Improving UAS Route Efficiency While Maintaining Higher Velocity

Cadet Tyler Harmon
United States Military Academy
West Point, New York
tyler.harmon@westpoint.edu

Dominic Larkin
United States Military Academy
West Point, New York
dominic.larkin@westpoint.edu

Christopher Korpela
United States Military Academy
West Point, New York
christopher.korpela@westpoint.edu

***Abstract*—This paper details a method of improving the accuracy with which a multi-rotor unmanned aerial system (UAS) can maintain a trajectory en route to pre-designated waypoints. Although most multi-rotor UAS's are nearly holonomic, variables in flight conditions such as wind speed and elevation changes make maintaining a set trajectory difficult. Here a hexcopter is used, and intermediate waypoints generated using Dubins vehicle kinematics are injected into the hexcopter's flight controller pre-flight. This allows the hexcopter to utilize its existing flight controller in a more efficient way, resulting in a flight trajectory with a measurable increase in accuracy and repeatability. Additionally, treating a multi-rotor UAS as a Dubins vehicle allows for a trajectory that passes through waypoints at speed and in a non-linear fashion, decreasing risk to the vehicle from ground forces.**

I. Introduction

Many Department of Defense (DoD) agencies seek to surveil of an Area of Operations (AO) to enable more intelligent decisions in the AO. Unmanned Aerial Vehicles (UAV's) are commonly used for high-level sweeps of a large geographic area and the identification of areas to be further surveiled. Flying a UAV to survey these identified points is expensive and timely, and often requires the UAV to fly at a lower elevation to complete this task. Therefore, multi-rotor UAS's are useful for the investigation of designated waypoints, because they are less expensive and more maneuverable. However, because of the smaller mass of a multi-rotor UAS, environmental factors tend to disrupt a flight trajectory more than they would for a larger UAV. Thus, the problem exists of how to best plan a trajectory that a multi-rotor UAS can maintain accurately, as opposed to a simple linear flight between waypoints.

Additionally, stopping in place to hover and turn at each waypoint puts a multi-rotor UAS at risk of destruction from enemy ground forces. Strictly linear flight near a waypoint also puts a multi-rotor UAS at risk of destruction because of the predictability of the flight trajectory. These considerations together frame the need for a flight trajectory that passes through a waypoint at speed and in a non-linear fashion, thus mitigating two significant risks.

Manuscript received December 20, 2019. This work is supported by the Defense Threat Reduction Agency and the Ground Vehicle Systems Center. The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements, either expressed or implied, of the U.S. Department of Defense or the U.S. Government.

Fig. 1: The hexcopter test route.

II. Related Work

Current research on efficient coverage and trajectory planning for Unmanned Aerial Systems (UAS) tends to be overly focused on simulation in predictable environments, while field robotics researchers are well versed in the challenges presented by nature in real-world dynamic environments [1]. For example, trajectory generation and control for most multi-rotor UAS's requires advanced closed-loop control, and is usually run on a given flight controller [2]. These flight controllers are designed and tuned for operation in ideal environments with platforms that remain constant in terms of size, weight, and payload. However, to be able to adapt

to changing payloads and environmental conditions, it is necessary to implement augmentation to the basic Proportional-Integral-Derivative (PID) controller to allow efficient waypoint following without having to retune PID coefficients for each mission or delivery [3].

III. System Design

The experimental setup in this study uses a six-rotor drone of dimensions 0.45m W, 0.45m L, and .32m H. The mass of the drone is 4.5kg, and there is an onboard Raspberry Pi 3 running ArduPilot.

The test route is a triangular path situated over a field at West Point. The coordinate waypoints of the route are as follows:

{41.39069200, -73.95326400}
{41.39066840, -73.95325340}
{41.39093800, -73.95339490}
{41.39167450, -73.95281020}
{41.39083940, -73.95320450}
{41.39069180, -73.95326080}

Given the above waypoints, ArduPilot will fly along the following route, connecting the waypoints together linearly and pausing at each designated waypoint to turn in place.

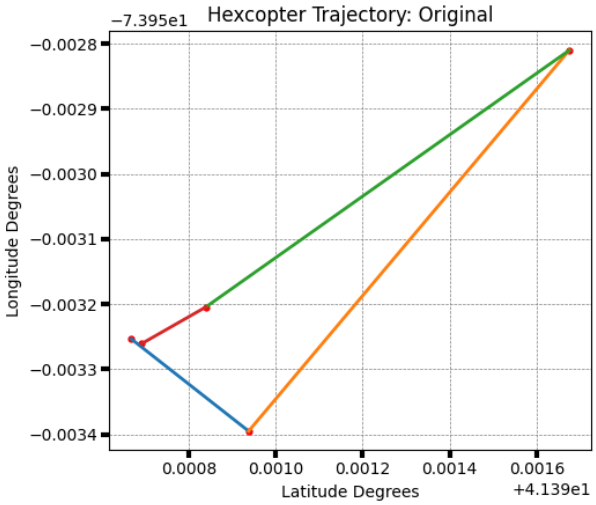

Fig. 2: The hexcopter test route, as plotted in PyPlot from MatPlotLib.

IV. Software Infrastructure

A dubins path library from PyPi [4] in conjunction with numerical Python from SciPy (NumPy) [5]was used to generate Dubins path between waypoints given a constrained turning radius. MatPlotLib's PyPlot [6] was used to graph results and verify the correctness of the intermediate point generation. The Dubins path library was able to interpolate between two points given a constrained position and heading, beginning three times the turning radius away from the original waypoint.

The pseudo code for the algorithm to generate intermediate waypoints is as follows.

For each pair of waypoints (w1, w2) in the range (1 to n):
Calculate the direct slope between w1 and w2:
angle = math.atan((w2.y - w1.y) / (w2.x - w1.x))
* Generate the first intermediate trajectory leg (Dubins path) beginning at (w1.x, w1.y) and terminating at (w1.x + (3 * turnRad * math.cos(angle)), w1.y + (3 * turnRad * math.sin(angle))
* Record the halfway point and the terminating point of the first intermediate trajectory leg as new waypoints
* Generate the second intermediate trajectory leg (Dubins path) beginning at (w2.x - (3 * turnRad * math.cos(angle)), w2.y + (3 - turnRad * math.sin(angle)) and terminating at (w2.x, w2.y)
* Record the initial point and the halfway point of the second intermediate trajectory leg as new waypoints
Return the new waypoints in a format readable by the ArduPilot control software.

This algorithm generates four intermediate waypoints between each desired pair of original waypoints. These intermediate waypoints are given to the ArduPilot flight controller.

Assuming a constant flight speed and constant turning radius, calculating a Dubins path for the waypoints given in part III produces the following trajectory.

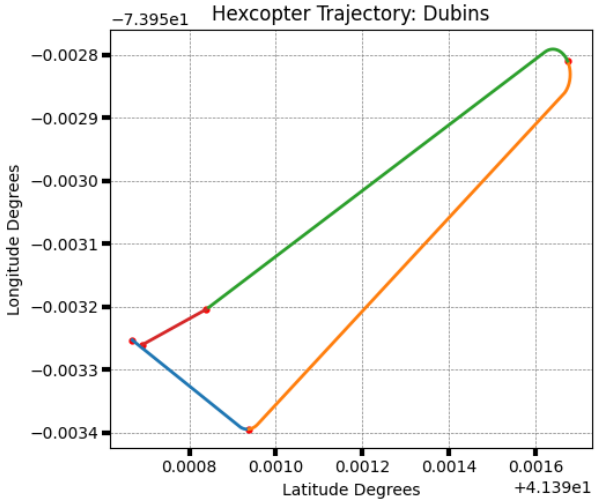

Fig. 3: The hexcopter test route connecting Dubins path segments between the waypoints.

The minimum distance necessary to generate a trajectory that is not forced to create loops in its path is approximately three times the turning radius of the hexcopter. Taking this into consideration allows the flight trajectory to be further refined such that a linear flight path is maintained between waypoints until it is necessary to begin a turn.

This method creates two intermediate waypoints around each turn, which increases the total number of waypoints in the original trajectory from 7 to 15. The new waypoints are fed back into the ArduPilot control software.

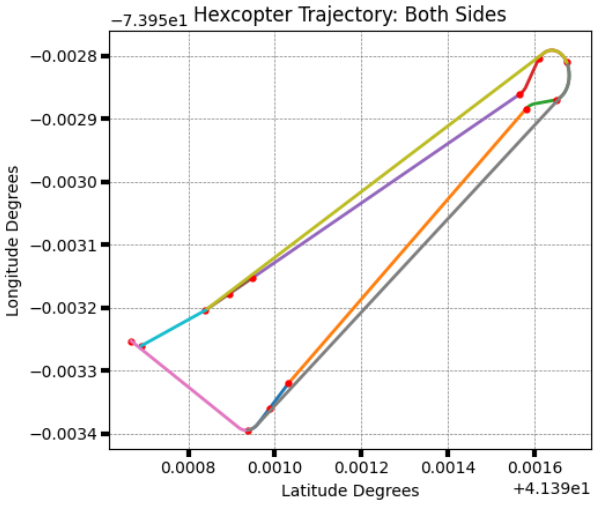

Fig. 4: The hexcopter test route connecting Dubins path segments between the waypoints.

V. Testing

Data is to be gathered using the ArduPilot system aboard the hexcopter described in section III. The hexcopter will be flown at three different speeds within its physical capability: 5 m/s, 10 m/s, and 14 m/s.

After collecting data for these speeds, an equation will be fit to the data to map speed to turning radius.

After gathering this GPS data, a python script was used to calculate the total sum of squared error during the path, the time on waypoint (ToW), and round trip time (RTT). The success condition is a decreased ToW or RTT which results in a subsequent increase in drone safety. Some analysis of the route will be qualitatively performed by visual comparison to determine if the adjusted route is correctly generated.

VI. Results

Due to the remainder of the United States Military Academy's semester AY20-2 being taught remotely because to COVID-19 considerations, testing conditions are restricted. The intermediate waypoints for the path shown in part I have been generated in the ArduPilot control format, but have not been tested.

VII. Conclusions

UAS's are valuable for their reconnaissance capabilities, but stopping in place or moving predictably can put them at risk of being destroyed. To combat this and to mitigate the effects of environmental factors like wind, the generation of intermediate waypoints has been able to decrease time on waypoint.

Future work should focus on modeling vehicle kinematics for a multi-rotor UAS to determine an equation for minimum turning radius as a function of speed. This would allow the algorithm described in part IV to be more flexible, allowing for different segments in the trajectory to be constrained by different curvatures depending on the desired flight speed in that segment.

VIII. Acknowledgments

The authors would like to thank Army Reasearch Laboratory - Sensors and Electron Devices Directorate (ARL-SEDD) for funding this work.

References

[1] N. Karapetyan, J. Moulton, J. S. Lewis, A. Q. Li, J. M. O'Kane, and I. Rekleitis, "Multi-robot dubins coverage with autonomous surface vehicles," in *2018 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2018, pp. 2373–2379.

[2] M. Hehn and R. D'Andrea, "Quadrocopter trajectory generation and control," *IFAC proceedings Volumes*, vol. 44, no. 1, pp. 1485–1491, 2011.

[3] A. Alkamachi and E. Erçelebi, "Modelling and genetic algorithm based-pid control of h-shaped racing quadcopter," *Arabian Journal for Science and Engineering*, vol. 42, no. 7, pp. 2777–2786, 2017.

[4] A. Walker, "Python wrapper of the C version of the Dubins-Curves library," 2018. [Online]. Available: https://github.com/AndrewWalker/pydubins

[5] P. Virtanen, R. Gommers, T. E. Oliphant, M. Haberland, T. Reddy, D. Cournapeau, E. Burovski, P. Peterson, W. Weckesser, J. Bright, S. J. van der Walt, M. Brett, J. Wilson, K. Jarrod Millman, N. Mayorov, A. R. J. Nelson, E. Jones, R. Kern, E. Larson, C. Carey, İ. Polat, Y. Feng, E. W. Moore, J. Vand erPlas, D. Laxalde, J. Perktold, R. Cimrman, I. Henriksen, E. A. Quintero, C. R. Harris, A. M. Archibald, A. H. Ribeiro, F. Pedregosa, P. van Mulbregt, and S. . . Contributors, "SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python," *Nature Methods*, vol. 17, pp. 261–272, 2020.

[6] J. D. Hunter, "Matplotlib: A 2d graphics environment," *Computing in Science & Engineering*, vol. 9, no. 3, pp. 90–95, 2007.