Micro-UAV Self-Characterization

Claire Beery, Greg Coleman, Sarah Walters

16 February 2015

1 Goal/Objective Statements

The purpose of this project is to explore self-characterization of unmanned aerial vehicles (UAVs) by executing and characterizing simple maneuvers in order to enable a micro-UAV to perform more complicated maneuvers. Our UAV will operate in an xy coordinate system which lies in a plane parallel to the ground. The UAV should be able to execute linear maneuvers in the independent x- and y-directions in order to characterize its flight in the plane, then combine the orthogonal movements in order to perform more complex maneuvers within the plane. The complex maneuvers will include diagonal lines, ellipses, and letters.

It is currently possible for UAVs to learn single maneuvers by executing them repeatedly and tuning performed trajectory in comparison to desired trajectory. However, single learned maneuvers provide extremely limited capability; a UAV would be far more flexible if it were able to learn a small number of simple maneuvers and then perform a wide variety of more complex ones autonomously.

2 Definitions

Below is a list of terms we will be using to discuss the project:

trajectory - the position and velocity of the UAV over time (either desired or performed)
player - the UAV (real or simulated); in our case, this is a quadcopter
stage - real world environment (RWE) or simulated environment (SE)
controller - the Pixhawk or the simulation thereof; collects sensor inputs about the player and the stage and sends motion commands to the player
model - the ROS we write which determines motion commands given sensor inputs and a desired trajectory
parameters - information about the UAV which affects its motion. There are two types: parameters that are characterized (e.g. weight) and parameters that are controlled (e.g. the thrust at each rotor over time).

Note: Gazebo (the simulator we're using) uses the word 'model' to refer to the combined mesh, texture, and behavioral characteristics given to an object in the simulated environment. We're using the word 'player' to refer to the UAV, and the word 'model' to refer instead to the ROS we write to determine motion control given a sensor input and a desired trajectory.

3 Materials

Initially, we'll be operating entirely within a simulation to develop a working model in isolation from the real-world variance that comes with actual hardware. We will be simulating the player and its environment using Gazebo (gazebosim.org), an open-source robotics simulator which includes a physics engine (to simulate the stage) and a sensor library (to simulate the controller's sensing capabilities). Gazebo has a built-in ROS interface, which will enable us to write the model in ROS and attach it to the simulated player via the simulated controller.

We're running Gazebo and ROS on our student laptops (Dell Latitude E6430s).

4 Methods

We plan to obtain an existing Gazebo model of a real-world UAV to use as a simulated player (again, note: we're referring to a Gazebo model as a player). We'll use Gazebo's built in sensing library to simulate the sensing capability of the UAV's controller, and the Gazebo-ROS interface to simulate the controller's ability to send motion commands to the UAV.

Our primary task is to write model software which can 1) collect sensor information (absolute position, orientation, and velocity) from the player while performing simple maneuvers in order to characterize the motion of the player and then 2) using the characterization, take sensor information and desired complex trajectory as input and determine motion commands which result in the complex trajectory. We will write the software in ROS, and we will use the built in Gazebo-ROS interface to send sensor information from the simulated player to the model and to send motion commands from the model back to the player. The UAV has four controlled parameters: the thrusts produced by each of the four rotors. As a result, when the model produces motion commands, it produces a thrust for each rotor.

Part of this process will be testing various models and parameters to find combinations that will allow for the characterization of UAV sufficient to perform the types of trajectories listed in our goal statement. This might include creating variations of models from pre-existing research and/or creating our own model from first principles.

5 Result/Goal Verification

To verify the the success of the self-characterization, we will compare the desired trajectory to the performed trajectory. A program external to our ROS model will collect absolute location, orientation, and velocity data from the simulation or stage (without the error innate to the player's controller) and compare them to the desired trajectory. We plan to compare by summing location and orientation absolute errors from desired trajectory over all of the points for which the controller collects information, then dividing the position absolute error by the longest dimension of the trajectory and the orientation absolute error by 90 degrees to standardize. We are placing a success threshold of 20% on each error statistic.

6 Reach

Once we get the UAV working in the simulation environment, our first reach goal is to transfer the model software we've created to the real world by applying it to an actual UAV. Even if our model workbs well in the simulated world, we may need to update it to account for real-world variance. If we are able to pilot a real UAV in two dimensions, we will return to the simulated environment and work to incorporate the third dimension. To self-characterize in three dimensions, the UAV will execute linear maneuvers in the x-, y-, and z-directions; it will then be able to combine those orthogonal motions to perform more complex maneuvers in space rather than constrained to a plane.

We plan to pilot one of the quadcopters owned by Olin's robotics lab (likely either a Parrot or an Iris), and we plan to control it using a Pixhawk PX4 Autopilot controller. The Pixhawk includes a gyroscope and an accelerometer, which it can combine using a built-in extended Kalman filtering process to produce estimated attitude and heading.

We will validate our results in the same way we validated the simulated results - we will store absolute location, orientation, and velocity data from the Pixhawk onboard the UAV, and we will compare it to the corresponding values from the desired trajectory using the same standardized absolute error process described in Section 5.

7 Schedule

Day	**Tasks**	**Deliverables**
Wed 2/18	Update quad chart	Experimental Write-up
Sun 2/22	Beginner and intermediate Gazebo tutorials complete; ROS downloaded	
Mon 2/23	Search for Gazebo model (player)	Updated quad chart
Wed 2/25	Create review slides	
Sun 2/28	Advanced Gazebo tutorials (+ as much ROS as is necessary) complete	
Mon 3/2	Working on setting up environment and control; researching mathematical models	Review Slides
Wed 3/4	" " "	
Wed 3/11	Create review slides and prepare for design review	
Thu 3/12	Design Review (complete environment w/ appropriate simulation of player and hard-coded control from ROS)	Review Slides
Mon 3/16	Week of Spring Break	
Mon 3/23		
Wed 3/25		
Thu 3/26		Updated quad chart
Mon 3/30		
Wed 4/1		
Thu 4/2	Project Reviews	Review Slides
Mon 4/6		
Wed 4/8		
Thu 4/9		Updated quad chart
Mon 4/13		
Wed 4/15		
Thu 4/16		First draft of report
Tue 4/21		
Wed 4/22		
Thu 4/23	Guest Speaker	Revised draft of report, draft of slides
Mon 4/27		
Wed 4/29	Last edit of final report; practice presentation	
Thu 4/30		Presentation/Final Report