance d_0 after which the obstacle will have no repulsive effect.

$$U_{rep_i}(q) = \begin{cases} \frac{1}{2}\gamma_{obst_i}\left(\frac{1}{d_{obst_i}(q)} - \frac{1}{d_0}\right)^\beta & if\ d_{obst_i}(q) < d_0 \\ 0 & if\ d_{obst_i}(q) \geq d_0 \end{cases} \tag{5}$$

Consider the quadrotor at the location $q = (x_{rob}, y_{rob})$ and given the linear nature of the problem, the overall potential results from the sum of the repulsive effects of all the obstacles:

$$U_{rep}(q) = \sum_{i \in obst} U_{rep_i}(q) = k_q \tag{6}$$

With that field in place, the vehicle is rejected as soon as it approaches an obstacle but with the absence of an attractive force from the target, the unit does not know in which direction to avoid the obstacle. To overcome that issue, the repulsive potentials from each obstacle are given an orientation to push the vehicle in the right direction as illustrated in Fig.5.

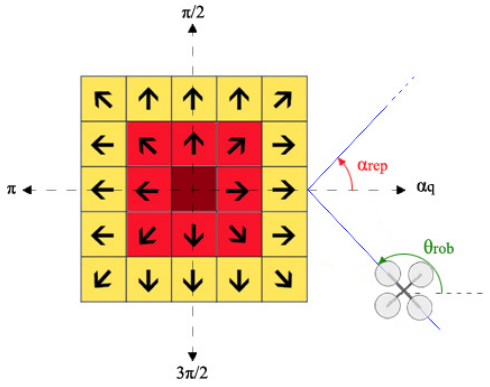

Fig. 5. Directions of rejection around an obstacle

Once again the effect on the quadrotor is converted into a new factor K_{obst}:

$$K_{H_{obst}} = \frac{k_{obst} \times k_q}{\alpha_{rep}}$$
$$with \quad \alpha_{rep} = mod\left[\theta_{rob} + \pi, 2\pi\right] - \alpha_q \tag{7}$$

The 2D point mass model used so far is controlled through differential drive. For this particular case, the coefficients presented would be used as follow:

$$V_{H_L} = K_{H_d} - (K_{H_\phi} - K_{obst})$$
$$V_{H_R} = K_{H_d} + (K_{H_\phi} - K_{obst}) \tag{8}$$

k_{H_d}, k_{H_ϕ} and k_{obst} were calibrated to ensure a smooth path around the obstacles while constantly aiming at the target ($k_{H_\phi} = R2D/5$, $k_{H_d} = 1$ and $k_{obst} = 100$).

4) Surveillance: Throughout this paper, the system has the ability the pick which units to assign and which to keep out of the assignment. The unassigned units can then divide the area to survey amongst them to cover it faster and more efficiently. When a new target is spotted (most likely a moving one for a realistic scenario) the system can be set to react in 2 ways.

- The closest unassigned unit is sent to the target. The remaining vehicles restart the division of the area to include the part left out by the assigned unit. This unit will follow the target wherever it goes.
- The unit whose surveillance zone includes the targets location is assigned to it while the others stay on their current path. When the target gets to another zone, the chasing agent goes back to surveillance and the agent whose zone has been entered starts chasing the target. This solution becomes interesting when each quadrotor is pre-assigned to a surveillance zone and cannot fly out of it. The reason can be that the vehicle needs to stay at all times close to a station (to charge or communicate information) in its own area.

III. Initial Results

All the methods introduced previously were then centralized into a single coherent system accessible through a graphical user interface (GUI). The user can change the number of units deployed, add/remove fixed targets and free-agents, modify the collaboration method to deal with manoeuvring targets, and take control of a live vehicle. At this stage, the algorithms were tested in 2D scenarios so the physical platforms used for testing were ground robots.

A camera tracking device is installed to get the position of every object in the surveillance area. That position is sent to the central computer that can follow the movements of the live vehicle in real-time while the computer simulates the robot's odometry on its own. By fusing both information through a Kalman filter the system could accurately locate the vehicles and correct potential errors in their paths via the Kalman updates. For the implementation of the filter, it is important to specify that the kinematics of the system are linear and that each iteration of the filter will occur at fixed discrete time intervals, meaning that the time evolution of the state vector can be calculated by means of a state transition matrix. The accuracy of the filter was tested on several types of trajectories, from the simple straight line to a figure of 8, and the physical vehicle was always kept on its path. But in order to successfully extend to the quadrotor platform the filter needs to be robust.

The first series of tests consisted in creating systematic errors as would the IMU bias create on the quadrotor platform.

To simulate that, Fig.6 shows the path travelled by the robot ROB_{real} with a wheel larger than the other (x1.5). The original path of the simulated vehicle ROB_{sim} is travelled three times in a row to make sure the physical one returns to its original position.

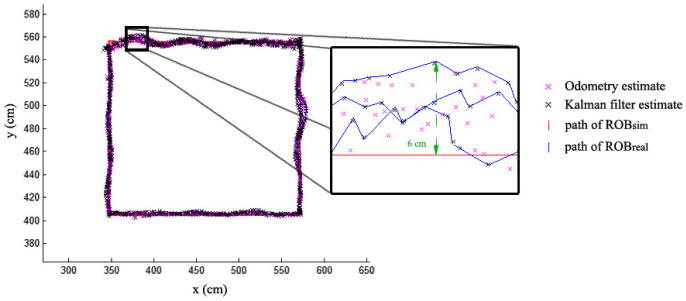

Fig. 6. Influence of a systematic error on a square path

Oscillations appear after the turns but they are rapidly damped in the straight segment. The maximum deviation from the original path was of 6cm representing only a third of the platform's width. This shows that the Kalman filter and on-board controller can keep the agent on track and on time even with sensor bias or physical damage.

To test the effects of non-systematic errors, the vehicle is returned to its calibrated state to travel along the same path. During a turn, one of the wheels is blocked to induce an error in heading then the robot is manually moved in the middle of its path to create a significant error in position. In both cases the vehicle finds its way back to the original path.

The discrete Kalman filter proves to be robust enough to overcome the effects of systematic and non-systematic errors.

IV. Extension to the third dimension

Once the algorithms were extensively tested in their 2D environment to prove their accuracy and robustness, they were extended to the third dimension. The model of the vehicles was switched back to a proper quadrotor helicopter [7][8] and the previous methods were optimized for 3D applications towards health management.

A. Quadrotor model

The quadrotor is very simply modelled as four rotors equipped at each end of a cross. All the propellers axes are parallel and fixed directly on the DC motors shaft. The blades have a fixed pitch and push the air downwards. Fig.7 shows the quadrotor structure in hover. Each propeller is defined by an orientation of rotation (blue circular vector), a rotating speed ($\Omega_1, \Omega_2, \Omega_3, \Omega_4$ for respectively the front, right, rear and left rotor) and a velocity (blue vertical vector) representing the amount of thrust created by the motor/propeller assembly. The fixed-body B-frame, in red, is set up to be front / left / up so that the altitude is measured positively as the quadrotor is climbing.

To take advantage of the body symmetry and keep the inertia matrix time-invariant, the equations of motion are formulated

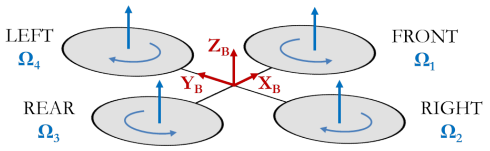

Fig. 7. Simplified quadrotor in hover

in the B-frame hence two assumptions can easily be made to simplify the model:

- The origin of the B-frame is coincident with the quadrotor's centre of mass.
- The quadrotor's axes of inertia coincide with the axes of the B-frame.

Under these assumptions and following the Newton-Euler formalism, the dynamics of a rigid body subject to external forces applied to its centre of mass can be expressed in the B-frame by Eq.9.

$$\begin{bmatrix} mI_{3\times3} & 0 \\ 0 & I \end{bmatrix} \begin{bmatrix} \dot{V} \\ \dot{\omega} \end{bmatrix} + \begin{bmatrix} \omega.mV \\ \omega.I\omega \end{bmatrix} = \begin{bmatrix} F \\ \tau \end{bmatrix} \tag{9}$$

To comply with real-time computation capabilities, the less influent effects such as the hub forces, ground effect and rolling moments were neglected and the thrust/drag coefficients were assumed constant. The matrix system (9) can then be rewritten in a state-space form $\dot{X} = f(X, U)$ where U is the input vector and X the state vector chosen as follows:

$$\begin{aligned} X &= [\, \phi\ \dot{\phi}\ \theta\ \dot{\theta}\ \psi\ \dot{\psi}\ x\ \dot{x}\ y\ \dot{y}\ z\ \dot{z}\,]^T \\ U &= [\, U_1\ U_2\ U_3\ U_4\,]^T \end{aligned} \tag{10}$$

with

$$\begin{aligned} U_1 &= b\,(\Omega_1^2 + \Omega_2^2 + \Omega_3^2 + \Omega_4^2) \\ U_2 &= l\,b\,(-\Omega_2^2 + \Omega_4^2) \\ U_3 &= l\,b\,-(\Omega_1^2 + \Omega_3^2) \\ U_4 &= d\,(-\Omega_1^2 + \Omega_2^2 - \Omega_3^2 + \Omega_4^2) \\ \Omega &= -\Omega_1 + \Omega_2 - \Omega_3 + \Omega_4 \end{aligned} \tag{11}$$

From which we can derive the following equations of motion:

$$\begin{aligned} \ddot{\phi} &= \dot{\theta}\,\dot{\psi}\,\tfrac{I_{yy}-I_{zz}}{I_{xx}} - \dot{\theta}\,\Omega\,\tfrac{J_R}{I_{xx}} + U_2\,\tfrac{l}{I_{xx}} \\ \ddot{\theta} &= \dot{\phi}\,\dot{\psi}\,\tfrac{I_{zz}-I_{xx}}{I_{yy}} + \dot{\phi}\,\Omega\,\tfrac{J_R}{I_{yy}} + U_3\,\tfrac{l}{I_y} \\ \ddot{\psi} &= \dot{\phi}\,\dot{\theta}\,\tfrac{I_{xx}-I_{yy}}{I_{zz}} + \tfrac{U_4}{I_{zz}} \\ \ddot{x} &= (s_\psi\,s_\phi\, + c_\psi\,s_\theta\,c_\phi)\,\tfrac{U_1}{m} \\ \ddot{y} &= (-c_\psi\,s_\phi\, + s_\psi\,s_\theta\,c_\phi)\,\tfrac{U_1}{m} \\ \ddot{z} &= -g + c_\theta\,c_\phi\,\tfrac{U_1}{m} \end{aligned} \tag{12}$$

A PID controller was added to the model to stabilize the model and then tested on the physical quadrotor. Without feedback from the camera tracking device presented above, the position controller was turned off for the first flights. The objective was to see if by simply controlling the throttle the attitude controller can stabilize the platform in hover. With the exact same PID coefficient as in the simulation, the flight was successful. A slight drift appeared on the X axis just like the simulation of a lone attitude controller would suggest but it was easily decreased by tuning the ESCs and the PID

coefficients (all less than 10% away from their simulation value) and will be completely cancelled once the position controller is in place.

B. Improvements on the 2D protocols

The A* path planner and the carrot following method were both kept for the guidance of the quadrotor platform. Compared to their previous centralized solution adopted for the 2D application, the decision-making is much more distributed. The quadrotor offers more processing power to run more complex algorithm so that both the path planning and tracking techniques could be run on-board. Apart from increasing the number of primitive movement at each iteration, the A* process stayed the same. Fig.8 shows the 3D path generated by the improved A* as well as the quadrotor following the carrot point as it travels the path.

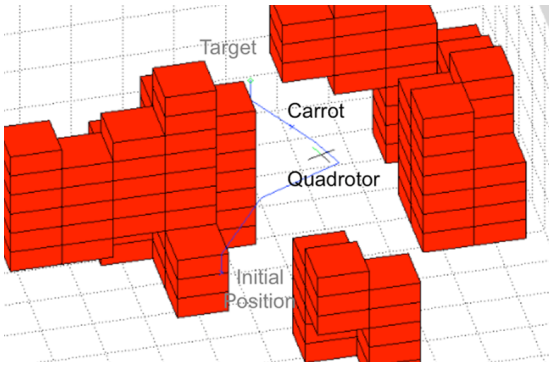

Fig. 8. Pure pursuit path following

On the other hand, significant changes were made to the pure pursuit to take more control over the carrot and facilitate the integration of a new Conflict Detection and Resolution scheme. The Distributed Reactive Collision Avoidance (DRCA) algorithm [9] is being developed to guarantee a conflict resolution scheme for n nonholonmic vehicles, based on speed changes and lateral manoeuvres. It also distributes the computation load among the different subsystems. Consequently, the central computer can focus on a high-level task management and would be able to handle more units. Hence the overall system does not depend exclusively on the centralized intelligence making it more robust to eventual bugs.

Originally the DRCA takes into account the dynamic limitations of the platform in the conflict resolution scheme. Here the carrot is assumed to be a projection of quadrotor in the near future so that the avoidance process can be applied to the carrot instead of the vehicle. Since the carrot has no reality, it has no dynamic limitations. It can be created, moved and deleted freely to go around an obstacle or avoid an incoming friendly agent. The dynamics of the quadrotor will be handled by the controller and path tracking methods already in place. This will enable the generation of new avoidance trajectory and extend the effectiveness of the DRCA to more complex scenarios.

The Auction algorithm was kept to deal with the target assignment problems, and was improve to incorporate the case of $n_{targets} > n_{units}$ previously covered by the genetic algorithm. Though it was giving good results, the genetic algorithm is computationally heavy and does not fit the real time expectations of this work. The new Auction algorithm also has the ability to assign units as to optimize time rather than energy. The 2D version tried to minimize the added length of all the paths thus minimizing the power consumed by the swarm. Now the algorithm can choose to minimize the length of the longest path thus trying to optimize the time needed to reach all the targets.

V. Further Work: Towards full autonomy

The quadrotor should be able to detect a faulty behaviour from its sensors and actuators. Thus the on-board decision making would be performed based on the current state of the vehicle and its priority level could be changed depending on the significance of the recorded problem. Once the central computer is aware of the issue, it can reassign the other agents to account for the vehicle in trouble.

The integrity of the swarm, the safety of its direct environment and the success of the mission will always prevail on the individual. In case of a minor issue however, the central intelligence can land the agent and warn the operator that it needs to be fixed or replaced.

VI. Conclusion

In this paper we presented a set of algorithms grouped into a single method in order to control the behaviour of a multi-agent system. The work was first restricted to a two dimensional scenario to focus on the versatility of the target assignment and its real time capabilities. All the scenarios implemented were successfully analysed by the task manager and properly carried out by the vehicles. The algorithms were then extended to the third dimension and optimized to account for the unstable quadrotor dynamics. Significant modifications on the pure pursuit method enabled a smooth and safe trajectory in a dense environment. The path tracking algorithm will be further improved to incorporate a DRCA-inspired conflict resolution and avoidance protocols in partially known environments.

References

[1] E. Saad, J. Vian, G. J. Clark, and S. Bieniawski, *Vehicle Swarm Rapid Prototyping Testbed*. The Boeing Company, Seattle, WA, 98124.
[2] G. M. Hoffmann, H. Huang, S. L. Waslander, C. J. Tomlin, *Precision Flight Control for A Multi-Vehicle Quadrotor Helicopter Testbed*. 2011.
[3] Khayyam, *Recherche de chemin par l'algorithme A**. 2008.
[4] A. Tsourdos, *Cooperative Path, Planning of Unmanned Aerial Vehicles*. 2011.
[5] D. P. Bertsekas, *Auction algorithms for network flow problems*. Computational Optimization and Applications. 1992.
[6] M. Becker, C. Dantas and W. Macedo, *Obstacle Avoidance Procedure for Mobile Robots*. Mechatronics, vol. 2. 2006.
[7] T. Bresciani, *Modelling , Identification and Control of a Quadrotor Helicopter*. 2008.
[8] S. Bouabdallah, *Design and Control of Quadrotors with Application to Autonomous Flying*. PhD thesis, 2007.
[9] E. Lalish, K. A. Morgansen, *Distributed reactive collision avoidance*. Autonomous Robots, special issue, 2012.