ASEN 5519: Final Project

Seif Said*
University of Colorado, Boulder

Multi-Path Planning for Quadcopters

*SID: 106981440

I. INTRODUCTION

With the world leaning towards autonomous capabilities for various tasks every day more than the other, motion planning problems are becoming more prevalent than ever. The scenario being addressed here is that of package delivery. The development of the planner is centered around 3 quadcopters that are to traverse to package locations on a given map and then traverse to a final goal region where the packages are to be dropped off. The focus of the development is the multipath planning of the 3 quadcopters and not the dynamics questions that arise from picking up a package and maneuvering around obstacles. Furthermore, the motion planner will be used in a small scale context (i.e. small maps and not, for example, fully scaled suburban neighborhoods). The development begins by creating a motion planner for 1 quadcopter; then, the planner is developed for 2 quadcopters with collision checking for the separate trajectories; then, the planner is developed for the case of 3 quadcopters. The base planner that is being used here is RRT with kindynamical constraints, where control inputs are sampled randomly for the extension of the tree. Furthermore, the space that is used for planning is the state space.

II. DYNAMICS & CONTROL MODEL

A. Linearized Dynamics Model

For this project, a linear dynamics model was used for the quadcopter. This model was obtained from ASEN 5014 (Linear Control Design). This model makes several assumptions for the formation of the dynamics. First, it is assumed that the quadcopter contains a separate controller that counters gravity and, thus, it is not accounted for in the model. Second, it is assumed that the rotors of the quadcopter are rotated inward such that the system is naturally lyapunov stable without any control. Third, the quadcopter is assumed to be symmetric. Fourth, the system is also assumed to contain a second separate controller to maintain the quadcopter orientation (i.e. the quadcoper is always pointing along the inertial x-axis). Furthermore, the dynamics are linearized about the hovering state of the quadcopter. The formalation leads to the following state space realization,

$$\dot{x} = Ax + Bu \tag{1}$$

$$\dot{\mathbf{x}} = \begin{bmatrix} 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & -0.0104 & 0 & 0 & 0 & \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & -0.0104 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & -0.0208 \end{bmatrix} \mathbf{x} + \begin{bmatrix} 0 & 0 & 0 & 0 \\ -0.04167 & 0 & 0.04167 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & -0.04167 & 0 & 0.04167 \\ 0 & 0 & 0 & 0 \\ 0.4 & 0.4 & 0.4 & 0.4 \end{bmatrix} \mathbf{u}$$

The state vector for this model is $x = [x, \dot{x}, y, \dot{y}, z, \dot{z}]^T$. The values for the constants shown in the matrices have been experimentally obtained and were provided on the ASEN 5014 document for the model. For more details on the model, see the document in the appendix.

B. Control Input

The control input that is used for the propagation of the quadcopter in this planner is known as the minimum energy input. It is an open loop control input. it has the following form,

$$\mathbf{u} = B^T e^{A^T} r \tag{2}$$

Here A and B are the same matrices specified for the linearized dynamics model above. r is a specified input that results in a trajectory from any arbitrary initial condition to any arbitrary final condition; the system is fully reachable.

C. Kinematic & Control Constraints

As mentioned in section A, the dynamics are linearized about the hover state, and thus, are only accurate within the neighborhood of the hover state. In order to ensure that the quadcopter's movements remain within the neighborhood, constraints were set on the input r as well as the velocity of the quadcopter. These constraints were determined on the basis of creating smooth trajectories as well as small Δx propagations. r is a 6x1 vector and all of its elements were constrainted to values between (-1,1) except for the last two elements which directly affect the vertical translation and velocity; the range was set to (-0.01,0.01) as that created a smoother vertical translation. The velocity was constrained to values of (-1,1) in each of the x,y, and z components. These ranges were tuned through trial and error that was performed on 1 quadcopter motion planning and the analysis that was performed on this system in ASEN 5014.

III. PROBLEM APPROACH

The development of the planner (RRT with kinodynamics) began by constructing it around the solution for the trajectory of only 1 quadcopter to ensure that the kinodynamic planning was functioning correctly. This is the stage where the constraints on the system and control were tuned for the dynamics. From there, the incorporation of intermediate goals in the trajectory of the quadcopter was implemented; A tree would grow from the start location towards the intermediate goal location and after the intermediate goal is reached by the tree a second tree grows from that end node of the first tree towards the final goal location. Once a solution is found, a path going through the first tree and the second tree is returned. The next stage of the development involved adjusting the planner for the trajectory computation of 2 quadcopters with collision checking between the two quadcopters. The planner at this stage computes a trajectory for the first quadcopter, and then, it proceeds to form the trajectory for the second quadcopter; as nodes are extended from the tree of the second quadcopter they are collision checked not only against obstacles but also against the first quadcopter trajectory. The nodes in the trees used for both quadcopters are given timestamps (Time from root node to the current node); in other words, when a full path is found, it is parameterized with time. This timestamp is based on the Δt that is used to propagate the dynamics every time the tree is expanded (i.e. The timestamp of a new node in the tree is equal to the timestamp of its parent plus the Δt that was used to propagate it). Thus, when a new node is added to the tree of the second quadcopter, it is checked against the node with the same timestamp from the trajectory of the first quadcopter for collision. From there, the planner was developed to include 3 quadcopters with nodes from the tree of the third quadcopter being checked against the trajectories of quadcopter 1 and quadcopter 2.

IV. ALGORITHM

The pseudocode shown here is for the 3 quadcopter trajectory planning. The psuedocode is divided into two sections below, one demonstrating the main section of the planner while the other demonstrates the kinodynamical RRT planner utilized by the main section of the planner. The same RRT planner is used for all 3 quadcopters, however, the difference lies in the collision checker used in each one.

A. Main Section Pseudocode

```
1: Quad1SubPath1 ← RRTQuad1($StartLocation1$, $IntermediateGoal1$)
2: $x_{start1}$ ← Quad1SubPath1.EndNode
3: Quad1SubPath2 ← RRTQuad1($x_{start1}$, $FinalGoal1$)
4: Path1 ← Append(Quad1SubPath1, Quad1SubPath2)
5: Quad2SubPath1 ← RRTQuad2($StartLocation2$, $IntermediateGoal2$)
6: $x_{start2}$ ← Quad2SubPath1.EndNode
7: Quad1SubPath2 ← RRTQuad2($x_{start2}$, $FinalGoal2$)
8: Path2 ← Append(Quad2SubPath1, Quad2SubPath2)
9: Quad3SubPath1 ← RRTQuad3($StartLocation3$, $IntermediateGoal3$)
10: $x_{start3}$ ← Quad3SubPath1.EndNode
11: Quad3SubPath2 ← RRTQuad3($x_{start3}$, $FinalGoal3$)
12: Path3 ← Append(Quad3SubPath1, Quad3SubPath2)
13: Return Path1, Path2, and Path3
```

The main section pseudocode above forms the trajectory for each quadcopter sequentially. For example, Lines 1-4 Form the trajectory for the first quadcopter. In Line 1, the subpath from the start location to the intermediate goal is computed. From there, the end node of subpath1 is used as the root for the tree that will grow from the intermediate goal to the final goal for the computation of subpath2. Then, subpath1 and subpath2 are appended. This process is repeated for the other two quadcopters. However, the collision checking is different for each quadcopter. As mentioned in section IV, nodes are given timestamps (When a full path is returned, it is traced with time). Thus, when, for example, quadcopter 2 propagates its nodes they are checked against a node with the same timestamp in the trajectory of quadcopter 1, if it exists.

B. RRT Pseudocode

```
1: Create a tree root at start location
2: While not at goal location
3:     $p$ ←GenerateRandomRealNumber(0,1)
4:     If $p$ is greater than $p_{goal}$
5:         $x_{rand}$ ←StateSample()
6:     else
7:         $x_{rand}$ ←GoalStateSample()
8:     $x_{new}$ ←nearest state (node) in the tree to $x_{rand}$
9:     for $m$ number of times
10:        $u$ ←SampleControlInput()
11:        $x_{extend}$ ←$x_{near}$ + $\int_0^t f(x(\tau), u)\, dx$
12:        if CollisionFree($x_{extend}$) and WithinConstraints($x_{extend}$)
13:            Add $x_{extend}$ to the candidate list
14:    $x_{new}$ ←$x_{extend}$ from the candidate list with the smallest distance to $x_{rand}$
15:    add $x_{new}$ to the TreeNodes list
16:    add edge between $x_{near}$ and $x_{new}$
17:    If WithinGoalRegion($x_{new}$)
18:        return path from root to $x_{new}$
```

Line 1: This line creates the first node (root) of the tree at the given location of the quadcopter.
Line 2: This line starts a while loop that will only terminate when a path has been found to the goal location (i.e. the tree has fully extended to the goal location)
Line 3: The variable p is set to a real number that is randomly generated (uniformly) between the values of 0 and 1
Line 4-7: This if statement is how the algorithm ensures that states from the goal location are sampled a percentage number of times to induce the goal bias. If p is greater than the goal bias p_{goal} (fraction) then x_{rand} is randomly sampled (uniformly). Otherwise, x_{rand} is set to a randomly sampled state that is within the goal region.
Line: 8 the variable x_{new} is set to the node (state) on the tree that is closest to x_{rand}.
Lines 9-14: This portion of the algorithm samples control inputs for a specified m number of times. From there it computes a Δx that it adds to x_{near}. This guarantees that the extending is subjected to the motion model. From there, it is checked if the extending is collision free and is within the kinematic constraints and if it is, the extension x_{extend} is added to a candidate list. From there, the state in the candidate list that is closest to x_{rand} is saved as x_{new} and the candidate list is emptied.
Line 15-16: Line 15 adds x_{new} to the list of all the nodes in the tree and Line 16 adds an edge between x_{near} and x_{new}.
Line 17-18: This portion of the algorithm checks whether x_{new} is within the goal region, and if it is, a path is returned from the root of the tree to x_{new}.

V. RESULTS

There were 3 workspaces that were made for the testing of the planner. The first workspace was a general design to evaluate how the planner would perform overall. The second workspace was designed for testing the planner's capability of dealing with the intermediate and final goal regions. The third workspace was designed for the testing of the planner's capability in returning paths with no trajectory collisions between the 3 quadcopters. 100 benchmark runs were performed for all the maps. The benchmark runs were primarily for the 3 quadcopter case rather than 1 quadcopter or 2 quadcopter as that inherently encompasses those cases. Furthermore, the parameters for the planner were tuned by investigating the behavior through trial and error, rather then, benchmark tests as there was not enough time to do so. Below are the results for each workspace. Each of the quadcopters was modeled as a square box of size 0.2 x 0.2 x 0.2. The value m detailed in Line 9 of RRT pseudocode was selected to be 10. A Δt of 0.5 seconds was used to propgate the dynamics. Moreover, euler's method was used for propgating the dynamics. The planner is evaluated through 4 metrics in the benchmark tests: success rate (Number of times a quadcopter independently reaches the intermediate and final goal), partial success rate (Number of times a quadcopter independently reaches at least 1 goal, intermediate or

final), path length, and computation time. Here is also a link for avi files of the motion of the 3 quadcopters in all 3 workspaces: https://drive.google.com/drive/folders/1dXbme519TNq3gNf5Ewo5JvsdKtTok9ql?usp=sharing
NOTE: The video speed is not actually representative of the realtime motion, it is just sped up due to how these videos were generated

A. Workspace 1

Setup:

- 15000 Nodes (Limit on number of nodes that can be used to find a solution)
- $\epsilon = 1$ (Spherical Goal Regions of Radius 1)
- $p_{goal} = 0.1$ (10% Goal Bias)

	Quadcopter 1	Quadcopter 2	Quadcopter 3
Start Location	(0,0,0)	(0,1,0)	(0,-1,0)
Int. Goal	(7,0,0)	(7,1,0)	(7,-1,0)
Final Goal	(12,0,0)	(12,0,0)	(12,0,0)

Table 1 (X, Y, Z) Start location and Goals Setup for each Quadcopter in Workspace 1

Solution:

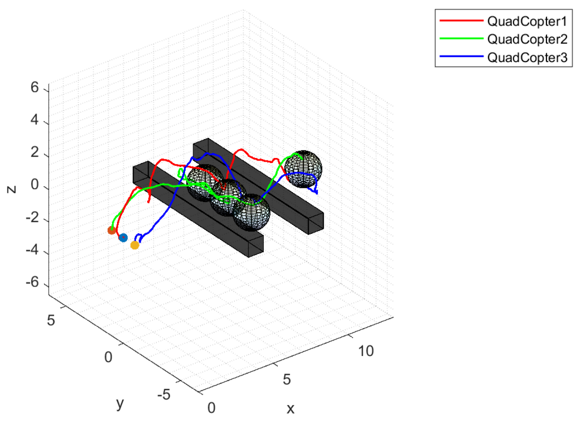

Fig. 1 View 1

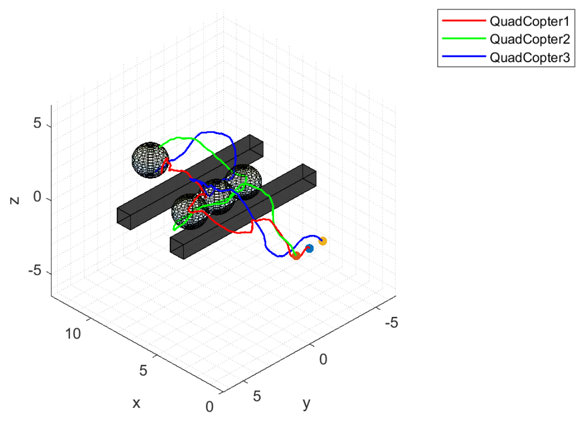

Fig. 2 View2

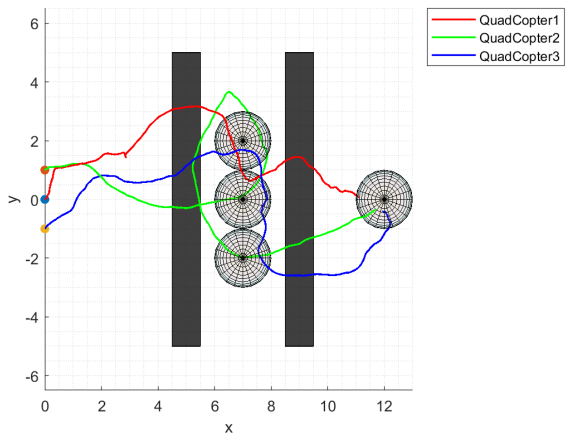

Fig. 3 View 3

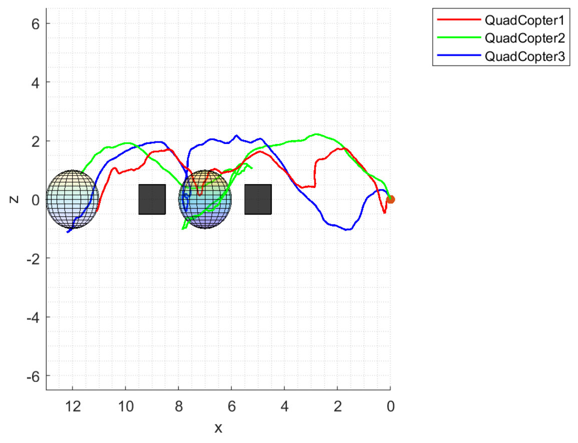

Fig. 4 View4

Benchmarking:

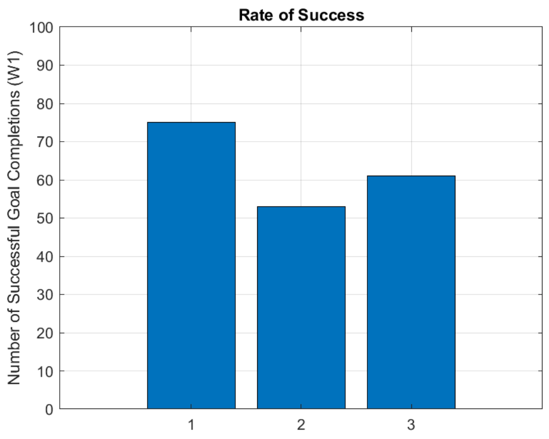

Fig. 5 Success Rate for Each Quadcopter

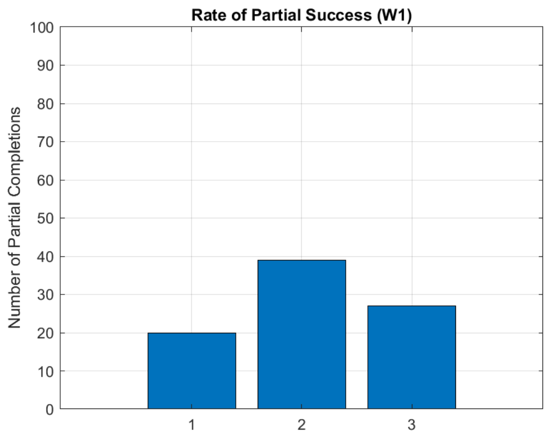

Fig. 6 Partial Success Rate for Each Quadcopter

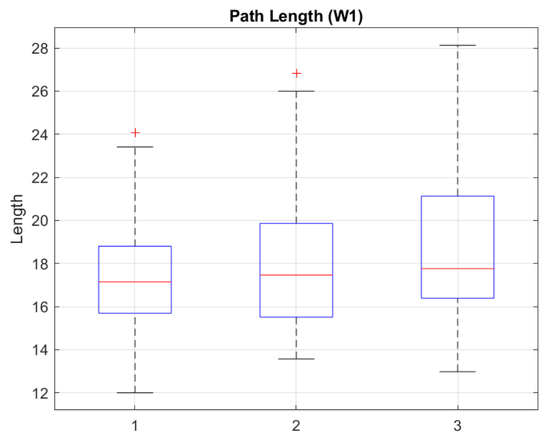

Fig. 7 Path Length for Each Quadcopter

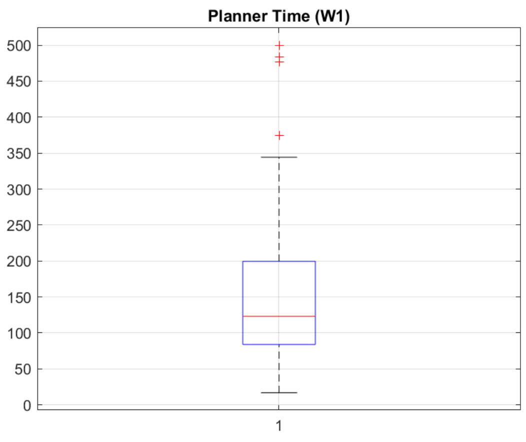

Fig. 8 Planner Overall Runtime (seconds)

Discussion:
The planner achieves 26 successful runs where all the quadcopters reach the intermediate goal and the final goal and 79 where each of the quadcopters reaches at least 1 goal. Quadcopter 1 appears to have the highest independent rate of success as seen in Figure 5 and that is expected as its trajectory is not dependent on that of the other 2 quadcopters. However, Quadcopter 2 has the highest independent partial success rate. Overall, Quadcopter 1 reaches at least 1 goal (intermediate or final) 95 times; Quadcopter 2 reaches at least 1 goal 92 times; Quadcopter 3 reaches at least 1 goal 88 times. Furthermore, all 3 quadcopters have very close Path Lengths trends as seen in Figure 7. The planner takes an average of 151 seconds (2.51 minutes) to complete the computation. The velocity of the goal states (intermediate and final) were set to be zero, however, the quadcopters only reach about a maximum of 10^{-2} order of magnitude for the velocity in each of the inertial directions; Quadcopter 1 has $(\dot{x}, \dot{y}, \dot{z})$ = (0.04, -0.015, 0.026) for the final goal; Quadcopter 2 has $(\dot{x}, \dot{y}, \dot{z})$ = (0.09, 0.064, -0.062); Quadcopter 3 has $(\dot{x}, \dot{y}, \dot{z})$ = (-0.05, 0.013, 0.028).

B. Workspace 2
Setup:
- 15000 Nodes
- $\epsilon = 1.5$
- $p_{goal} = 0.1$

	Quadcopter 1	**Quadcopter 2**	**Quadcopter 3**
Start Location	(1,0,0)	(0,1,0)	(0,-1,0)
Int. Goal	(6,0,0)	(0,6,0)	(0,-6,0)
Final Goal	(0,0,0)	(0,0,0)	(0,0,0)

Table 2 (X, Y, Z) Start location and Goals Setup for each Quadcopter in Workspace 2

Solution:

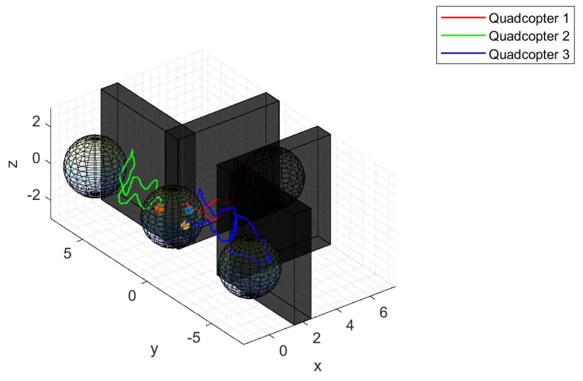

Fig. 9 View 1

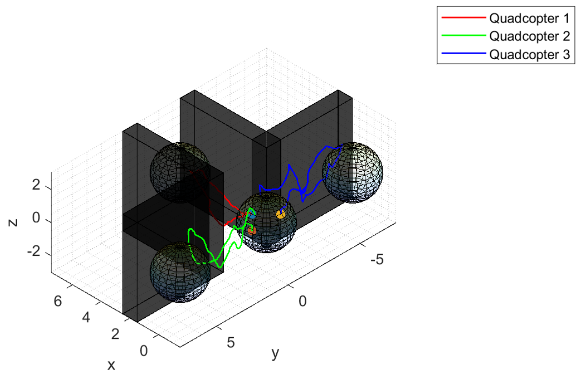

Fig. 10 View2

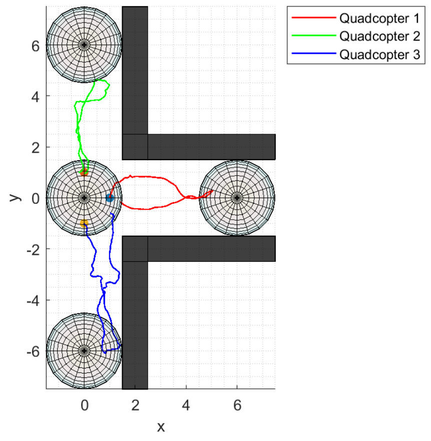

Fig. 11 View 3

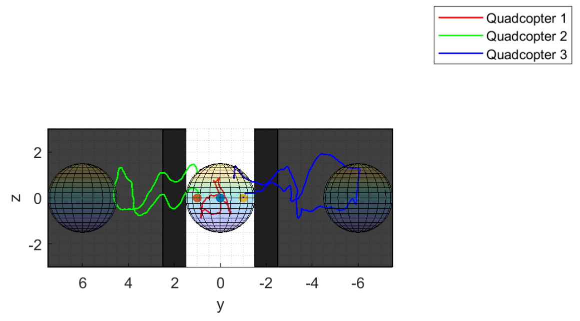

Fig. 12 View 4

Benchmarking:

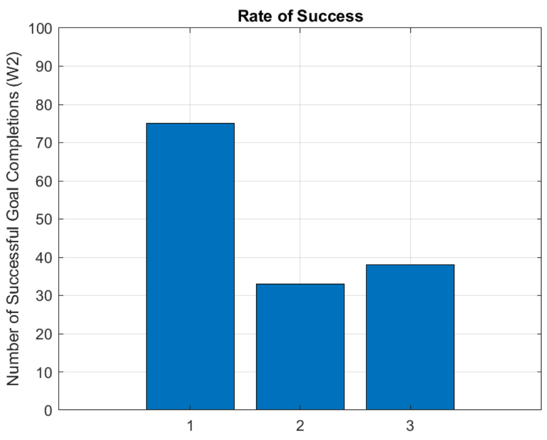

Fig. 13 Success Rate for Each Quadcopter

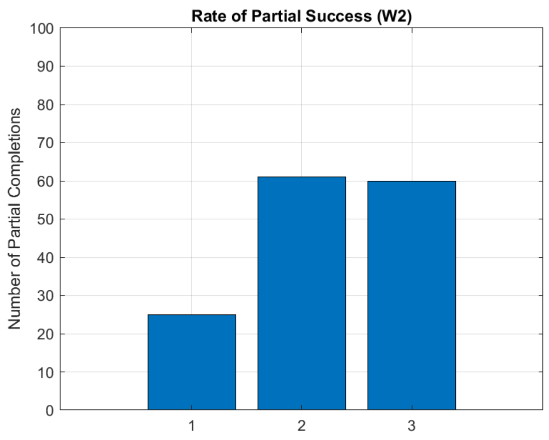

Fig. 14 Partial Success Rate for Each Quadcopter

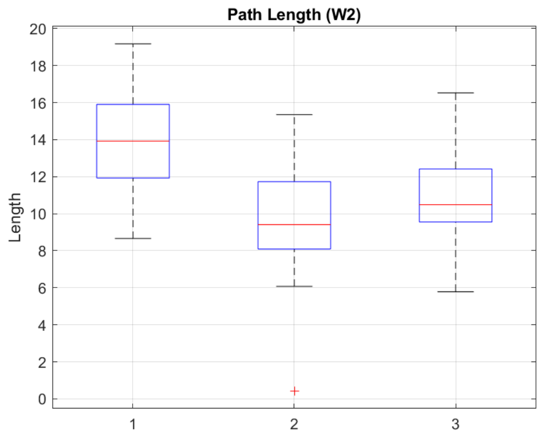

Fig. 15 PathLength for Each Quadcopter

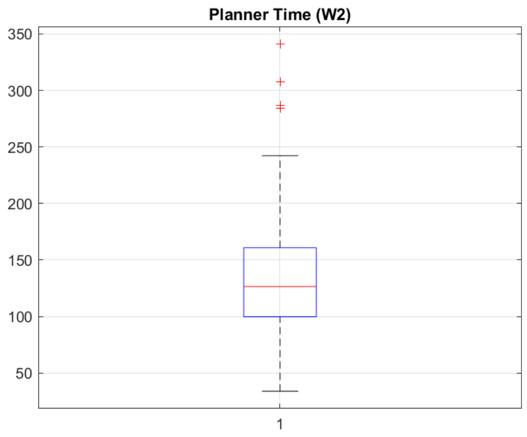

Fig. 16 Planner Overall Runtime (seconds)

Discussion:
The planner only achieves 8 successful runs (Intermediate and final goals reached for all quadcopters) for this workspace and 92 successful runs where all the quadcopters reach at least 1 goal. Again, Quadcopter 1 has the highest independent success rate (75 times) as seen in Figure 13. Overall, Quadcopter 1 reaches at least 1 goal 100 times; Quadcopter 2 reaches at least 1 goal 94 times; Quadcopter 3 reaches at least 1 goal 98 times. Quadcopter 2 and quacopter 3 have path length trends that are close, while Quadcopter 1 has a path length trend that is higher. The planner has an average of 134 seconds (2.2 minutes) for the computation in this workspace. Quadcopter 1 has $(\dot{x}, \dot{y}, \dot{z})$ = (0.036, 0.0633, 0.044) for the final goal; Quadcopter 2 has $(\dot{x}, \dot{y}, \dot{z})$ = (-0.033, 0.03, -0.059); Quadcopter 3 has $(\dot{x}, \dot{y}, \dot{z})$ = (-0.0039, -0.013, -0.064).

C. Workspace 3

Setup:
- 15000 Nodes
- $\epsilon = 1$
- $p_{goal} = 0.1$

	Quadcopter 1	Quadcopter 2	Quadcopter 3
Start Location	(0,0,0)	(0,0.75,0)	(0,-0.75,0)
Int. Goal	(9,0,0)	(9,0,0)	(9,0,0)
Final Goal	(0,0,0)	(0,0,0)	(0,0,0)

Table 3 (X, Y, Z) Start location and Goals Setup for each Quadcopter in Workspace 2

Solution:

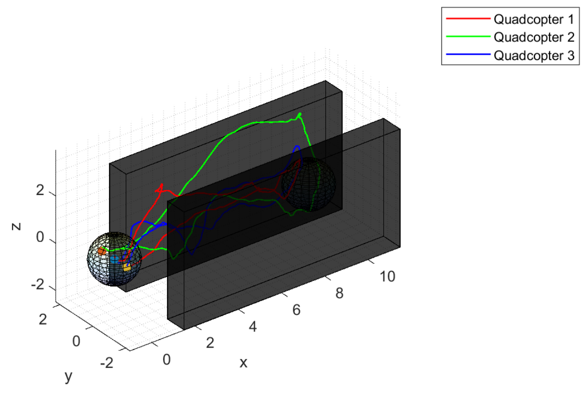

Fig. 17 View 1

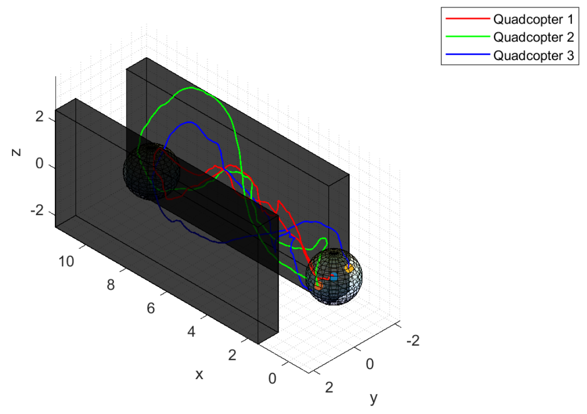

Fig. 18 View 2

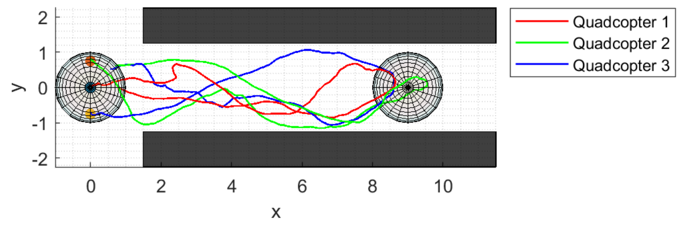

Fig. 19 View 3

Benchmarking:

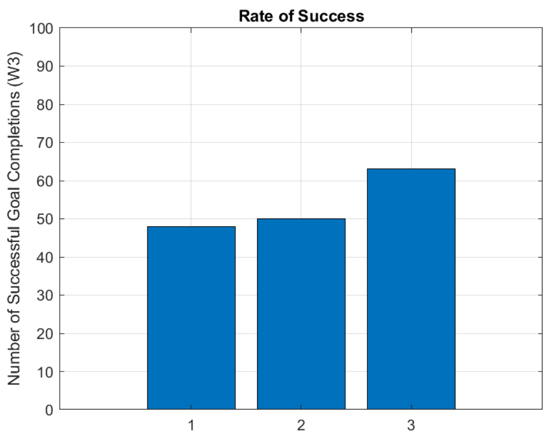

Fig. 20 Success Rate for Each Quadcopter

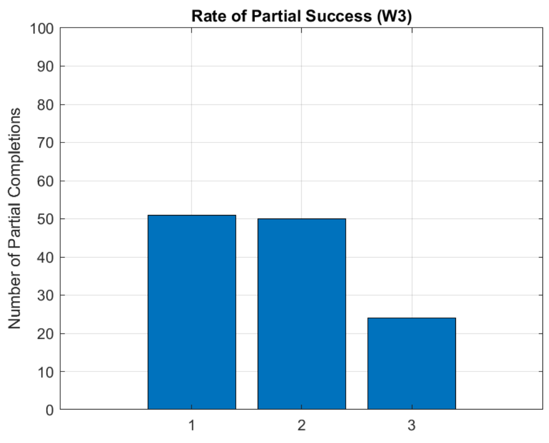

Fig. 21 Partial Success Rate for Each Quadcopter

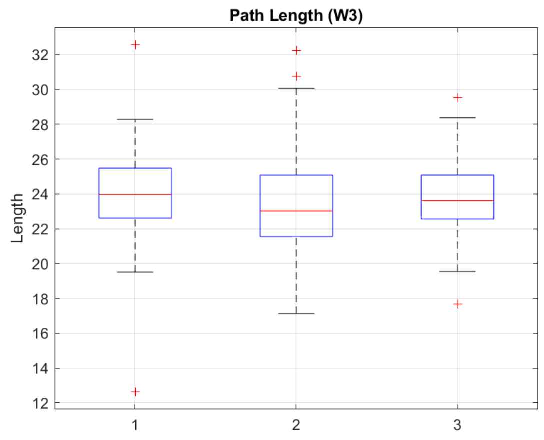

Fig. 22 PathLength for Each Quadcopter

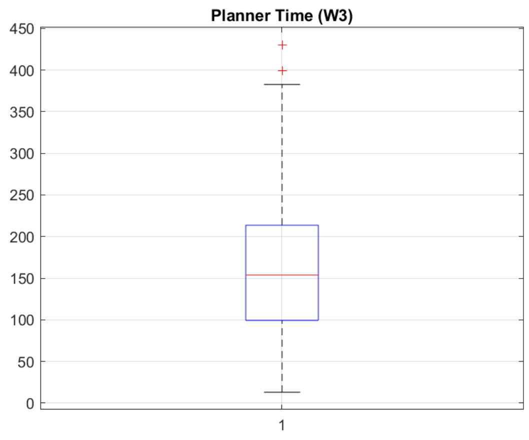

Fig. 23 Planner Overall Runtime (seconds)

Discussion:
The planner only achieves 17 successful runs for this workspace (Intermediate and final goals reached for all quadcopters) and 97 successful runs where all the quadcopters reach at least 1 goal. Quadcopter 1 has an independent success rate of 48 while quadcopter 2 and quadcopter 3 have rates of 50 and 63, respectively; The performance of quadcopter 1 is lower than that of quadcopter 2 and 3 for this workspace. Overall, Quadcopter 1 reaches at least 1 goal 99 times; Quadcopter 2 reaches at least 1 goal 100 times; Quadcopter 3 reaches at least 1 goal 98 times. The Path length trend is similar for all quadcopters as seen in Figure 22. The planner has an average of 166 seconds (2.77 mins) for the computation time in this workspace. Quadcopter 1 has $(\dot{x}, \dot{y}, \dot{z})$ = (-0.086, 0.005, 0.082) for the final goal; Quadcopter 2 has $(\dot{x}, \dot{y}, \dot{z})$ = (-0.1, -0.008, 0.018); Quadcopter 3 has $(\dot{x}, \dot{y}, \dot{z})$ = (-0.05, -0.003, 0.032).

VI. CONCLUSION

The planner does not have a proper success rate for returning a solution for all 3 quadcopters to reach the intermediate and final goals. It does, however, appear to function better with only 1 goal defined for all 3 quadcopters. Furthermore, the computation time of the planner is not optimal; This could be a result of the closest node computation in RRT or the general design structure of the implementation and the method by which multiple trajectory collision detection is computed. The computation time does continue to increase with larger maps. Other maps had been constructed which were more complex and larger; however, due to time limitations, benchmarking tests and further examinations were not possible in the time frame of this project. The focus was primarily on the 3 simple workspaces shown in the results section to understand the fundamental behavior of the planner. Additionally, the velocity of the quadcopters at the final goal are in the ranges of 10^{-2} order of magnitude. This could be further improved by using a better distance metric as the distance metric used here was the norm of the state vector. Moreover, different ways of forming the 3 trajectories would have been investigated such as using a different base planner (SST) or using a joint state space (Formalizing the state space differently).

VII. Appendix

A. Subproblem 1: 1 Quadcopter

NOTE: Only workspace 1 was used to test this case and no benchmarking was performed.

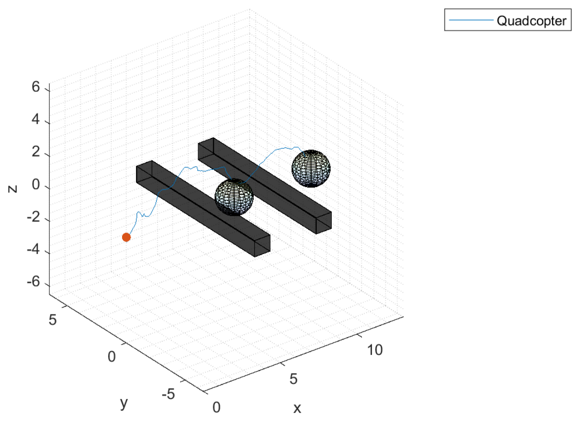

Fig. 24 View 1

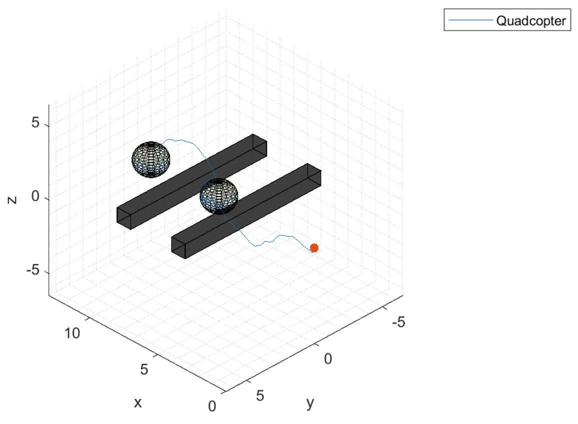

Fig. 25 View 2

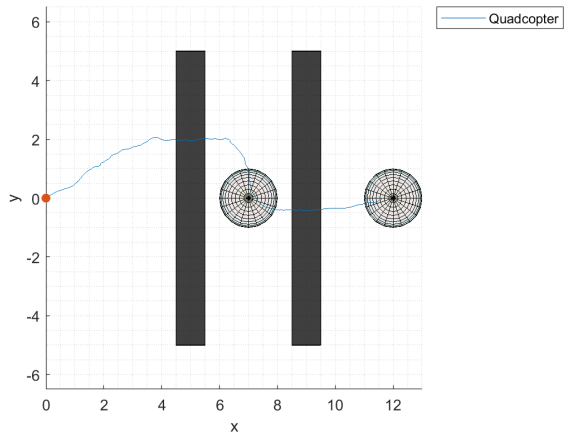

Fig. 26 View 3

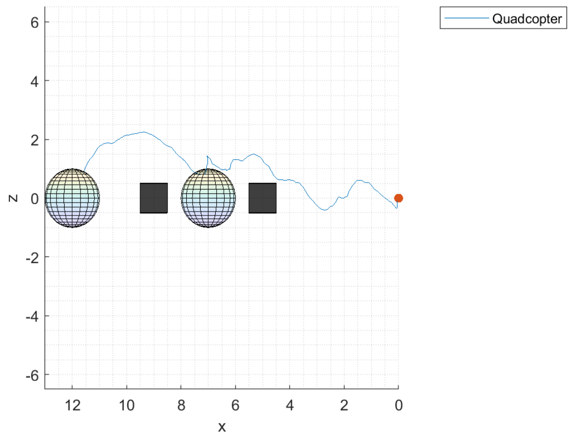

Fig. 27 View 4

B. Subproblem 2: 2 Quadcopters

NOTE: Only workspace 1 was used to test this case and no benchmarking was performed.

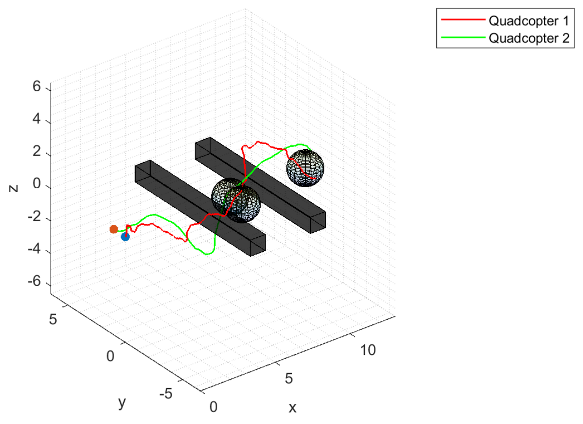

Fig. 28 View 1

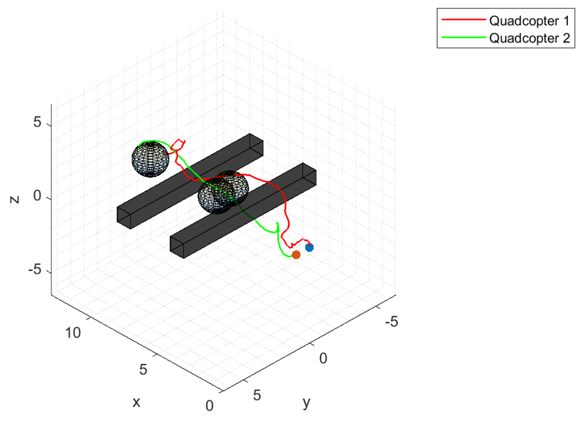

Fig. 29 View 2

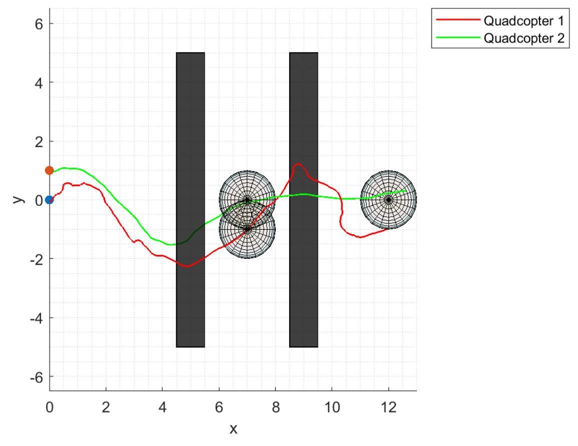

Fig. 30 View 3

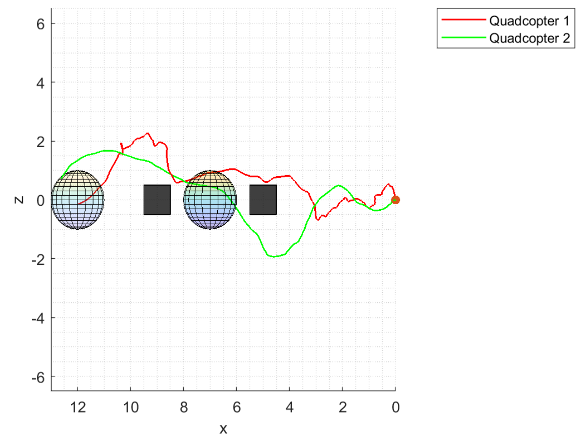

Fig. 31 View 4

C. Dynamics Document

Multirotor Aircraft Project

ASEN 5014 - ASEN 5044

November, 2016

1 Introduction

Multirotors are versatile vehicles with a wide variety of applications. They also represent an interesting control problem as they are inherently unstable. The vehicle is controlled by commands that alter the trust produced by the rotors.

2 Physical system

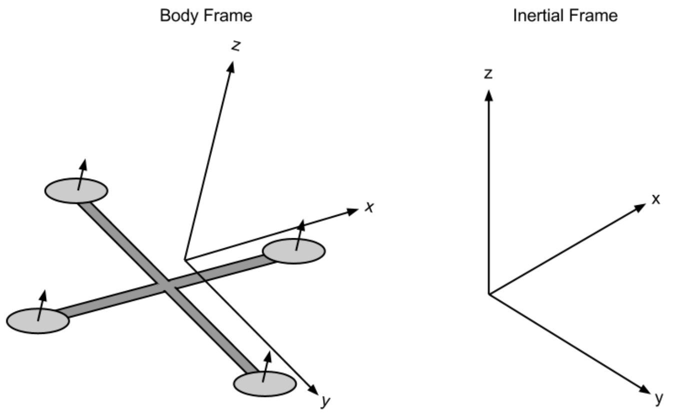

Figure 1: Quadcopter Coordinate Frames

3 Linearized State Space Model

Assuming a symmetric vehicle with four identical rotors, where one pair spins in one direction and the other pair spins in the opposite direction, the following linearized model can be derived for motion in the neighborhood of level hovering flight.

State variables in this model are

$$\mathbf{x} = \begin{bmatrix} x & \dot{x} & y & \dot{y} & z & \dot{z} \end{bmatrix} \tag{1}$$

consisting of inertial positions and velocities in the X, Y, and Z directions. It is assumed that an inner loop control is acting to keep the vehicle in a horizontal orientation, and to keep the body x axis pointing in the X direction. It also assumed that the rotor normals are all tilted in toward the center of the vehicle at a fixed angle, so that a component of the thrust in each is available for horizontal translation. There are four rotor thrust controls (inputs) T_1 through T_4. These produce perturbation forces (away from the nominal needed to oppose the vehicle's weight) as follows.

$$F_X = \delta(T_3 - T_1) \tag{2}$$
$$F_Y = \delta(T_4 - T_2) \tag{3}$$
$$F_Z = \gamma(T_1 + T_2 + T_3 + T_4) \tag{4}$$

The motion of the vehicle is retarded by aerodynamic drag, using the $k_{d,lat}$ and $k_{d,vert}$ coefficients in units of $[\frac{N \cdot s}{m}]$. Vehicle mass is given by m. Outputs are the X, Y, and Z positions, given by GPS.

The resulting state space realization

$$\dot{\mathbf{x}} = \mathbf{Ax} + \mathbf{Bu} \tag{5}$$
$$\mathbf{y} = \mathbf{Cx} + \mathbf{Du} \tag{6}$$

has the specific form:

$$\mathbf{A} = \begin{bmatrix} 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & -\frac{k_{d,lat}}{m} & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & -\frac{k_{d,lat}}{m} & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & -\frac{k_{d,vert}}{m} \end{bmatrix} \tag{7}$$

$$\mathbf{B} = \begin{bmatrix} 0 & 0 & 0 & 0 \\ -\frac{\delta}{m} & 0 & \frac{\delta}{m} & 0 \\ 0 & 0 & 0 & 0 \\ 0 & -\frac{\delta}{m} & 0 & \frac{\delta}{m} \\ 0 & 0 & 0 & 0 \\ \frac{\gamma}{m} & \frac{\gamma}{m} & \frac{\gamma}{m} & \frac{\gamma}{m} \end{bmatrix} \tag{8}$$

$$\mathbf{u} = \begin{bmatrix} T_1 \\ T_2 \\ T_3 \\ T_4 \end{bmatrix} \tag{9}$$

Using parameters from a particular vehicle, we get the following values for the state space realization:

$$\dot{\mathbf{x}} = \begin{bmatrix} 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & -0.0104 & 0 & 0 & 0 & \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & -0.0104 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & -0.0208 \end{bmatrix} \mathbf{x} + \begin{bmatrix} 0 & 0 & 0 & 0 \\ -0.04167 & 0 & 0.04167 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & -0.04167 & 0 & 0.04167 \\ 0 & 0 & 0 & 0 \\ 0.4 & 0.4 & 0.4 & 0.4 \end{bmatrix} \mathbf{u}$$

$$\mathbf{y} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \end{bmatrix} \mathbf{x} + \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} \mathbf{u}$$