Mustard Yellow: Why Walk When You Can Fly?

Autonomous Navigation using SLAM / PTAM-based Mapping

By Michael Eve, ECE '21

Introduction

In the last decade, research being done on autonomous MAVs for hobbyists, as well as businesses, has evolved how we think about travel, delivery systems, and ease of access. There is still much work being done to find a happy medium between the testing of larger MAVs like the AR 2.0 in educational settings and the cost that can incur when purchasing a drone, which isn't cheap if you're trying to test computer vision for payload-carrying drones.

Figure 1 – Original Modification to Parrot AR 2.0

A tradeoff we'll investigate briefly will be the idea of onboard processing versus a 'ground control' approach, as the methods for parallel programming, image processing, and tracking of the MAV itself are all very labor intensive on a 32-bit 468MHz ARM9 RISC with only 128MB DDR-RAM running at 200MHz [1]. This processing chip just maintains the basic functions of the AR 2.0, including the stabilization functionality of the quadcopter. With this in mind, we're focusing more on the 'ground-control' approach rather than an onboard processing implementation. This is the more ideal approach since there will need to be a more limited flight range for testing and there is limited opportunity to collaborate in a more hands-on environment, which makes this software-driven implementation more attractive.

Related Work

The work that has been done in PTAM and SLAM implementation has had a major focus in outdoor versus indoor testing. The indoor experiments are mainly focused motor control and stabilization in 'ideal' environments. The outdoor implementations are more focused on the software's ability to estimate pose, or the given position of an aircraft at a given moment.

The work done for both PTAM and SLAM navigation also focuses on GPS-denied environments [1]. This is important to note for testing reasons because the complex system involving the timing of a both a GPS signal and a PTAM/SLAM application would be very difficult to implement with the scope of our project.

We will cover how this is done using pre-defined mapping techniques and more dynamic parameters for tracking the position of the MAV.

AR Hardware

The platform that is the AR drone's built-in Flight controller and IMU (inertial measurement unit) is fairly limited in terms of programming accessibility. Even though this was concerning at first glance, the cost of the quadcopter was free since it was provided

by the ECE department. Additionally, as opposed to more modern models, the AR 2.0 offers a safer hull for indoor testing as well as an outdoor hull for more maneuverability.

The method for communicating with the AR 2.0 is through WIFI. The handshake done between the router on the drone and the laptop running our program.

The AR 2.0 came equipped with a 3-axis gyroscope and accelerometer, an ultrasound altimeter and two cameras. The primary camera being used for this implementation is the front camera, which gives our program a first-person view (FPV) of the drone's flight. The downwards facing camera gives us a perpendicular view and can also be toggled during flight.

Figure 2 – Second Modified Drone design + payload

Navigation Techniques

The PTAM navigation software being used has a few advantages as well as disadvantages. PTAM works best in indoor environments that have a lot of "structure" from the FPV of the AR drone [2]. Structure being cabinets, desks, furniture, polygonal objects, varying textures etc. PTAM does not navigate the drone well with different obstacles such as trees or other outdoor obstacles that have a less standardized shape, especially when this environment is windy. This violates what we would call the "static world" assumption, which states that to properly perform PTAM, every object that is being tracked in order to orient the drone should not be moving.

PTAM is required to implement our SLAM software for figure flying because it provides a basis of intelligence such that the drone can learn from an environment it has never seen before, create visual key points to identify, and stabilize itself when pushed off a fixed coordinate.

The SLAM navigation software being used will alleviate some of the disadvantages that come with purely PTAM-based navigation. SLAM allows us the freedom to program a pre-specified flight plan on the AR 2.0. The main challenge that we will face with SLAM is the scale of the map (flight plan) that we define and initialize for the drone.

The map must be estimated not only from cameras, but the additional sensors on the AR 2.0 such as the IMU (inertial Measurement Unit), accelerometer, gyroscope, ultrasound altimeter. We will be using the data provided by the IMU to estimate the pose of the Parrot AR. To be specific, the downwards camera combined with the IMU data will be used to estimate current velocity of the drone to best guide the drone's trajectory such that it doesn't overshoot the parameters of the map, but also completes the map in a fast enough time.

SLAM is the ideal approach in this project because it uses the tracking and mapping algorithm that PTAM utilizes to help the AR 2.0 track itself as well as *key points* in its area of vision. SLAM uses this tracking methodology to acknowledge the drone's presence in a space, then manages to eliminate drifting to keep the drone as close to the map it has been given to traverse.

Our Implementation

In our design of the ROS (Robot Operating System) environment, the required version of ROS that needed to be used was Indigo. This distribution of ROS allowed us to run Ubuntu 14.04.6 LTS, which is an operating system capable of running the software that PTAM/SLAM navigation is based on and has been tested in. When the environment was properly set up, the output of the drone's motor speed, the status of the drone (landed, flying, unknown), and the commands being sent to the drone can all be

captured by the graphical user interface (GUI). This GUI can manipulate the drone manually as well as send a pre-defined flight plan for the drone to follow.

The main objective with our strategy is to have the drone pilot itself with a given flight plan from one location to another. It will be given discrete points to slow down, stop, turn, and even land. The beauty of the PTAM/SLAM software running on the drone is that it will correct the drifting of the drone off the flight plan it is given. While this 'error' isn't very large, a significant enough change in the drone's position can make it difficult for the drone to regain control of itself and continue to follow the flight plan.

Some issues that have come up that prevent the drone from performing as well are environmental factors like wind, cold temperature (faster battery drain) and a poorly structured space to fly in. The wind would push the drone far enough from its flight plan that the path would be difficult to recover, and the drone would need to land. The cold temperature would make the battery drain faster thus making the power of the drone's motors not as strong. In turn, this would make the correction of the drone not as good. The significance of the space we're flying in is that if there is a table on one side of the drone, or it is unevenly surrounded, then the wind that the drone pushes beneath itself will travel up the surrounding object and push down the side of the drone it is closest to, thus preventing a balanced takeoff.

Conclusion

Through the successful implementation of the flight path, a payload will be attached to the bottom of the drone. This payload has been designed to sit at the center of gravity of the drone. The successful flight of the drone will trigger a releasing mechanism (solenoid) to let the hatch of the box drop, thus releasing the payload inside. The main issue we may face would be the weight's effect on the drone's ability to stabilize and navigate itself through the given flight plan. We've mitigated this by decreasing the overall length of the landing supports. This proved to be the most effective solution since it was the most flexible part of our design process. We look forward to refining the navigation metrics to get the most out of the drone's performance while insuring a safe and stable flight for the payload design.

References

[1] J. Engel, J. Sturm, and D. Cremers, "Accurate Figure Flying with a Quadrocopter Using Onboard Visual and Inertial Sensing,".

[2] J. Engel, J. Sturm, and D. Cremers, "Camera-based navigation of a low-cost quadrocopter," in *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, Vilamoura-Algarve, Portugal, Oct. 2012, pp. 2815–2821, doi: 10.1109/IROS.2012.6385458.

[3] J. Engel, J. Sturm, and D. Cremers, "Scale-Aware Navigation of a Low-Cost Quadrocopter with a Monocular Camera," *Robotics and Autonomous Systems (RAS)*, vol. 62, 2014.

[4] J. Engel, *Computer Vision Group - Software - Visual Navigation for the Parrot AR.Drone (ROS Package)*, 08-Mar-2016. https://vision.in.tum.de/data/software/tum_ardrone

[5] "Parallel Tracking and Mapping for Small AR Workspaces (PTAM)." http://www.robots.ox.ac.uk/~gk/PTAM/

[6] D. Mellinger and V. Kumar, "Minimum snap trajectory generation and control for quadrotors," in *2011 IEEE International Conference on Robotics and Automation*, May 2011, pp. 2520–2525, doi: 10.1109/ICRA.2011.5980409.

[7] A. Carrio, J. Tordesillas, S. Vemprala, S. Saripalli, P. Campoy, and J. P. How, "Onboard Detection and Localization of Drones Using Depth Maps," *IEEE Access*, vol. 8, pp. 30480–30490, 2020, ACCESS.2020.2971938.

[8] "tum_ardrone/drone_stateestimation - ROS Wiki." https://wiki.ros.org/tum_ardrone/drone_stateestimation#Tips_for_good_PTAM_and_Scale_Estimation_Performance

[9] S. Fraundorfer, "Odometrey_lecture_CS_toronto." University of Toronto: Institute for Aerospace Studies , Toronto, 09-Feb-2016.