sdmay19-35: Implementing a Web Portal System for Drone Simulation and Control

Week 7 Report

October 27 – November 2

Client: Ali Jannesari

Faculty Advisor: Ali Jannesari

Team Members

Bansho — *Test Engineer. Sensors Hardware Developer.*

Ian — *Scrum Master. Full Stack Developer.*

Li — *Test Engineer. Back-end Developer.*

Jawad — *Meeting Manager. Embedded Systems Developer.*

Mehul — *Project Lead. Computer Vision Developer.*

Sammy — *Report Manager. Lead Front-end Developer.*

Summary of Progress this Report

- Decided on and transitioned to Gazebo
  - Gazebo provides a web-client called GzWeb that eliminates much of the work for us and is maintained by an active and ambitious community
    - Moves the physics computations to the Gazebo's GzServer component, which executes on the server.
    - The front-end communicates with GzServer via web sockets in order to keep the client up-to-date in real-time. Three.JS is used to render the models using the data received from GzServer.
    - Uses ROS topics, sent via the GzBridge component, to make communication with the server easy to handle and adjust.
      - A custom topic was added for drone movement.
  - Gazebo model plugins are chunks of C++ code that allow control of a model in a Gazebo simulation environment.
    - Created a plugin for drone movement that receives directions and velocities via ROS topics from the front-end and moves the model accordingly.
  - Provides an extensive model library that contains a simple quadrotor that we can use for our purposes.
    - Created a simple environment with a couple vehicles and a building with a drone.
  - However, Gazebo has heavy dependencies. Particularly for development
    - C++ development requires libgazebo9-dev package
    - JavaScript development requires npm package
    - However, installing one uninstalls the other and some build processes require both
    - Discussed Dockerizing to avoid the issue.

- Experimented with WebODM for generating 3D models from aerial images
    - When provided with enough pictures to effectively determine the attributes of the pictured subjects, accurate 3D models can be generated.
        - 15 images generated a patchy looking model after 20 minutes.
        - Tasks for image stitching can be ran in parallel.
        - Discussed with client about utilizing a cluster machine in order to support the resource intensive process of generating the 3D models.
- Began setting up communication between a client and ROS
    - Our front-end will send binary commands to the server which will interpret them as ROS commands, executes them, and returns the output to the client.
    - Set up 2 processes that communicate with each other.
        - Not currently communicating via TCP connection.
        - The “client” process sends a message to the other process in binary and the other process interprets the message but does not execute it yet.

Pending Issues

- The dependency issues associated with Gazebo make development and deployment difficult. Consider Dockerizing.
- WebODM takes a long time to produce results, explore methods for optimizing the performance.
- The drone must still be ordered, some issues occurred with finding a vendor that can ship in a reasonable time.
- Still awaiting the camera for the Raspberry PI, cannot proceed with that component until it is received.
- All the additional functionality added in previous sprints needs to be reimplemented.

Individual Contributions

Team Member	Contribution	Weekly Hours	Total Hours
Bansho	Started setting up ROS packages on the Raspberry PI	7	45
Ian	Worked with Sammy to transition the simulator to Gazebo.	6	48
Jawad	Set up communication between processes via sockets	7	48
Li	Researched PID controller for controlling the drone.	8	44
Mehul	Implemented simple examples for rendering 3D models using WebODM	9	46

Sammy	Added rotation to each propeller individually, began researching alternatives	8	51

Plans for Upcoming Reporting Period

- Frontend
  - Dockerize the GzWeb component to avoid dependency issues.
  - Re-implement features such that they work with the new simulator.
  - Improve the movement and tweak the physics to be more believable.
- Backend
  - Improve the inter-process communication to by adding TCP connections and execution of the ROS command.
  - Implement more complex WebODM scenarios and determine best methods for optimizing the performance.
  - Establish communication between ROS and the Raspberry PI to mock the communication with the drone.
  - Coordinate with the client to place an order for the drone as well as any other necessary hardware.