Multi-Robot Coordination: C-CAPT and Modified CAPT Extended Algorithm Implementation

Jay Davey
School of Engineering and Applied Science
University of Pennsylvania
jaydavey@seas.upenn.edu

Eduardo Garcia
School of Engineering and Applied Science
University of Pennsylvania
edgarc@seas.upenn.edu

Caio Mucchiani
School of Engineering and Applied Science
University of Pennsylvania
caio@seas.upenn.edu

***Abstract*— Multi-robot coordination is one interesting challenge for micro-UAVs. In this sense, this paper includes the implementation of the state-of-the-art algorithms to coordinate the assignment of N-robots from an initial position S to a goal final position G. Initially, the linear assignment problem is done with the Hungarian algorithm implementation, and for the second part the CAPT algorithm is extended to the implementation of the Goal Assignment and Trajectory Planning algorithm (GAP)**

***Index Terms*—CAPT, multi-robot, GAP,C-CAPT.**

Introduction

Multi-robot assignment tasks represent a challenge for the actual robotic scenario, and it is a problem that should be solved efficiently. In this paper, we use the algorithms discussed in [1] and [2] and the approach to the multi-robot assignment problem is done in two different manners: the first phase consists of an implementation of the centralized CAPT algorithm (C-CAPT) for circular first order robots that operates in a 2D obstacle free environment. When scaling to 3D we discuss the algorithm in terms of runtime against number of robots (which varies from 10 up to 1000), with equal number of goals. We expanded this to a 3D simulation of quadrotors taking into account quadrotor dynamics that showed consideration for downwash from the flying robots. Robots flying above each required a minimum distance above which quadrotors should be allowed to pass which was different from the nominal x-y margin of $2R\sqrt{2}$. Furthermore, we showed the algorithm to work in two further cases; 1) Number of robots was greater than the available goal locations, and 2) when there existed more goal locations than available robots.

For the second case, the Goal Assignment and Planning algorithm proposed in [2] was implemented, in order to fly the quadrotors in an obstacle-known environment, and assign their priority depending on their initial condition and trajectory planning, with cubic complexity in the number of robots. Simulations were made to test all implementations and videos are attached to this paper.

I. C-CAPT ALGORITHM

The concurrent assignment and planning of trajectories (CAPT) algorithm is defined considering an N multi-robot task assignment problem in which the robots are unlabeled (or it does not matter who and where robots are assigned) and assigned to M goal locations. The proposed modification of this problem consists of an obstacle-free environment and combines the sub-problems of assignment and trajectory generation, seeking to provide a computationally tractable solution for large number of robots. This centralized version of the CAPT algorithm is known as a centralized CAPT (or C-CAPT).

II. C-CAPT FOR FIRST ORDER ROBOTS

As the first part of this paper, we implemented the C-CAPT algorithm assuming a 2D obstacle free workspace filled with circular first order robots. Under this scenario, and following [1] a few assumptions are made:

(A1) Interchangeable and homogeneous robots, and no preference for goal location among them
(A2) Circular robots with radius R
(A3) Obstacle free region (defined as Σ)
(A4) Random initial locations for the robots
(A5) Random locations for the goals
(A6) Robots fully actuated and perfect state knowledge
(**A7**) The robot stage area of flight α is defined in Σ

We define the space Σ as an obstacle free region and as the Minkowski sum of the circular robot and the union of the initial and final location. The formal definition will not be addressed here since it can be found on [1]. The robot stage area α represents a 2D area defined as follows (dimensions are in meters):

$$x_{min} = -2;\ x_{max} = 4$$
$$y_{min} = -2\ ;\ y_{max} = 1$$

We also define the robot as a circle of radius R = 0.08 meters.

III. Implementation of C-CAPT for phase 1

Based on assumptions from A1 to A7, we implemented the C-CAPT using MATLAB. Since the interest of this Phase focus on runtime for this algorithm, the number of robots was varied according to the following vector:

$$N_{Robots} = [10, 50, 100, 200, 300, \ldots \ldots, 1000]$$

10 trials ($N_{trials} = 10$) for each N_{Robots} were computed and the minimum, median and maximum runtime values were recorded and graphed in Figure 3.

A. Start and goal location

The assignment for the start and goal position for this phase was done randomly within the boundary limits allowing for a margin of $R\sqrt{2}$ to avoid collisions with the borders of the obstacle free space.

B. Hungarian Algorithm

For the solution of the linear assignment problem, the Hungarian algorithm was implemented following the steps from [5]:

- **Step 0**: For a $n \times m$ matrix (the cost matrix), where each element represents the cost assignment of one robot from a start to a goal position, we rotate the matrix so that the number of columns is equal or higher than the number of rows. Let k be the minimum value of the number of rows and columns;
- **Step 1:** For each element in the cost matrix, look for the smallest and subtract it from every present element in its row; Go to step 2.
- **Step 2:** Find a zero in the resulting matrix, and mark its row or column in case it has not being marked. Repeat this for each element in the cost matrix; Go to step 3.
- **Step 3:** Cover each marked column from Step 2. If k columns are covered, this corresponds to a complete assignment and the algorithm is done; otherwise, go to step 4.
- **Step 4:** Find an uncovered zero and prime it. In case there is no marked zero in the row contained this primed zero, go to step 5. Otherwise, cover this row, uncover the column containing the starred zero. Continue this process until there are no uncovered zeros left and save the smallest uncovered value; Go to step 6.
- **Step 5**:Alternate primes and zeros as:

  $Z_0 \rightarrow uncovered\ primed\ zeros\ from\ Step\ 4$
  $Z_1 \rightarrow marked\ zeros\ from\ Z_0 (if\ any)$
  $Z_2 \rightarrow primed\ zero\ in\ the\ row\ of\ Z_1 (always\ exists)$

  Continue until the series terminates at a primed zero that has no marked zero in its column. Unmark each starred zero of the series, mark each primed zero of the series, erase all primes and uncover every line in the matrix; Return to Step 3.

- **Step 6:** Add the minimum uncovered value to every element of each covered row, and subtract it from every element of each uncovered column. Return to Step 4 without altering any marked, primes, or covered lines.

This algorithm was implemented on MATLAB and the following results from this implementation.

C. Cost function minimization

C-CAPT algorithm seeks [1] to minimize the following cost functional in order to obtain an optimal trajectory $\gamma * (t)$:

$$\gamma * (t) = \underset{\gamma(t)}{argmin} \int_{t0}^{tf} L\big(\gamma(t)\big) dt \qquad (1)$$

Where (1) is subjected to the following conditions:

- Valid assignment
- Full Resource utilization
- Initial and Terminal conditions
- Robot Capabilities
- Collision avoidance

Added to these conditions, we shall consider the ones mentioned from A1 to A7 in II.

D. Runtime for C-CAPT implementation

For the implementation of the CAPT algorithm, start and goal locations for the 2D environment were generated randomly and the cost matrix defined as:

$S \rightarrow N_{robot}\ collision\ free, random\ start\ locations$
$G \rightarrow M\ collision\ free, random\ goal\ locations$

$$D = (S - G)^2$$

where *D* represents the cost matrix. For the 2D case, we have:

$$N_{Robots} = 40$$
$$N_{Goals} = 30$$

From assumption (A2), and the α area previously defined, the results obtained were

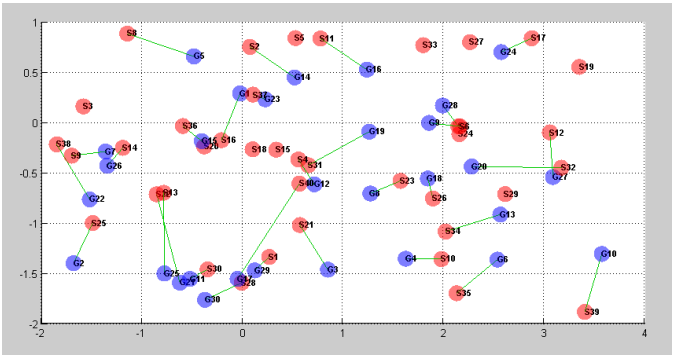


Fig. 1 2D C-APT for Phase 1

For the 3D case, the following area β was defined:

$$x_{min} = -2;\ x_{max} = 4$$
$$y_{min} = -2\ ;\ y_{max} = 1$$
$$z_{min} = 0\ ;\ z_{max} = 2$$

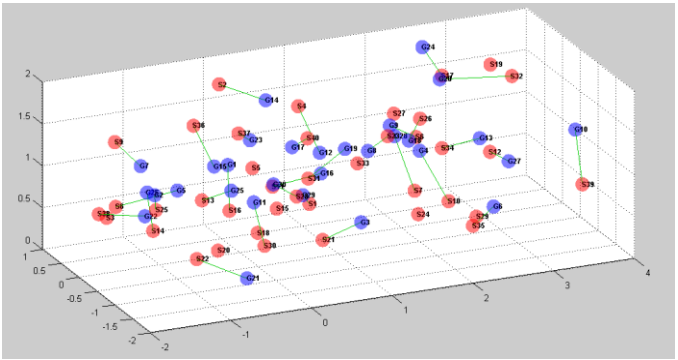


Fig. 2 3D C-CAPT for Phase 1

The implementation of the C-CAPT algorithm for N robots varying from 10 to 1000, navigating to the same number of goals in each case was done by following the process described in B. The result for the implementation (which can be found attached to this document) is as follows:

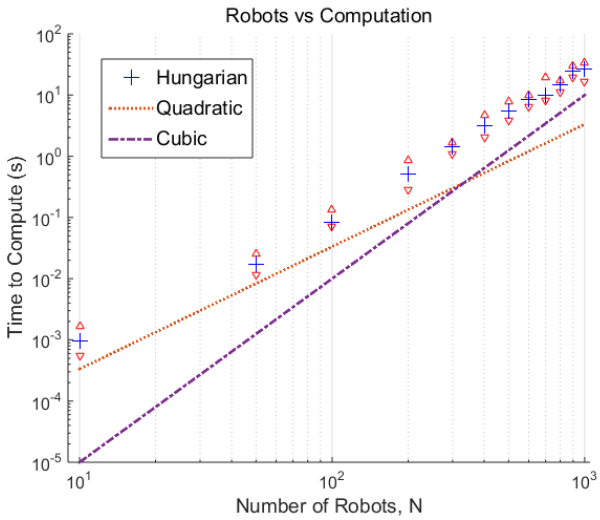


Fig. 3. Runtime for Hungarian ,Quadratic and Cubic methods for multiple robot assignment

We can see from the graph that the computational Hungarian method is approximately of cubic form with respect to N_{robots}.

IV. Implementation of the Extended CAPT Algorithm through the Goal Assignment and Trajectory Planning for Phase II

In order to extend the algorithm implemented on Phase I, a few modifications with regard to the search method and start and goal assignment were made. The quadrotor simulator written by K'Mel [6] was used in this Phase.

A. C-CAPT for 3D modification

To implement the C-CAPT algorithm in the 3D case, it was necessary to perform a change of variables and quadrotor aerodynamic model.

For the cost minimization problem, the idea is now to minimize the fourth derivative of position (snap), differently than just the first derivative (acceleration) [1] as follows:

$$minimize \int_{t0}^{tf} \ddot{X}'(t)^T \ddot{X}'(t)\, dt \qquad (2)$$

This justifies as for quadrotors as it yields reference trajectories and inputs highly desirable for the latter. Likewise, since the quadrotor flies in a 3D environment and in teams, the aerodynamic model should be carefully planned so that safety is guaranteed.

To guarantee a safe interaction among the quadrotor team, an ellipsoidal model [1] was implemented to ensure a separation between vertical and horizontal direction. This change is necessary as a quadrotor that flies under another will be subjected to downwash from the quadrotors above, and for

this reason a larger separation on the vertical direction rather than the horizontal is justifies the ellipsoidal model:

$$\sqrt{\left(\frac{x_i(t)-x_f(t)}{1}\right)^2+\left(\frac{y_i(t)-y_f(t)}{1}\right)^2+\left(\frac{z_i(t)-z_f(t)}{4}\right)^2} > 2R \;\forall\; i,j,t \quad (3)$$

Eq. (3) implies that the separation in the vertical direction for a safe condition among quadrotors should be of at least 8R.

For a teams of 10 and 15 quadrotors, the new stage limits for flying area were defined as:

$$x_{min} = -1;\ x_{max} = 3$$
$$y_{min} = -1\ ;\ y_{max} = 3$$
$$z_{min} = 0\ ;\ z_{max} = 1$$

The results for the 3D C-CAPT algorithm can be viewed as follows:

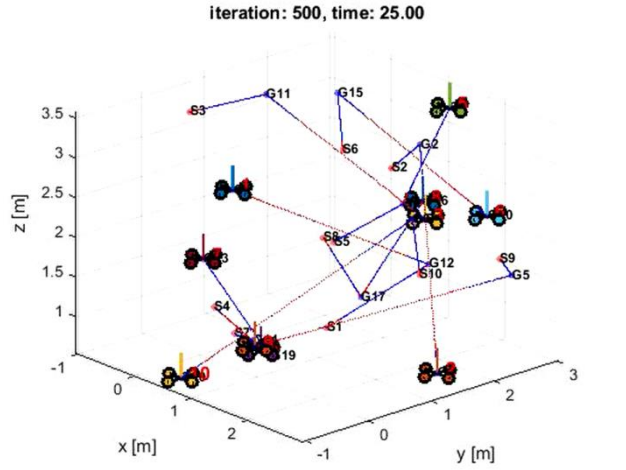


Fig. 4. 3D C-CAPT for a team of 10 quadrotors with twice number of goals

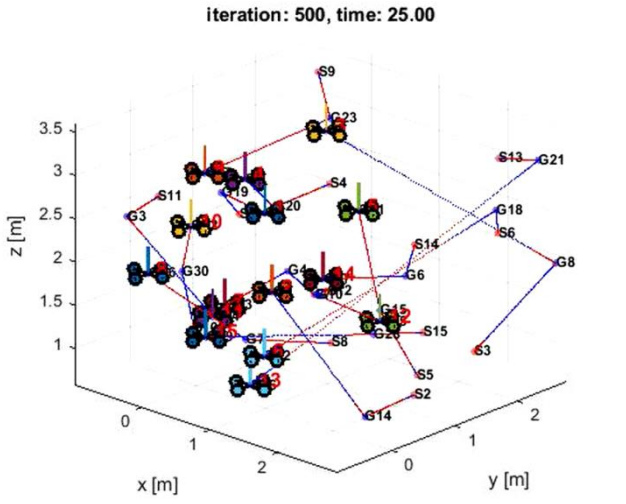


Fig 3. C-CAPT for a team of 15 quadrotors with twice number of goals

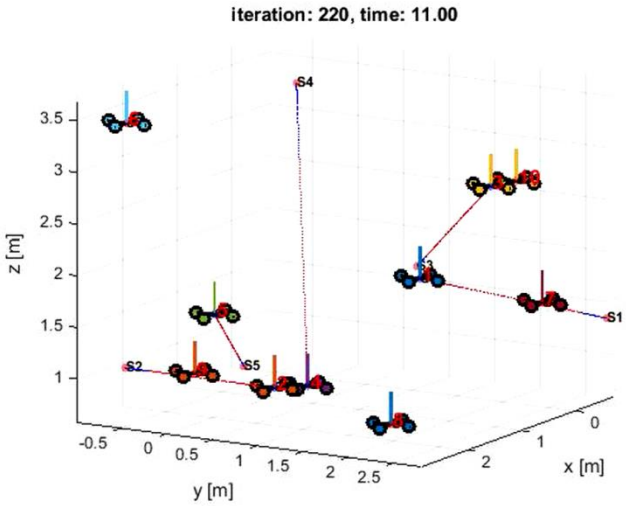


Fig 4. C-CAPT for a team of 10 quadrotors with less goals than quadrotors

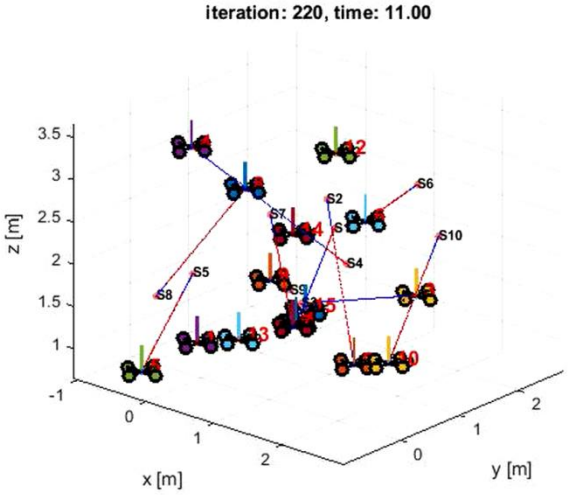


Fig 5. C-CAPT for a team of 15 quadrotors with less goals than quadrotors

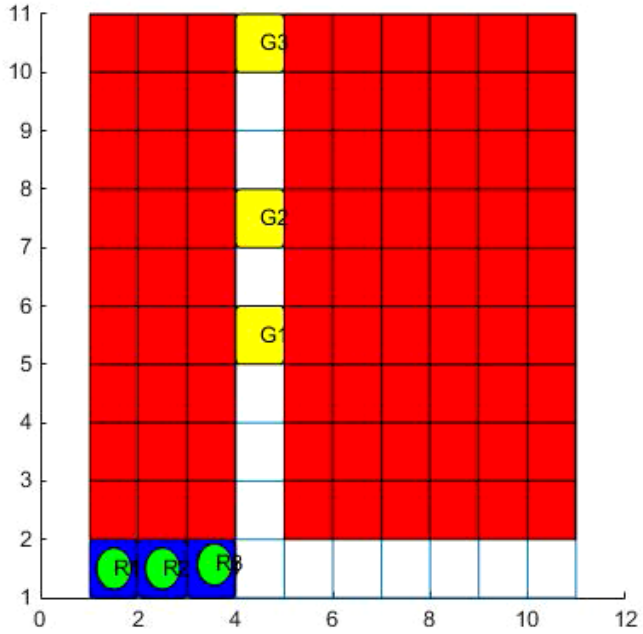
It should be pointed out that for the case where there were more goals than quadrotors, the C-CAPT algorithm had to be iterated twice. The video detailing these scenarios is attached to this paper.

B. Goal Assignment and Trajectory Planning (GAP)

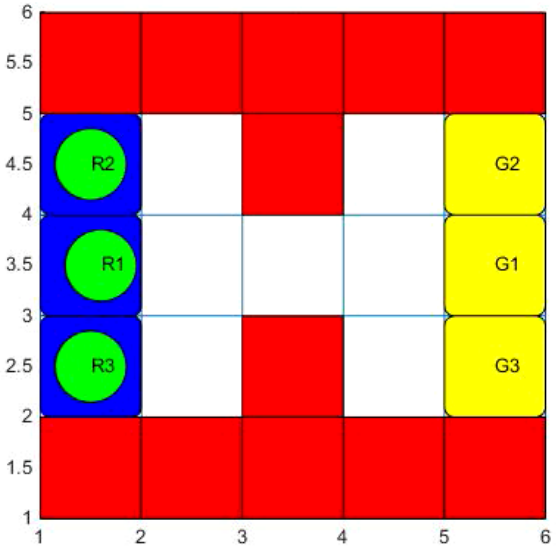
In order to extend the C-CAPT algorithm to operate in a known obstacle filled environment, the method implemented by [2] was chosen.

Following the algorithm described in (2), the following cases were proposed:

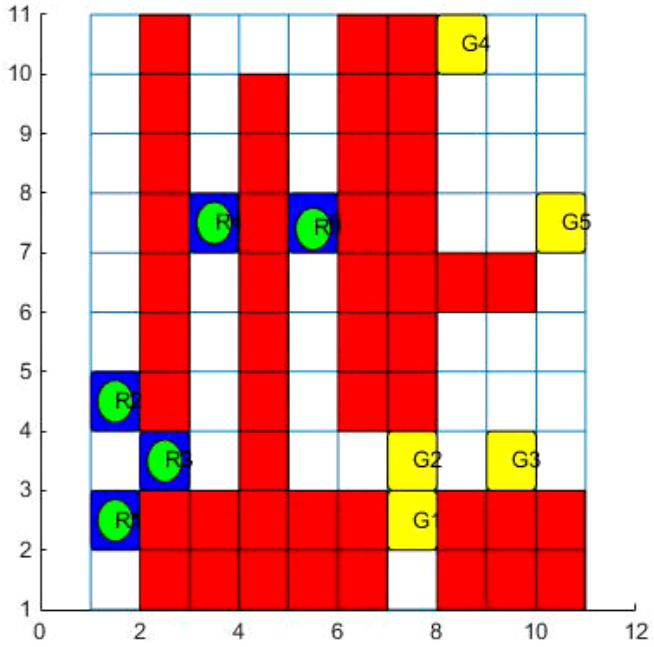
CASE 1: Three Quadrotors and obstacle Environment A



CASE 2: Three Quadrotors and obstacle Environment B



CASE 3: Five Quadrotors and obstacle Environment C



For the implementation of the GAP algorithm, the minimum cost path was established via A*'s algorithm, so that the cost matrix D could be created, and afterwards find the start and goal assignment, as well as define the priority order and trajectory generation through GAP implementation. This resulted in:

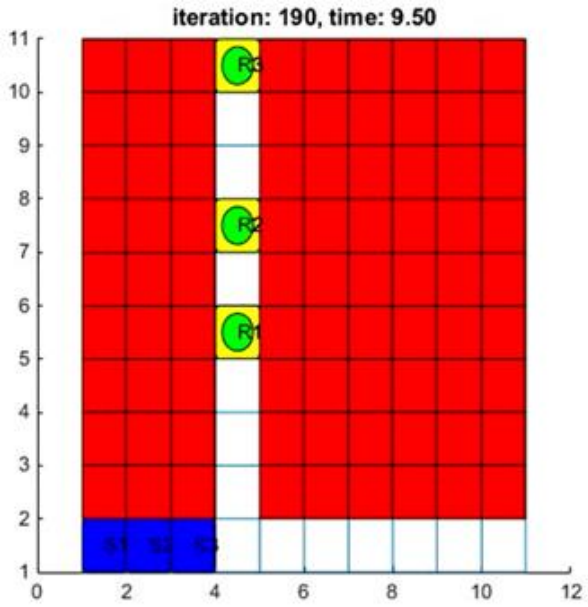


Fig. 6 Result for CASE 1

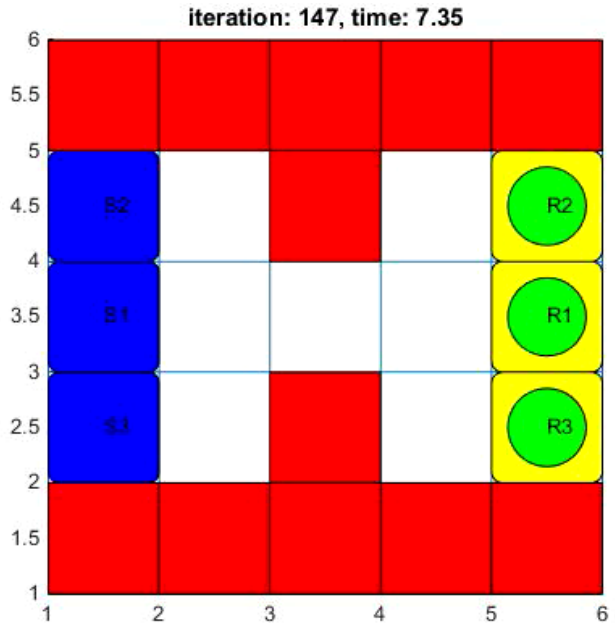


Fig. 7 Result for CASE 2

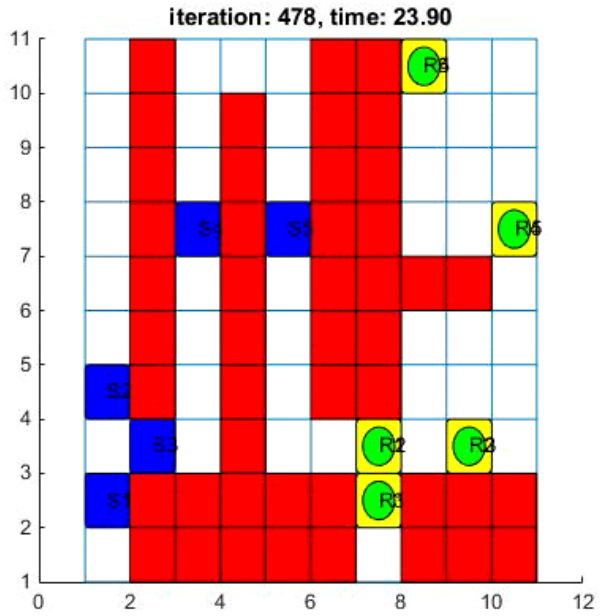


Fig. 8 Result for CASE 3

The algorithm performed efficiently in all three cases and detailed video of its performance can be found attached to this paper.

ANALYSIS OF TEST CASES – ADVANTAGES AND DISADVANTAGES

The three test cases shown for GAP (CASE 1-3), where specifically made to show the algorithm's prioritization of robots traversing a narrow hallway that only permits a single robot to move through the gap at a time (extreme cases when robots share a node on a path).

CASE 1 shows how the algorithm will assign goals to achieve this in such a way that the robots would not collide or overlap. The algorithm succeeded at both goal assignment and collision free path planning by making the robots traverse hallways (possible paths with a minimum thickness of one robot wide) in a queued order that prevents robots colliding along the way. The assignment was created in such a way that each robot travelled along the hallway at a speed that maintained traffic flow and no robots needed to overtake other robots when seeking goals.

For CASE 2 and CASE 3, these cases show the effect of prioritization of robots when paths are intersecting. Where there is a narrow gap and nodes on paths intersect, the algorithm effectively assigns a priority to each robot crossing the node by shifting the time it is allowed to enter the gap to allow a higher priority robot to enter first. The time shift is such that the robots will not collide and a safe minimum distance is maintained.

From these cases the advantages of these algorithms are that they are simple to implement, the paths that are planned are safe and collision free, and they are computationally acceptable (approximately cubic). The algorithm is tractable given the assumption that robots are interchangeable. Some disadvantages of this implementation was the need to discretize the map which does not scale well for path planning purposes with A*. Other disadvantages include a lack of consideration to disturbances in individual robot real-time performance. If a robot is not performing satisfactorily, collisions may still be possible where one robot may catch up to another if it is not tracking its trajectory well.

CONCLUSION

By the implementation of the C-CAPT algorithm for 2D and 3D cases, the efficiency for task assignment was measured in terms of runtime. It was observed that although the computational time grew in a factorial rate, considering the number of robots being assigned the algorithm had a good response. For the second phase, even though the obstacles were known, assigning multiple robots susceptible to collision needed not only task assignment, but also path planning and priority establishment, where the former was implemented using A*'s algorithm and the latter by extending C-CAPT with [2]. Within the proposed three cases, the algorithm performed correctly as expected.

ACKNOWLEDGMENT

The authors would like to thank Dr. Vijay Kumar, Dr. Kostas Daniilidis and the entire Teaching Assistants group of the MEAM 620 course for all the help in accomplishing this project.

REFERENCES

[1] M. Turpin, N. Michael and V. Kumar "CAPT: Concurrent assignment and planning of trajectories for multiple robots," The International Journal of Robotics Research 2014,Vol 33(1)

[2] M. Turpin ,K. Mohta , N. Michael and V. Kumar : "Goal Assignment and Trajectory Planning for Large Teams of Aerial"

[3] W. Burgar, M. Moors, C. Stachniss, F.Schneider : "Coordinated Multi-Robot Exploration" University of Bonn- Germany

[4] L. Luo, N. Chakraborty, K. Sycara: "Multi-Robot Assignment Algorithm for Tasks with Set Precedence Constraints"

[5] "Munkres' Assignment Algorithm Modified for Rectangular Matrices" : available at http://csclab.murraystate.edu/bob.pilgrim/445/munkres.html