PROJECT KINGFISHER

Critical Design Review

Eitan Babcock, Ryan Gibbs, Shu-Kai Lin, Kelsey Ritter, Samuel Wang

December 17, 2015

Abstract

This paper describes our plans to solve the problem of landing a helicopter on a ship deck. Ships in open water are inherently difficult to land on, as they are not stationary objects. A ship's deck is a moving target, which adds uncertainty to landing any flight, especially with a pilot's constrained field of view, inhibited by the aircraft itself. These factors make the landing portion of flight one of the most dangerous. An autonomous landing control technology is proposed that utilizes infrared beacons to determine the position and characterize the movement of the landing deck. The characterized movements will be used to determine a safe landing time and location. This will allow the onboard system to calculate a trajectory to land safely and quickly. The controller will then execute this planned trajectory to meet the landing deck at the calculated time and position. This technology will improve the safety and efficiency of the landing process, preventing human injury as well as saving money by reducing expenses from recovering crashed aircraft.

Table of Contents

1. Project Description

Project Kingfisher aims to develop a sensor suite and trajectory planning software for an autonomous rotorcraft to land on ships without the use of GPS or radio. Military operations at sea often utilize air vehicles traveling to and from ships. Aircraft carriers are large enough to allow landing and takeoff of fixed wing aircraft, but smaller ships are being used to support rotorcraft vehicles. With varying ship sizes and sea states, the pilot of a helicopter must be highly experienced to safely land the aircraft on the moving deck.

With the performance improvement of processors and the maturity of sensor technologies, unmanned aerial vehicles (UAVs) are becoming popular in a wide range of applications. These applications include military service, aerial photography, surveillance, environment mapping, cargo shipping, etc. [1]. Work has been shared between man and machine, and the working efficiency and capability has been increased since UAVs can access places humans cannot and can operate more precisely. However, without intelligence onboard the vehicles, UAVs can be very dangerous. A collision between a UAV and a landing base can cause significant loss and even human injuries.

Therefore, a technology that can autonomously land a rotorcraft on a shipdeck by combining sensing and prediction is a potential solution for improving landing performance. Project Kingfisher will utilize a vision system suite carried by a small-scale quadrotor to demonstrate algorithms that can safely land a rotorcraft on a dynamically moving deck.

2. Use Case

Mr. Gman is a pilot in the Navy. His main mission as a helicopter pilot is to survey the environment of the battlezone. Every time he goes on a mission, he is worried about whether he will be able to come back to the ship safely because the helicopter landing environment can be very dangerous. Beside the threat from enemies, natural disturbances such as wind, lack of lighting, and rough seas can easily cause an accident.

The Navy utilizes many resources to reduce the risk of helicopter landing. Every possible effort has been made to solve the problem, including intense training for pilots and more stable vehicle platforms. The military has realized that the limitation of human capability is a key factor that causes accidents. Since environment surveying can be done by unmanned machines, one of the solutions is to apply the autonomous pilot system on the helicopter. After decades of cooperative research with the Robotics Institute at Carnegie Mellon University, a sensor suite and trajectory planning software are mature enough to provide autonomous landing for helicopters.

With this new technology, Mr. Gman no longer needs to operate the helicopter during landing. Instead, he flies the helicopter to the vicinity of the ship and pushes a button to activate landing mode. During landing mode, the controls are taken over by the landing autopilot. In this mode, the helicopter can sense the location of the ship deck from a far distance via cameras and

infrared beacons without the use of GPS. Although the waves of the ocean heave the ship deck, the helicopter is able to predict the motion to determine the point in time when the deck will be in the safest landing position. The helicopter determines the speed and trajectory it must fly to meet the deck at that point in time. The helicopter meets the deck safely and smoothly, though the deck is still heaving from the ship riding the waves. The landing mode then deactivates, sending a signal to Mr. Gman that the autopilot has successfully landed the helicopter on the deck.

3. System-level Requirements

Functional Requirements		
User Need	**Performance Requirement**	**Mandatory/Desired**
Rotorcraft shall identify the deck within the environment.	1 false positive deck identification allowable in 100 trials from 100m.	Desired
	Rotorcraft shall sense position of deck in relation to the rotorcraft with accuracy as function of distance as follows: 1. Detection at minimum 100m distance 2. <100m distance : 0.4cm allowable error for each meter away	Mandatory
When within landing range, rotorcraft shall predict dynamics of the deck.	Rotorcraft shall predict movement of deck within 0.8cm/m of current distance to deck over total algorithm loop time.	Mandatory
The rotorcraft shall robustly follow a planned trajectory.	Follow planned trajectory 99% of time, within 0.8cm/m of current distance to deck.	Desired
Rotorcraft shall land on the landing zone of the deck.	Rotorcraft shall land within 50cm of center of the deck, as measured from the center of the rotorcraft.	Mandatory
	Rotorcraft shall land on dynamically moving deck.	Mandatory
	Rotorcraft shall perform 8 successful landings over a 10 cycle lifetime.	Desired

Nonfunctional Requirements		
User Need	**Constraint/Requirement**	**Required/Desired**
System shall operate in EMCON conditions.	Operation of autonomous landing mode shall be possible without radio commands and in GPS-denied conditions.	Mandatory
System shall operate with minimal user input.	Autonomous landing sequence shall execute after a single user "go" command.	Mandatory
Rotorcraft shall land safely.	System shall operate without damage to rotorcraft or deck.	Mandatory
Rotorcraft shall operate in varying environmental conditions.	System shall operate through steady wind up to 5mph.	Desired
	System shall operate through gust wind up to 10mph.	Desired
	System shall operate in 50m visibility.	Desired
	System shall operate from minimum 0.0001 lux to maximum 100,000 lux	Desired

4. Functional Architecture

4.1. Visual Structure:

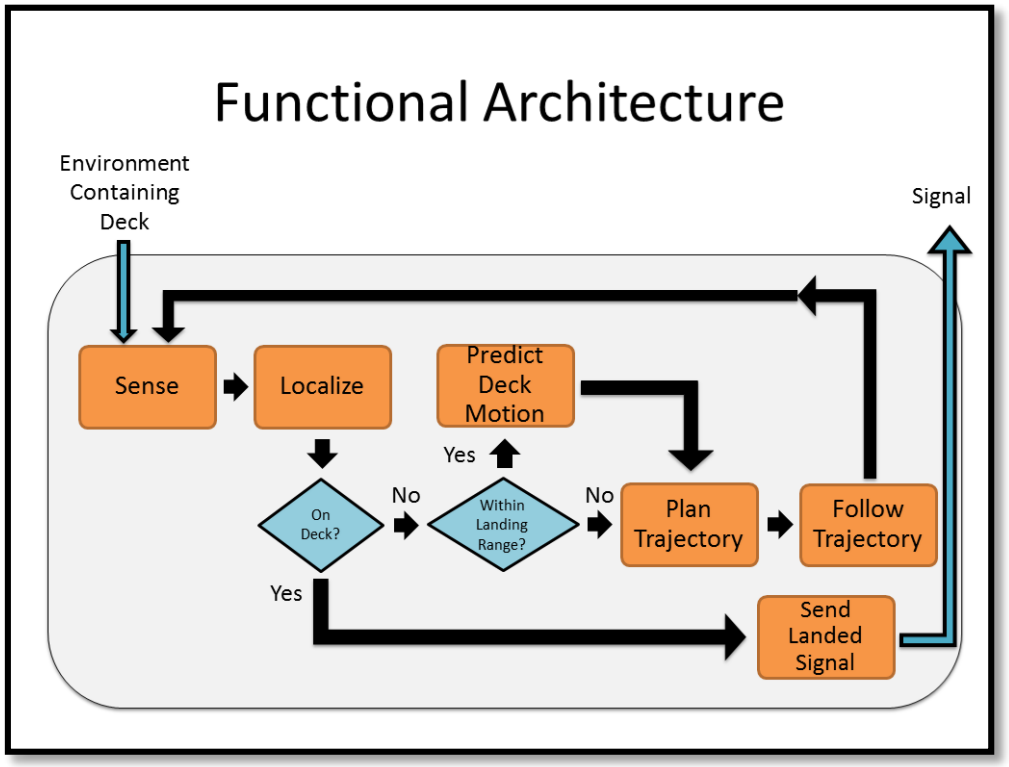


Figure 1: Functional Architecture of Landing Process

4.2. Inputs

The input to the autonomous landing system is an environment containing the ship deck. It should be noted that for our system, the assumption is made that the rotorcraft and the ship deck are the only objects in the environment besides the ground/water. We assume that there are no obstacles either on the deck or in the path of the rotorcraft.

4.3. Sense

The sensors output data about the environment at the maximum data rate that can be handled by the system.

4.4. Localize

At long range, the position is sufficient to construct a heading for the rotorcraft to approach the deck. Once within landing range, a constructed frame is used to determine the relative position and orientation of the ship deck with respect to the rotorcraft.

4.5. Predict

Using the history of deck poses, the motion of the ship deck is modeled in time. Once sufficient data are collected for an accurate model, the model is used to predict the future pose of the ship deck.

4.6. Plan Trajectory

At long range, the planned trajectory is simply the shortest distance between the rotorcraft's current position and the detected position of the ship deck. Once within landing distance, the prediction model for the motion of the ship deck is used to determine which windows in the deck's cycle are safe to land during. Taking the system's current position and average flight velocities into account, a trajectory is planned that times the landing with one of the future windows for safe landing.

4.7. Follow Trajectory

The desired trajectory is fed from the system's computer into the flight controller for the rotorcraft. The rotorcraft then executes the planned trajectory until it is altered or the rotorcraft lands successfully.

4.8. Send Landing Signal

Upon landing, the system needs to send a confirmation signal to the user that it has landed. This allows the user to issue further commands or to perform some manual task on the rotorcraft.

4.9. Outputs

The system output is a landed rotorcraft and an output signal to the user that the system has landed successfully. The system will then enter an idle state awaiting further instructions, releasing autonomous control of the rotorcraft.

5. Cyberphysical Architecture

5.1. Visual Structure

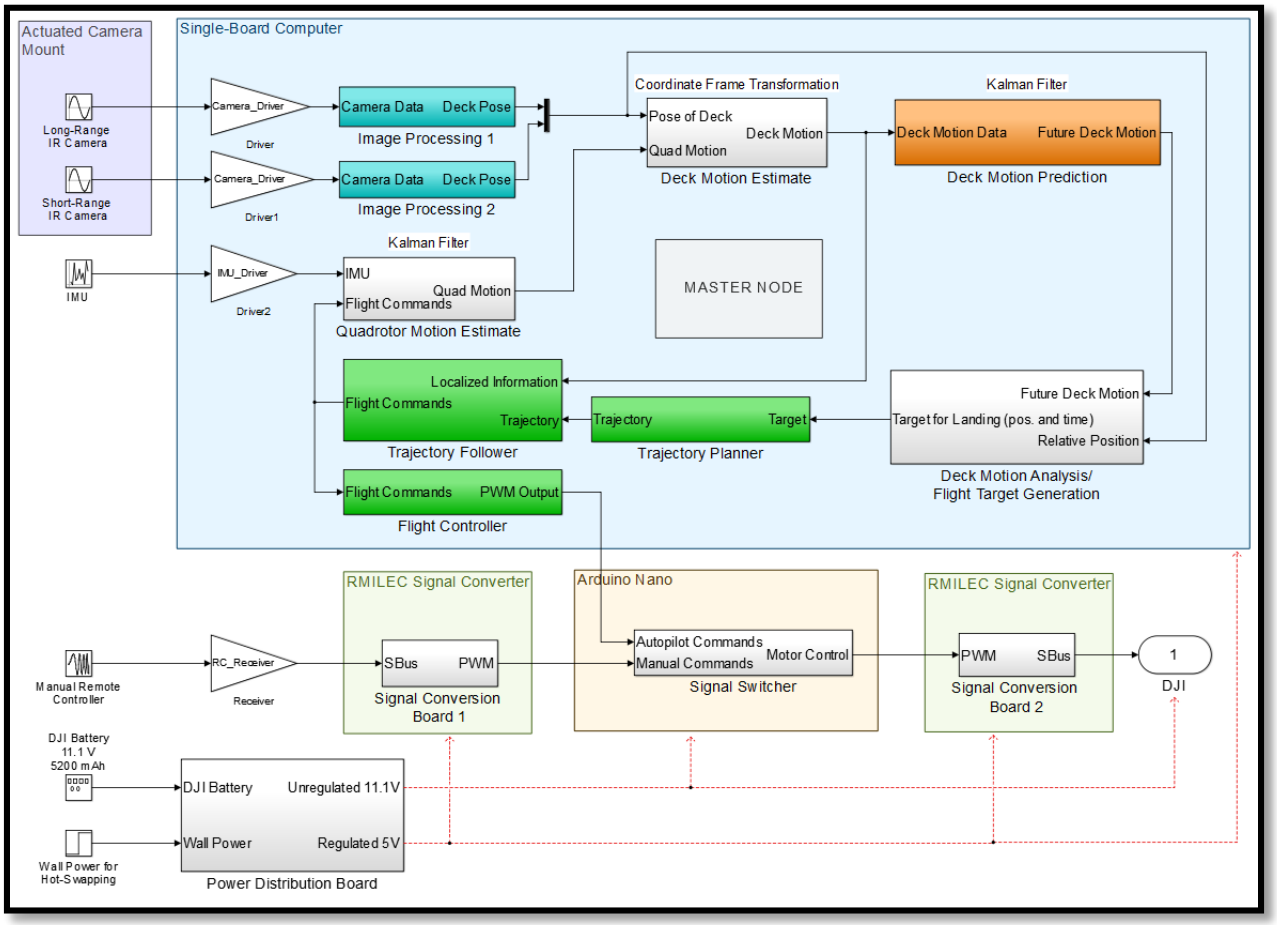


Figure 2: Cyberphysical Architecture of Landing System

5.2. Camera Mount

Our cameras need to be mounted to the mobile platform. The exact mount that will be used is yet to be determined. The primary options we are considering are fix-mounting two cameras at different angles or building a motorized camera mount that can tilt in pitch to account for motion of the rotorcraft.

5.3. Single Board Computer

Our single board computer houses the primary software and processing for our system. The image processing, localization, prediction, trajectory planning, trajectory following, and flight command subsystems all reside here. Additionally, there is a master node that handles the routing of command signals and switching between states.

5.4. Additional Boards

In addition to our primary single board computer, we have two signal conversion boards and an Arduino Nano. The signal conversion boards convert the remote control signal from SBus to PWM and the motor controls from PWM to SBus. This is needed because the DJI Phanom II expects SBus communication from the remote controller, but our single board computer and Arduino Nano use PWM communication. The Arduino Nano takes the inputs from the remote controller and the single board computer and determines which signal should be routed to the rotorcraft.

5.5. Power Distribution

Our printed power distribution board takes power from the DJI Phantom II's battery and distributes it to the various other boards. There is a 5V regulator on this board to provide power to the single board computer and the conversion boards. The Arduino Nano has an internal power regulator, so it receives the full voltage of the battery. The cameras and IMU are powered via USB from the single board computer.

6. Current System Status

6.1. Fall-Semester Targeted System Requirements

Table 7.1 – Fall Semester System Requirements and Associated Subsystems

Functional Requirements		
User Need	**Performance Requirement**	**Relevant Subsystems**
Rotorcraft shall identify the deck within the environment.	Rotorcraft shall sense position of deck in relation to the rotorcraft with accuracy as function of distance as follows: 1. Heading detection at minimum 50m distance 2. <50m distance : 25cm+0.8cm/meter away	Vision, Deck Marking, Localization Algorithm
Rotorcraft shall land on the landing zone of the deck.	Rotorcraft shall land within 50cm of center of the deck, as measured from the center of the rotorcraft.	All **except** Prediction Algorithm
	Rotorcraft shall land on a stationary deck.	All **except** Prediction Algorithm

Nonfunctional Requirements		
User Need	**Constraint/Requirement**	**Relevant Subsystems**
System shall operate in EMCON conditions.	Operation of autonomous landing mode shall be possible without radio commands and in GPS-denied conditions.	Entire System
System shall operate with minimal user input.	Autonomous landing sequence shall execute after a single user "go" command.	Entire System
Rotorcraft shall land safely.	System shall operate without damage to rotorcraft or deck.	Entire System

6.2. Current System Descriptions

6.2.1. Overall System Status:

The kingfisher system as a whole contains an implementation of every major subsystem at the conclusion of fall semester. Many of the current iterations of these subsystems are limited in their application scope, and will require considerable improvement over the course of the next semester. Specific system statuses and planned changes are outlined by subsystem below.

6.2.2. Vision Subsystem Status

The vision subsystem is responsible for the detection of the ship deck and its relative pose. At long ranges, the position will be determined using the long range IR camera on the rotorcraft and the IR beacons on the corners of the deck. At medium range, all four of the IR beacons will be distinct enough for the camera to interpret individually. Finally, at close range, the vision system will switch to the short range landing camera to allow for a greater field-of-view as the distance to the deck decreases.

As of the end of fall semester, the initial version of the vision subsystem is fully implemented. The long and short range cameras are both assembled with their respective lenses and are fitted with long-pass IR filters. The lens option selected for the short range camera may have to be changed next semester to accommodate a wider field of view. The current plano-convex lens with a 7.5mm focal length yields a 39 degree field of view already. Unfortunately, the motion of the quad is significant enough that some maneuvers cause one or more of the deck beacons to leave the field of view. As a result, we will investigate the use of a fish-eye lens for the short range camera to combat this effect. The motion of the quad is less of an issue at long range where the angular separation of the beacons from the perceived center of the deck is much less.

6.2.3. Deck Marking Subsystem Status

The deck marking subsystem is responsible for indicating several points of interest on the ship deck to the rotorcraft. The IR beacons on the deck provide an approximate location at long

distance due to their greater than .5 mile visibility. Once the rotorcraft is close enough to distinguish between them, the center of the deck and the pose of the deck can be determined based on the prior knowledge of their configuration on the deck.

Currently, they are arranged on the deck in a square of 1 foot side lengths offset from the center of the deck by a foot and a half in the vertical direction. This offset is to account for the current fixed mounting of the cameras at 30 degrees from horizontal. The offset allows the beacons to remain in the field of view even when the rotorcraft is on its final descent directly above the deck.

6.2.4. Localization Algorithm Subsystem Status

The input to the localization algorithm is the two images from the long range camera and the short range landing camera, as well as the data from the rotorcraft IMU. Let landing range be the range within which the deck beacons are within the field of view of the short range landing camera, but the long range camera can no longer see all 4 deck beacons. Let prediction range be the range beyond landing range, but within long range, within which the system will attempt to perform prediction of the deck's motion. Let long range be the range beyond prediction range but within the maximum detectable range.

While at long range, the algorithm will identify the centroid of the four beacons and output the relative heading direction between the rotorcraft and the centroid, using the known mounting configuration of the cameras to construct a heading for the rotorcraft.

Once within prediction range, in addition to outputting the centroid location as above, the algorithm also reconstructs the pose of the deck. To do this, the algorithm identifies the centroid of the beacons to determine the deck's relative position as before. Then, relative pose will be determined by solving for the perspective transform of the known configuration of beacons on the deck. Finally, the "absolute" motion of the deck from one frame to the next will be determined by subtracting the rotorcraft's motion model data from the corresponding frames. This accounts for the motion of the rotorcraft during the time between frames. This model is generated through a combination of the corresponding motion command from the path planner to the flight controller and the IMU data for the target time window. The prediction range localization algorithm outputs both the absolute and relative pose of the deck.

Finally, within landing range, the algorithm will compute the relative and absolute poses as above, with the difference that the landing camera will be used in place of the long range camera.

In its current implementation, the algorithm is vulnerable to several conditions that will need to be addressed in spring. The first of these conditions is if one or more, but not all, of the beacons departs the field of view of the cameras. Since we're currently using K-Means to cluster the beacons to identify a single point to operate the perspective transform on, if there are fewer than four beacons in the image, the algorithm will incorrectly split one of the beacons to accommodate. This functionality is desired at longer ranges where the beacons are discernable,

but still contiguous, but causes problems at shorter ranges. To solve this for next semester, frame differencing will be used as a component of the filtering of the vision data to detect large perceived changes in beacon position and check if they're caused by loss of field of view. This will also be useful to maintain the performance of the system if one or more beacons are temporarily disabled or occluded. Another vulnerability condition is the presence of glare or extra IR light in the scene. Previous frame beacon position and size will be used to filter out false beacons in the spring implementation.

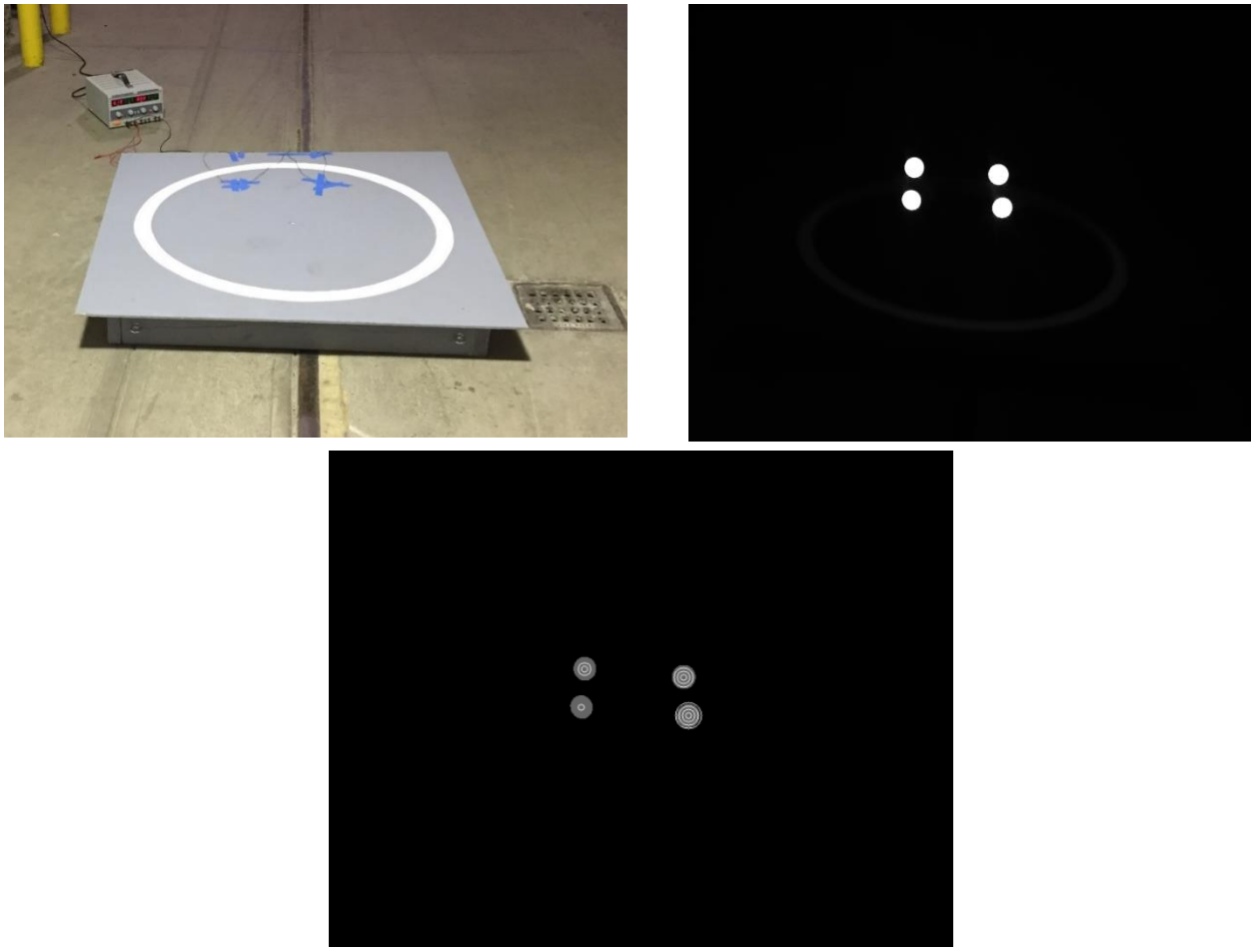


Figure 3: Current Status of Vision System. Upper-left: Scene image, with deck beacons where the blue tape is. Upper right: Raw camera input image. Bottom: Detected beacons numbered by concentric rings.

6.2.5. Prediction Algorithm Subsystem Status

The prediction algorithm will use the absolute pose of the deck over time to produce data of roll, pitch, and swell over time. This data will be in reference to an absolute world frame centered at the deck. The algorithm will then use a Kalman filter to predict the motion of the deck into the future. This will allow the rotorcraft to plan a trajectory such that it meets the deck at a relative high point and a relative flat point on the deck's path. This is critical in order to perform a safe landing with minimal damage to the rotorcraft or the ship.

This algorithm has only undergone initial development at this time, since the deck was static for the fall semester. Some basic state estimation and Kalman filtering has been done on false data for a simple harmonic system to familiarize ourselves with the implementation of Kalman filters in code. The bulk of the prediction algorithm will be developed and tested next semester when the deck is in motion.

6.2.6. Trajectory Generation Algorithm Subsystem Status

The trajectory generation system is for generating a trajectory from initial state to a target position [14]. The trajectory planning algorithm will produce that ideal trajectory based on the predicted movement of the deck. The planned trajectory should allow for the rotorcraft to seamlessly adjust its velocity during flight to meet with the deck at the proper time and position. This algorithm has two separate stages. The first stage occurs outside the prediction distance, where the localization algorithm computes relative pose between the rotorcraft and the deck, and sends this information to the trajectory algorithm. The trajectory algorithm uses this information with desired state to compute a thrust and rotation matrix to send to the flight controller. The desired state in this stage is to minimize the relative distance. This allows for the rotorcraft to simply fly in the direction of the deck, reducing the distance and increasing the vision system's ability to distinguish the individual beacons to get an accurate position and pose of the deck. The second stage of trajectory generation occurs within the prediction distance. In this stage, the trajectory algorithm not only takes the information from the localization algorithm, but also from the prediction algorithm. The prediction algorithm predicts the dynamics of the deck, and sends the predicted pose of the deck over time to the trajectory algorithm. The trajectory planning algorithm analyzes the deck motion over time to identify acceptable landing times, and identifies the ideal landing time based on the rotorcraft's distance from the deck. The trajectory algorithm will then generate an ideal trajectory to land on the deck at the specified "ideal" time. The trajectory planning algorithm will take this ideal trajectory, and convert it into aircraft roll, pitch, yaw, and thrust commands. These commands will be fed to the flight control microcontroller, which will in turn control the motors on the mobile platform.

6.2.7. Trajectory Following Algorithm Subsystem Status

The trajectory following algorithm subsystem is responsible for detecting errors in our current flightpath and making the corresponding course corrections to put the rotorcraft back on the target trajectory generated by the trajectory generation algorithm.

Currently, the trajectory following algorithm is a PID controller used to keep the rotorcraft at its current waypoint based upon feedback from the vision system and the IMU. This subsystem requires further development due to the coupling between X-Y and Z position.

6.2.8. Flight Control Subsystem Status

The flight control subsystem is responsible for converting the commands output from trajectory generation in terms of roll, pitch, yaw, and thrust to the necessary stick commands to emulate to the flight controller onboard the rotorcraft. Additionally, this subsystem toggles between manual flight of the rotorcraft by remote control and autonomous flight. Autonomous flight is achieved by emulating manual flight commands from the "remote control" generated by the trajectory generation and trajectory following algorithms. Control can be returned to the human safety pilot through the use of an extra channel on the remote control.

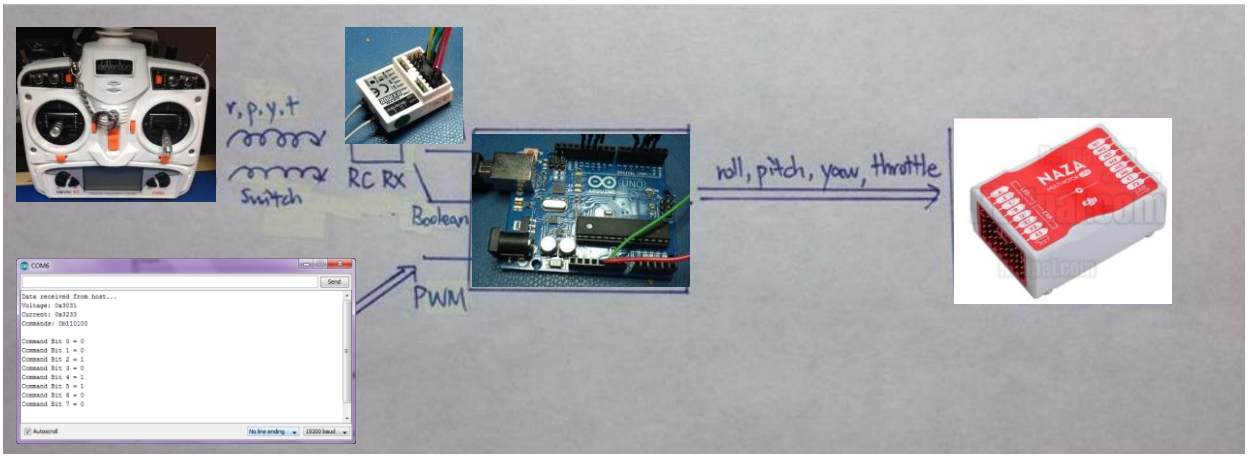


Figure 4: Flight Control Subsystem Depiction

6.2.9. Power Distribution Subsystem Status

The power distribution subsystem is responsible for the distribution of power from the standard rotorcraft battery to all onboard subsystems, as well as the addition of the ability to hotswap batteries with only the actual rotorcraft motors needing to be powered down. Due to errors in design during the fall, the board and design will need to be reworked and refabricated in spring to achieve full functionality. In its current state, the power output is noisy but usable for all subsystems. However, the hotswap feature is inoperable and power consumption is higher than designed.

6.2.10. Rotorcraft Subsystem Status

The rotorcraft is responsible for executing the command sent from the single board computer such as flying, landing, and flight mode switching. The rotorcraft consists of flight avionics, actuators, radio transmitter and receiver, vehicle frames, and battery. Flight Avionics is the brain of the system; it consists of a processor, sensors, and input/output (I/O) pins. The basic sensor a rotorcraft requires is the gyroscopes, which provides the angular information of the rotorcraft. However, without the accelerometer, the user needs to manually control the orientation of the rotorcraft. Typically, most flight controllers apply proportional-integral-derivative (PID) control for stabilization of the rotorcraft. The I/O pins are responsible for providing connections of actuators and command receiver to the processor. In this work, there are two inputs responsible

for the dynamic movement of the rotorcraft. One is the trajectory command send by single board computer while the other one is from radio transmitter in case of emergency. For actuators, brushless dc motors are used; each of the motors is driven by electronic speed controller. The thrust of the vehicle is provided by attaching propellers on the motors [13].

In the ideal situation, the rotorcraft should autonomously follow the exact planned trajectory under the influence of natural disturbance and in EMCON condition. Therefore, dynamic stability of the rotorcraft is the primary concern in this work since any disturbance to the rotorcraft could result in bias to the sensors. In the scope of rotorcraft platform trade study, the disturbance includes wind, payload, and etc. The rotorcraft should provide enough maneuverability and stability with the installation of single board computer, sensors and battery. Moreover, the landing gear should withstand multiple tests; so the cost can be minimized and the components can be protected. Finally, the application programming interface (API) and community support of the rotorcraft platform can be great sources to reduce development time.

6.3. Modeling, Analysis, and Testing

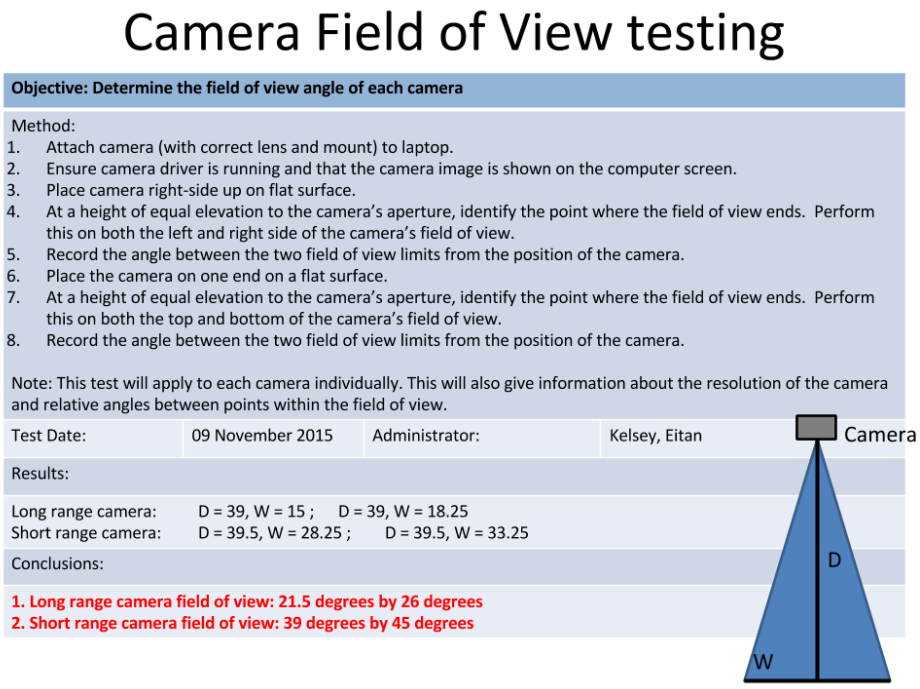


Figure 5: Camera field of view testing results.

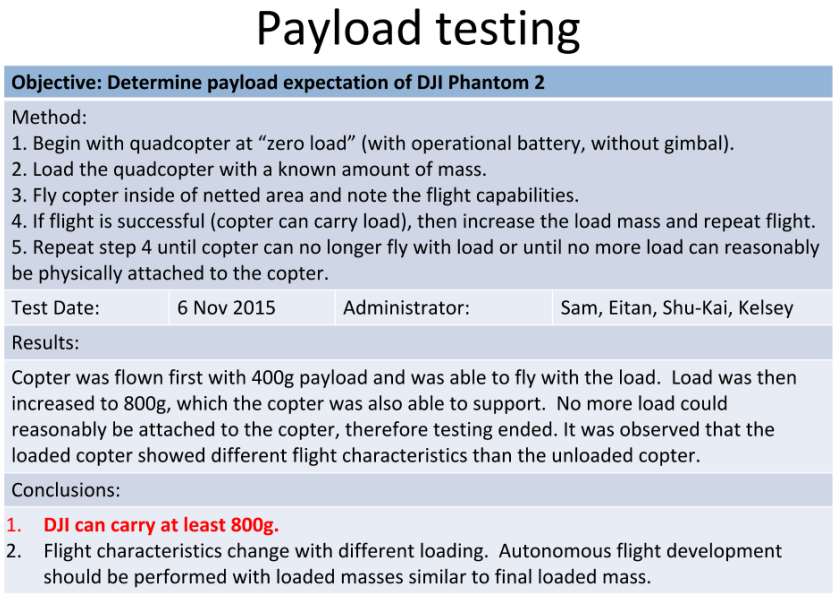


Figure 6: Payload testing results.

Additional testing was conducted prior to the FVE for the purpose of tuning the PID controller for trajectory following.

6.4. Performance evaluation against the Fall Validation Experiment (FVE)

Table 1: Requirements and test items for FVE.

Functional Requirements	**Test item**	**Result**
Rotorcraft shall sense position of deck in relation to the rotorcraft.	Heading detection at minimum 50m distance	Check ✓
	10m distance : Orientation and position of deck can be seen	Check ✓
Rotorcraft shall land on stationary deck.	Rotorcraft shall land within 50cm of center of the deck, as measured from the center of the rotorcraft.	Fail ✗
Non-Functional Requirements	**Test item**	**Result**
Operation of autonomous landing mode shall be possible without radio commands and in GPS-denied conditions.	Does the system operate in GPS-denied condition?	Check ✓
Autonomous landing sequence shall execute after a single user "go" command.	Is the user interface very simple?	Check ✓
System shall operate without damage to rotorcraft or deck.	Is the system still perfect?	Check ✓

Five requirements were needed in the fall semester. Two of them are functional requirements, and the rest of them are non-functional requirements.

The first functional requirement is for verifying the localization algorithm, and there are two test items in this part. First, at long distances (50m) the localization algorithm should detect the heading toward the deck. In this test item, the algorithm not only detects the correct direction but also provides the accurate position information. Next, the localization algorithm should provide relative position and orientation between the quadrotor and the deck at distance below 10m. We demonstrated a video and an image at the FVE to show that the algorithm can provide accurate data.

For the second functional requirement, we need to demonstrate the quadrotor landing on a stationary deck autonomously. We did not accomplish this goal, but instead demonstrated autonomous hovering. Though the position and orientation data provided by localization algorithm are accurate, we still cannot get the correct transform between coordinate frames. Also, the vision field of view range for detecting the beacons is limited. For these reasons, we were not able to land on the stationary deck.

Our system satisfies all of the non-functional requirements. It can operate in GPS-denied conditions, the user interface in our system is very simple, and our system operates without any damage.

According to the result mentioned above, we completed most of the requirements in FVE, and still need to work on the localization subsystem. Once we have correct feedback data, we can start to fine tune our controllers and move forward with landing the quadrotor.

6.5. Strong Points

6.5.1. Designed and built mechanical parts and power distribution board

In this semester, we have built the mechanical parts to mount all of the components on the quadrotor including the power distribution board, the IMU, the signal switch board, and the single board computer. Also, we have designed and built the power distribution board to apply a 5V and 11V voltage source to all of the components. Both of them work very well and reduce the difficulty to do experiments and testing. Though we might change some designs in spring, we can still leverage on this experience.

6.5.2. Implemented the baseline of algorithm for most of subsystems including

We implemented most of the subsystems in this semester and did the unit test on each of them. Every algorithm met their baseline requirements. All of us became familiar with ROS, so we are in a good position to modify or implement new algorithms next semester.

6.5.3. Integrated the system including mechanical parts, power distribution board, single board computer, and software algorithms

We have already integrated most of our subsystems on a single board computer. Though we still have a mismatch between the single board computer and the camera driver, it can be fixed by either changing components or finding another driver. Once we solve this issue, we will have a complete system. Then we can focus on developing software algorithms.

6.5.4. Built simulation environment to simulate the motion of the quadrotor

We have implemented a simulator in Gazebo for simulating the motion of the quadrotor. Base on this experience, we can easily extend the functionality to help test algorithms in simulation before implementing them on the physical system. This simulator will be very useful in the next semester.

6.6. Weak Points

6.6.1. Need more stable localization data for the trajectory following

Our localization algorithm now can provide very accurate position information while the quadrotor does not rotate. If the quadrotor does rotate, the position information is not accurate anymore. This is because the transform matrix between the frame of the deck and the frame of the quadrotor has proven to be difficult to determine. Another issue is that once the localization algorithm loses the view to the beacons, the information feedback is wrong. We plan to implement a tracking algorithm to help with this in the spring.

6.6.2. Need to fine tuning the PID controllers

We have implemented three PID controllers for position control in the x, y, and z directions. In the spring, we will fine tune the controller gains of each PID controller.

6.6.3. Camera does not work on the single board computer

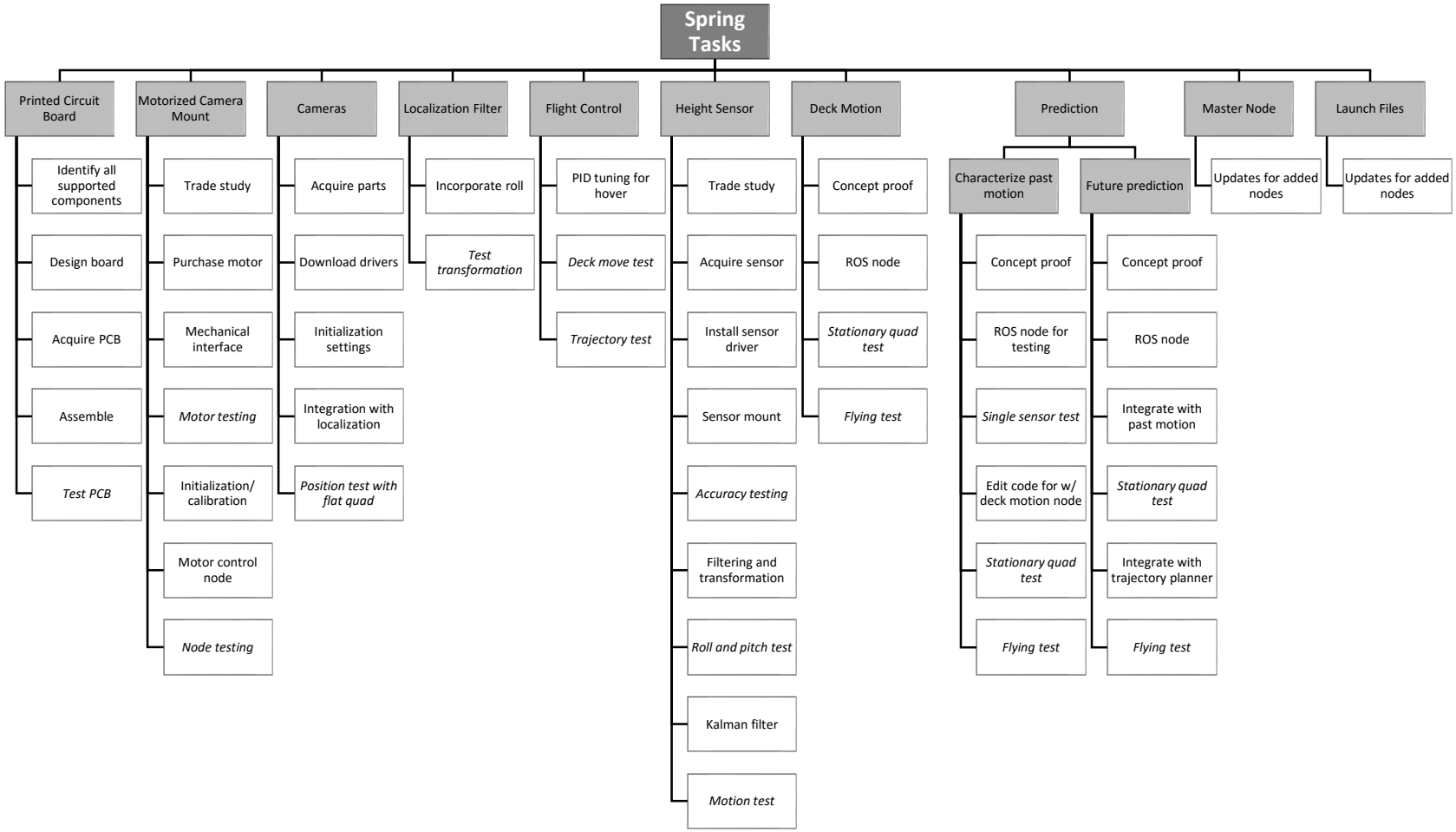
Our cameras work very well on the laptop. However, they do not work on our single board computer. To solve this issue we plan to either change the camera or change the Odroid if we cannot find a suitable driver.

6.6.4. Gimbal does not work

We planned to use Gimbal to maintain the angle of the camera. However, the cable is very stiff and tends to retain its shape, and the motor on the Gimbal does not provide enough power to adjust the Gimbal. To solve this issue, we will build our own Gimbal or fix-mount the cameras.

7. Project Management

7.1. Work Breakdown Structure



7.2. Schedule

The table below shows task breakout by category and the corresponding due dates for each task.

Task ID	Category	Task	Sub-Task	Estimated Hours	Predicate Task IDs	Due Date
5	Motorized Camera Mount	Trade Study for Motor		2	-	11-Jan-16
10		Purchase Motor		0.5	5	11-Jan-16
20		Design Mechanical Interface	CAD draft	8	5	11-Jan-16
25			CAD updates for actual part	1	10, 20	PR 8
30			3D Print	1	25	PR 8
40			CAD updates	2	30	PR 8
50			Machine parts	4	40	PR 8
60		Motor Testing	Camera load testing	2	50	PR 8
70			Encoder Accuracy	1	60	PR 9
80		Initialization/calibration		8	70	PR 9
90		Write motor Control Node		4	80	PR 9
100		Node testing		1	90	PR 9
110	Cameras	Contact Point Grey about 32bit architecture		4	-	23-Dec-15
120		Spec new parts - trade study		2	110	4-Jan-16
130		Acquire parts		0.5	120	11-Jan-16
140		Download drivers		2	130	PR 8
150		Determine initialization settings		2	140	PR 8
160		Integration with localization		2	150	PR 8
170		Position test with flat quad		1	160	PR 8
180	Localization Filter	Fix calculation to incorporate quad roll		4	-	11-Jan-16
190		Test transformation		1	180	PR 7
200	Flight Control	Change PID node to fix subscription timing		8	-	PR 7
205		Test still hover		2	200	PR 7
210		Test deck movement effects on hover		1	210	PR 7
220		PID test with a short trajectory		4	220	PR 8
230	Height Sensor	Trade study		2	-	3-Jan-16
240		Acquire sensor		0.5	230	11-Jan-16

Task ID	Category	Task	Sub-Task	Estimated Hours	Predicate Task IDs	Due Date
250		Install sensor driver		2	250, 130	PR 7
260		Sensor Mount		4	240	PR 8
270		Accuracy testing		2	250	PR 7
280		Filtering and IMU orientation transformation		4	270	PR 8
290		Roll and pitch testing		2	260, 280	PR 8
300		Kalman filter integration with camera data		8	290	PR 9
310		Motion test		1	300	PR 9
320	Deck Motion	Concept Proof in Matlab		4	-	11-Jan-16
330		Build Node in ROS		4	320	PR 9
340		Test with stationary quad and known deck motion		2	330, 300	PR 9
350		Test with flying quad and known deck motion		2	340	PR 10
360	Prediction	Past Motion Characterization	Concept Proof in Matlab	4	-	11-Jan-16
370			Write ROS node to test sensor data	4	360	PR 8
380			Test with single sensor data	2	370	PR 8
390			Edit code for use with deck motion node	4	380, 330	PR 9
400			Test with stationary quad and known deck motion	2	390	PR 10
410			Test with flying quad and known deck motion	2	400	PR 10
420		Future Prediction	Concept Proof in Matlab	4	-	11-Jan-16
430			Write ROS node	4	420	PR 8
440			Test with single sensor data	4	430	PR 8
450			Integrate with past motion characterization	2	440, 380	PR 8
460			Test with stationary quad and known slow deck motion	2	450, 390	PR 9
470			Test with flight and progressively faster deck motions	4	460	PR 11
480			Integrate with trajectory planning	8	470	PR 12
490	Master Node	Updates for new nodes		8	(ongoing)	(ongoing)
500	Launch files	Updates for new nodes		4	(ongoing)	(ongoing)

7.3. Test Plan

7.3.1. Progress Review Capability Milestones

Progress Review 7	Localization node gives correct relative position and rotation at all tilt angles
Late January	Quad can hover in place using camera information about the deck
	Quad maintains relative position to deck when deck is translated along the ground
	Accuracy and limitation testing of height sensor
Progress Review 8	Quad can execute a short trajectory using camera information to orient itself
Mid-February	Past motion characterization can be done with data from one sensor
	Height sensor gives correct height during roll and pitch
	Cameras and localization work on single board computer
Progress Review 9	Functionality of ROS control of motor for camera mount
Late February	Printed Circuit Board Rev C assembled and functional
	Functionality of height sensor on single board computer
	Landing possible on stationary deck
Progress Review 10	Functionality of motor mount control node - response to movement of deck in camera frame
Mid-March	Deck motions are accurately calculated
Progress Review 11	Past motion characterization can be done for deck motions
Early April	Future ideal landing times can be predicted
Progress Review 12	Trajectories are generated to meet correct location at correct landing time
Mid-April	

7.3.2. Spring Validation Experiment

A. Objective: Execute an autonomous landing of a quadcopter on a dynamic mock ship deck with three degrees of freedom (roll, pitch, swell).

B. Elements to be tested:

Vision/localization subsystem
Prediction subsystem
Autonomous landing
Safety features

C. Requirements of relevance/significance to the test:

Functional Requirements
Rotorcraft shall sense position of deck in relation to the rotorcraft with accuracy as function of distance as follows: 1. Detection at minimum 100m distance 2. <100m distance : 25cm + 0.4cm/m allowable error for each meter away

Rotorcraft shall land on dynamic deck moving in swell.
Rotorcraft shall land within 1 foot of center of the deck, as measured from the center of the rotorcraft.
Non-Functional Requirements
Operation of autonomous landing mode shall be possible without radio commands and in GPS-denied conditions.
Autonomous landing sequence shall execute after a single user "go" command.
System shall operate without damage to rotorcraft or deck.

D. Methods

Rotorcraft shall sense position of the deck:

1. Power on all beacons on the mock deck
2. Power on the on-board computer on the DJI and all sensors
3. Plug in a laptop running sensor testing framework
4. Without turning on the DJI motors, hold the DJI with the camera facing the deck
5. Starting at 100m away from the deck, carry DJI towards deck
6. Show on the laptop that the cameras are detecting the deck from 100m away, with at least 0.4cm accuracy per meter distance

Rotorcraft shall land on dynamically moving deck:

1. Power on all beacons on the mock deck
2. Power on actuation of mock deck
3. Power on the on-board computer on the DJI and all sensors
4. Power on the DJI motors and controller
5. Manually operate the DJI to optimal starting position relative to the deck
6. Send “go” command
7. Rotorcraft will then autonomously fly to the deck and land on it
8. Safety operator will be standing by to take back manual control of the DJI if conditions become unsafe for any people or equipment

Rotorcraft shall land within 1 foot of the center of the deck:

1. Once completing the previous portion of the test, the rotorcraft will have landed on the deck
2. There will be a circle of radius 1 foot painted on the deck. Ensure that the center of the rotorcraft is within this circle by inspection

Demonstrate Prediction Algorithm:

1. Display graphs of pre-recorded data showing that the results of the prediction algorithm over time

E. Logistics of testing

Date: 03 December 2015

Location: NSH Level B

Equipment and supplies:

- Rotorcraft
- Sensor Suite
- Deck
- Spare Battery & Hot Swap Port
- Laptop

7.4. Budget

Table 2 below shows a summary of the budget for the project. To date, we have spent about two thirds of the total MRSD budget. The big ticket items that remain are a new single board computer, a rework of our power distribution board, and potentially an actuated camera mount.

Table 2: Budget summary for project.

	MRSD Budget	**Sponsor Budget**	**TOTAL BUDGET**
Initial	$ 4,000.00	$ 5,000.00	$ 9,000.00
Quadrotor (with spares)	$ (700.00)	$ (800.00)	
Optics	$ (1,230.00)		
Computation	$ (378.00)		
Other	$ (543.58)		
Spent to Date	**$ (2,851.58)**	**$ (800.00)**	**$ (3,651.58)**
Remaining to Date	**$ 1,148.42**	**$ 4,200.00**	**$ 5,348.42**
Antipicated Additional Spend:			
Computation	$ (500.00)		
PCB	$ (250.00)		
Actuated Camera Mount	$ (200.00)		
Actuated Deck		$ (500.00)	
Total Additional Spend	**$ (950.00)**	**$ (500.00)**	**$ (1,450.00)**
TOTAL ANTICIPATED REMAINING	**$ 198.42**	**$ 3,700.00**	**$ 3,898.42**

7.5. Risk Management

Risk #	Risk	Type	Description	Likelihood	Consequence	Likelihood Reduction Plan	Consequence Reduction Plan
1	Unstable Autonomous Flight	Technical, Schedule, Budget	Autonomous landing mode does not control flight adequately/correctly, causing DJI to maneuver in unsafe ways.	4	2	RC switch for manual override	Order extras of cheaper parts
2	Light interference in vision system	Technical	Vision system identifies non-beacon source of light as beacon, causing localization to give inaccurate positions causing erratic flight behavior.	5	4	Infrared pass filter, testing indoors	RC switch for manual override
3	Inadequate near vision	Technical	Vision system field of view cannot detect deck location/orientation at short distances	2	4	Off-center location of beacons with compact orientation	Feed-forward landing tests Additional sensors
4	Asynchronous timing	Technical	Algorithms do not correctly interpret timing of signals	2	5	Create publish rate dependencies on subscribing nodes	Early implementation for troubleshooting
5	Test location unavailable	Schedule	Testing is unable to be done at the planned location due to distance, access limitations, timing, or weather conditions.	2	3	Early coordination with sponsor	Construction of small test platform to be modular with large test platform
6	Payload too heavy	Technical	Weight of sensors and circuitry is too heavy for DJI to perform flight maneuvers effectively	1	5	Weight tracking of components and payload testing	Identify and limit unnecessary weight
7	Printed circuit board nonfunctional	Schedule, technical	Printed circuit board not designed correctly, resulting in rework, unpowered parts, or damaged components	4	4	Perform all component planning prior to ordering board Get board reviewed by TA or professor	Define and order needed parts early
8	Inaccurate prediction	Technical	System is unable to predict an appropriate and executable landing time	2	4	Early development of prediction algorithm to detect knowledge gaps and technical issues	Manual override to prevent system from draining battery Identify alternative prediction model
9	Microcontroller dies during flight	Technical	Microcontroller for forwarding flight commands loses functionality during flight, causing DJI to crash	2	5	Robust testing in controlled environment	Order spare parts
10	Incompatibility between single board computer and ROS architecture	Technical	Software drivers for hardware components are not compatible with single board computer	5	4	Investigate OS requirements of hardware prior to purchasing and development	Reserve budget for acquiring alternative sensor or SBC

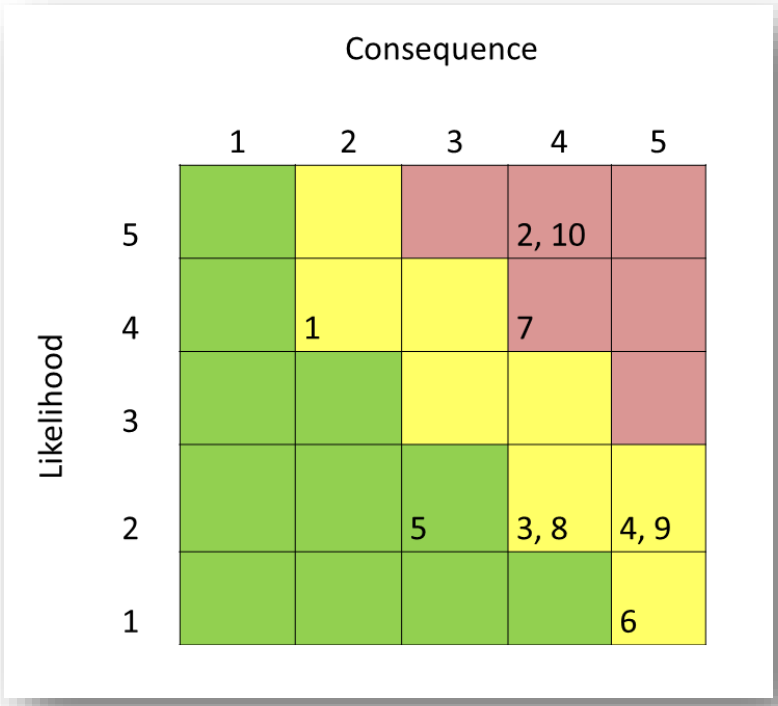


Figure 7: Consequence/likelihood risk chart.

8. Conclusions

8.1. Key Fall semester lessons learned

The first lesson we learned from this project is the underestimation of the workload. With our limited experiences, we were not able to accurately estimate the task length and workload. Uneven workload distribution results in blocking actions and delay schedules.

After the project definition with our sponsor and MRSD advisors, we only have one month left to finish our task. However, we promised to finish a prototype that can land on a static deck in Fall semester. We should have a more realistic goal. Instead of promising to have all subsystems integrated together, we should define our requirements on each individual subsystem first.

Most of our team members don't have enough knowledge in software development. We spent a great amount of time learning ROS, github, and single board computer. In the perspective of learning, we did a great job. In the perspective of the project development, this results in delay of the development and a great burden on the software person in our team. In addition, we should have start our version control before any software development, especially syncing all the results with our single board computer. Due to the delay arrival of our single board computer, the

compatibility issues between our laptops and the single board computer already pile up by the time we integrate our codes into it.

A majority of our work has already been done in this field. Our sensors and algorithms are all well developed. Yet we didn't ask for help often enough. We have a great sponsor that can provide us technical support but we only report our progress during weekly meeting. By the time we realized we need help, it was already too late to make any changes due to the deadline of FVE.

We made our schedule too tight to account for any unexpected problems. There are many tasks that is hard to foresee. Our management needs to be more agile to meet our goals. For example, we promised to demonstrate the quadrotor flying using the integration results of the trajectory generation and flight control subsystems at the 4th progress review. To meet this goal, we not only need the software to work with each other, but also need the single board computer, the power distribution system, and the sensor mount to be ready to use. However, we didn't plan these tasks in our schedule. Our solution to this problem was to plan a mini schedule for each member to work on these unscheduled tasks. The results were pretty successful. We need to employ this method more frequently.

Although we have team meetings very frequently and we all know the progress of each subsystem. But when one subsystem goes wrong, we still need that person to fix it. We initially assigned two people responsible to one subsystem but we didn't implement it. This results in unnecessary delay to our schedule. This problem should be avoided in the next semester.

8.2. Key Spring semester activities

The first thing we need to do for the next semester is to reevaluate our choice of the camera and the single board computer. If the camera driver is still not compatible with the single board computer, we might need to make some corresponding changes. After this problem is fixed, we need to change the design of the power distribution system. In addition, since we are using USB hub to provide more ports for our sensors and microcontrollers, we also need to consider using a powered hub and design a corresponding power outlet from our power distribution system.

For the software development part, we need to integrate our code more often and install all required software and libraries on everyone's laptop and the single board computer.

Most of our subsystems need to upgrade for future challenges. Navigation strategy needs to account for the possibility of losing sight of the beacons. Prediction algorithm needs to be employed to give better state estimation and sensing results. The performance of the vision subsystem will be improved by fusing with inertial navigation techniques.

9. References

[1] http://www.webdesignschoolsguide.com/library/10-things-we-couldnt-do-without- robots.html

[2] https://upload.wikimedia.org/wikipedia/commons/7/74/US_Navy_091031-N-1251W-009_An_SH-60B_Seahawk_helicopter_assigned_to_Helicopter_Anti-Submarine_Squadron_Light_(HSL)_51_approaches_the_guided-missile_destroyer_USS_Lassen_(DDG_82)_during_deck_landing_qualifications.jpg