**Graduate Projects**

University of Colorado at Boulder

Aerospace Engineering Sciences

ASEN 5018/6028 –Fall 2015

|  |
| --- |
| **FlyNet**  **Perception Subsystem Summary/Continuity Document** |

|  |  |  |
| --- | --- | --- |
| **Approvers List** | | |
|  | Title | Name |
| Prepared By | Perception Team |  |
|  |  |  |
| Approved By | Perception Lead |  |
| Approved By | Systems Engineer |  |
| Approved By | Project Manager |  |

**1: Introduction & Summary**

The goal of the perception subsystem is to provide the onboard infrastructure needed to implement target identification and provide online updates to the a priori map.

We have implemented a simultaneous localization and mapping (SLAM) solution using a hardware-assisted stereo vision setup. The SLAM solution, known as RealTime Appearance-Based Mapping (RTABMap), provides the onboard navigation algorithms an estimate of the UAV's currrent position in the world map. Target identification is accomplished via a FLIR Lepton sensor, while the hardware stereo setup is provided by a DJI Guidance subsystem.

# **2: Semester Report**

## 2.1: Objectives and Tasks List

Here is where you will list **ALL** goals and tasks that you’ve either been assigned or have determined yourselves, **complete or not**. Tie to a requirement where applicable.

**Completed**:

1. Completed a trade study of perception sensors and associated SLAM techniques
2. Demonstrated a functional SLAM solution appropriate to flight hardware limitations without UAV integration
3. Implement sufficient glue ROS nodes to enable system integration between Guidance hardware and Pixhawk autopilot
4. Demonstrated functional SLAM solution onboard flight hardware

**Incomplete**:

1. Closing the loop between flight control and planning using SLAM outputs
2. Updating a priori map with SLAM results

## 2.2: Issues

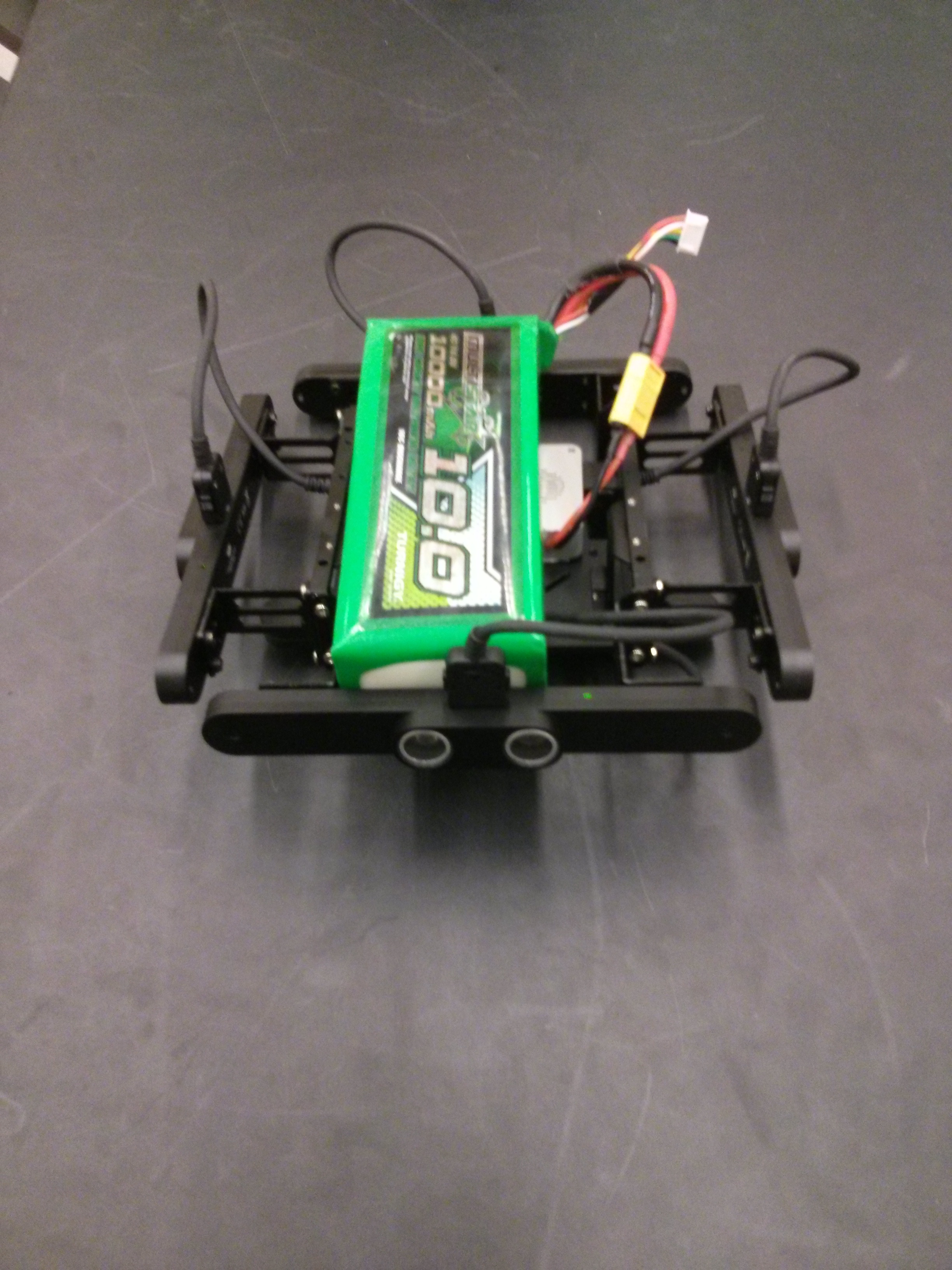
1. Many implementation problems stem from the use of the DJI Guidance sensor. While this sensor has a strong potential, its limited market penetration has led to underdeveloped software support that had to be backfilled by team members. While alternative sensors exist that could readily be used to implement SLAM, Guidance is able to offload many of the most CPU-intensive tasks and preserve the onboard execution budget. Guidance was unavailable until approximately midway through the semester, shortening the amount of time available to identify and solve integration problems.
2. Planning and map update integration tasks remain incomplete due to time constraints.

## 2.3: Lessons Learned

1. All block-matching stereo vision algorithms include a blank area on the left of the depth image as a side effect.
2. Passive stereo vision is fundamentally more constrained than structured light active sensing due to a requirement for feature correspondences.
3. Having a technology in which only one person on the team is familiar presents management difficulties to prevent burnout and/or task stovepiping.
4. Binary-only blobs are a recipe for disaster.

## 2.4: Procedures

The perception subsystem is implemented in three layers: mechanical, logical, and electrical. SLAM is implemented using Guidance's outputs, while targeting uses the FLIR output coupled with the current pose reported by SLAM to localize targets. The remainder of this section assumes the user is familiar with ROS, a data distribution and processing middleware layer; the reader is directed to [www.ros.org](http://www.ros.org/) for further information and background. All nodes are implemented in ROS Indigo.

Figure 1: Guidance Sensor

Mechanically, the Guidance system has its own carrier that support five sensor nodes and its central processing core. Four nodes face in cardinal directions, while the fifth faces downward. Each node consists of a stereo pair as well as an ultrasound sensor. Further information is available from DJI at <https://developer.dji.com/guidance/>

Electrically, Guidance is capable of directly being driven by the onboard 4S LiPo battery; Figure Figure shows a testing configuration with a battery directly connected to Guidance via an XT60 connector. Data are output from Guidance using either serial connection or a USB connection. Due to bandwidth limitations of the serial interface, we used the USB connection via a micro-B to A cable to the onboard CPU.

Logically, the USB connection does not implement any standard device classes; rather, the vendor defines a set of custom URBs. Of note is that DJI does not have an officially sanctioned VID from the USB Implementer's Forum, which could lead to VID/PID conflicts in the future. Each of the custom URBs is accessed via a libusb-1.0 based API that is only distributed in binary form from DJI. Currently, x64, x32, and ARMv7 are available on Linux. Via the API, Guidance's sensor information is available, including ultrasound, rectified imagery, visual odometry, and IMU data.

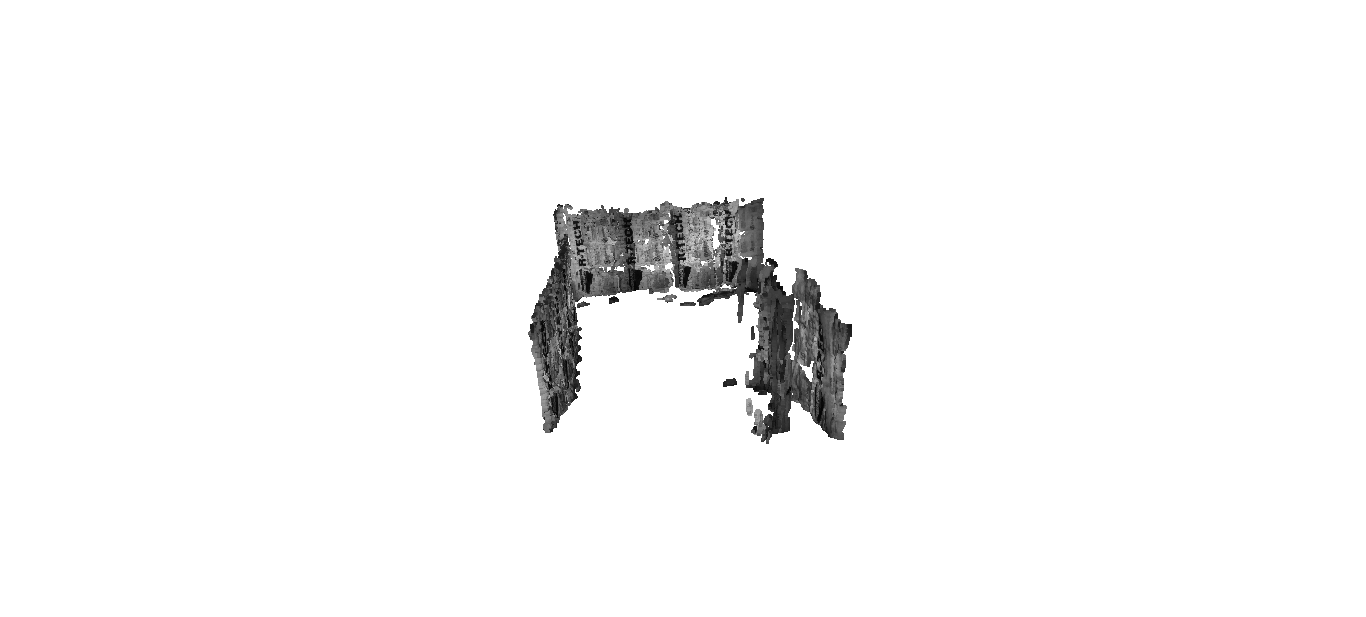
In our testing of the Guidance system, the onbaord block-matching based stereo pipeline produces inferior results that are unusable for the purposes of depth-based SLAM. The depth images are sparse and riddled with inconsistencies that lead to extensive mapping errors. As a result, we implemented a workaround that utilizes Guidance to produce visual odometry, but instead uses the rectified stereo pair as input to RTABMap. This pipeline shows promise and is a realistic solution going forward.

As provided by DJI, the Guidance ROS node was unusable and required several modifications. The updated source is available through the BitBucket repository under <https://bitbucket.org/cuflynet/guidance-sdk-ros.git>, while the upstream is <https://github.com/dji-sdk/Guidance-SDK-ROS.git>. Our customizations include the addition of code to cleanly shutdown the Guidance interface and prevent a situation where the Guidance core does not respond to new connection requests, as well as adding runtime configuration flexibilty to change operations without requiring code recompilation.

The bulk of the SLAM operations are provided by RTABMap running onboard the host CPU, available at <http://introlab.github.io/rtabmap/>. ROS nodes are provided that nicely wrap the underlying codebase and provide a clean interface to sensor inputs and SLAM outputs while providing for headless operation (so that no UI need be rendered aboard the UAV). The primary configuration required of RTABMap was to ensure that all needed ROS topics were publishing, either from the *guidance* ROS node directly or from intermediate processing nodes.

As a first cut, RTABMap was run against an Asus Xtion Live Pro RGBD camera that utilizes structured light to produce a depth image. RTABMap was able to produce a reasonable map of the Fleming workspace using this input while running aboard a laptop, but demonstrated high CPU utlization related to the computation of visual odometry.

Once the Guidance system was available and its ROS node was usable, RTABMap was run against a single image and the corresponding depth image. The produced map was useless, a mass of blobs. The mitigation strategy was to migrate processing to the host in an incremental basis to attempt to recover function and minimize impact on host CPU.

Figure 2: RTABMap Output of a basic hallway

The next strategy was to use a rectified stereo pair produced by Guidance as input to RTABMap, generally following the tutorial at <http://wiki.ros.org/rtabmap_ros/Tutorials/StereoOutdoorMapping> and adjusting parameters as necessary to conform to our system architecture. An interim result is shown in Figure Figure that clearly shows a nominal hallway and a clear space between walls. This result was produced using a laptop instead of flight hardware; the CPU utilization of a flight hardware solution has yet to be characterized.

This section will likely be the bulk of your report. What did you actually do? Be as detailed as you can. **List any software used, including the version of the software**. You started from nothing, how did you get to the completed tasks? What progress have you made on incomplete tasks and what have you been doing to make that progress? Where can more detailed documentation be found?

Table 2.1: Software list

|  |  |  |
| --- | --- | --- |
| **Program Name** | **Version** | **Purpose** |
| **Hitchhiker Galactic** | 4.31b | Determine the meaning of life |
| **Wonderbread Enhanced Vision** | 2.0 | Sandwich simulator |

# **3: Next Semester/Future Expectations**

## 3.1: Prioritized List of Tasks and Objectives

Include your incomplete tasks ***and*** next steps for your subsystem. It is important for you to be thinking ahead. If your project is not continuing next semester, summarize what you think could be done if your project was reinstated in the future.

1. Finish preliminary design of toaster
2. Finalize the decision on the peanut butter – jelly interface
3. Start analysis of fish stick crumbliness

## 3.2: Starting Points

For each one of the tasks and objectives in 3.1, describe to the best of your knowledge where the person reading this can start. The person reading this is either going to be you after over a month off, or someone totally unfamiliar with how to go about things. **Point to the locations in the server where you have pertinent files saved.** Now the person has the file open in whatever software, what should they work on doing to start with?

1. Structural design of the toaster can be found on the project server under Working Directories > By Subsystem > Structures > CAD > Toaster.cad. We suggest first adding a best estimate for the heating coils to place volume constraints on the real coils that can be used. Other work that needs to be done is ensuring the polish will be shiny enough to see your face in the reflection.
2. The lingering issue in finalizing the peanut butter – jelly interface is the type of jelly to be used. We have collected some data on various jellies, but a trade study should be performed ASAP to determine the best jelly option.
3. Fish sticks have already been chosen but their crumbliness has not yet been analyzed. We suggest importing the CAD model of the fish stick and performing FEA stress analysis on the fish sticks to determine how much mass will be lost to rogue crumbs.

## 3.3: Improvement, Updates, Verification

For tasks you have completed, what could/should be done to improve or update them in the future? Here is a good place to blatantly state all the assumptions you have made, and prioritize them in order of the impact the assumption has on your result. As assumptions later get filled with more concrete data, your analysis will need to be updated and/or verified to ensure no issues have been raised.

Note: Be careful with improvements -- remember the goal is always to meet the requirement and not go any further.

1. We have assumed that there will be a large body of water near Boulder, CO for the team yacht. When that assumption is verified, the yacht should be moved to the body of water. Should this assumption prove to be invalid, drastic redesign of the boat may be necessary.
2. We have assumed that the question “what is the meaning of life” is one that makes sense. We have an answer to this question, but the question may need to be updated in the future as it may be out of the scope of this project.
3. In the design of the perpetual motion machine, we assumed the existence of tachyon particles to provide superluminal ghost-forcing on our gyroscopic, electromagnetic tether. Specifically, the tachyons are expected to produce a force of 30N. This number was determined using superluminal, reverse-time Feynman diagrams but should be updated when a better understanding of the interaction between the Higgs boson, muons, and tau-neutrinos is obtained.