BRADLEY University

Formation Control of Crazyflies

Bryce Mack, Chris Noe, and Trevor Rice

Advisors: Dr. Ahn, Dr. Wang

November 30, 2017

Table of Contents

Introduction

- UAVs have attracted significant attention over the past five years
- We are exploring distributed control for multiple UAVs in formation
- Sensing/communication among individual UAVs and how to design simple yet efficient local control strategies for each UAV
- Design practically implementable distributed control algorithms for UAVs and implement using an agile nano quadcopter, the Crazyflie

Motivation

- Lots of UAVs to choose from
- Chose the Crazyflie
  - Agility, durability and programmability
  - Indoor use, quick charge

Problem Statement

- Using a Crazyflie 2.0 by Bitcraze (open source hardware/software)
    - 5-10 minute flight time
- All hardware development has been done by Bitcraze
- Software will be developed using the ROS environment

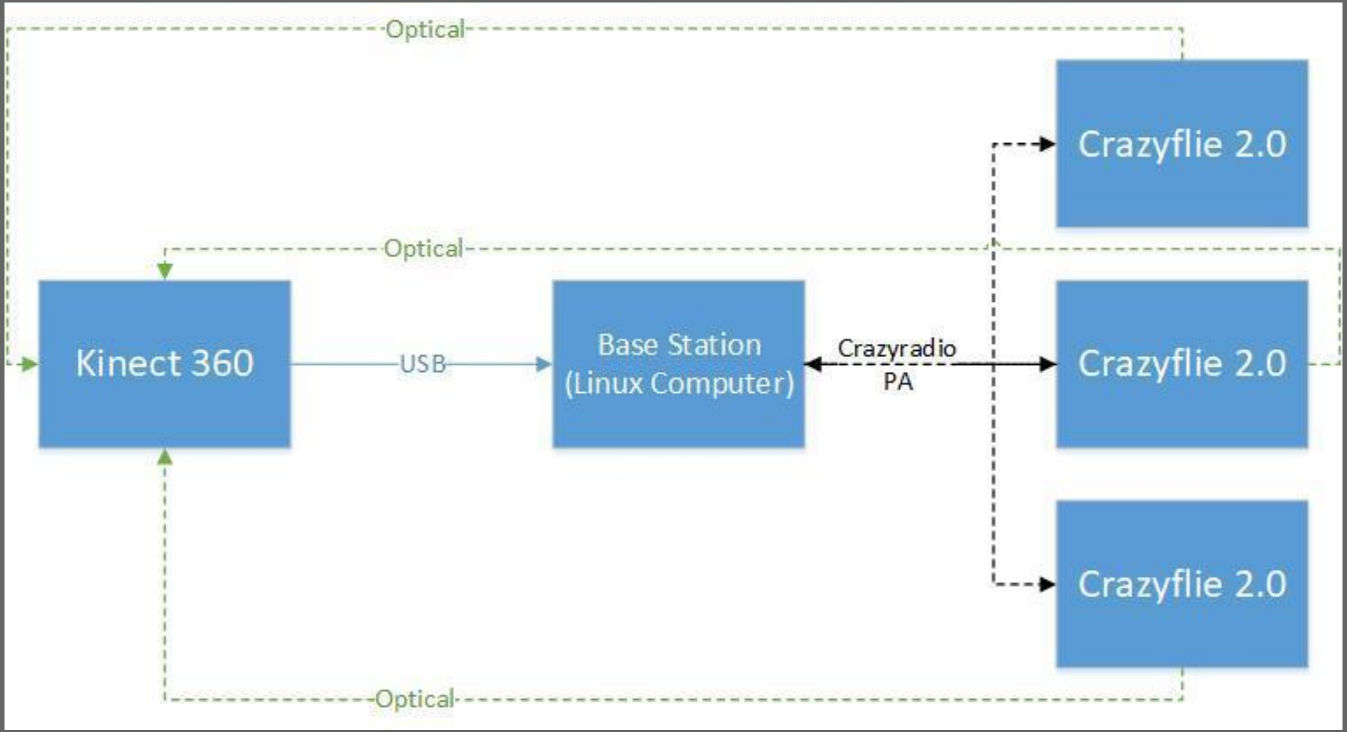
Design Tasks

I. Modeling and Control Design
II. Control Implementation Using ROS
III. Localization Using Kinect and LOCO
IV. System Integration and Formation Control Implementation

Functional Requirements

- Programming and control through the Crazyradio PA
  - USB Radio Dongle
- Control code will be based in C/C++
  - Compiled in ROS on Ubuntu 14.04 Trusty
- Localization by Kinect and/or Loco Positioning System
- Control algorithm will be run on the on-board chips
- See High Level System Diagram on the next slide

High Level System Diagram



Specifications

- Weighs 27 g
- Size (WxDxH): 92x92x29mm (motor-to-motor and including motor mount 4 Figure 1: Crazyflie 2.0 feet)
- 20 dBm radio amplifier tested to more than 1 km range LOS with Crazyradio PA
- STM32F405 main application MCU (Cortex-M4, 168MHz, 192kb SRAM, 1Mb flash)
- nRF51822 radio and power management MCU (Cortex-M0, 32Mhz, 16kb SRAM, 128kb flash)
- IMU: 3-axis gyro, accelerometer, and magnetometer
- Max recommended payload weight: 15 g

Parts List

Bitcraze Components

- 6 x Crazyflie 2.0
- 3 x CrazyRadio PA
- 1 x Z-Ranger Deck

LOCO Positioning System

- 6 x LOCO Anchors
- 6 x Anchor Power Supply
- 6 x 3D Printed Anchor Brackets
- 1 x LOCO Crazyflie Deck

Xbox Components

- 3 x Xbox 360 Kinect
- 3 x Xbox 360 Stand
- 3 x Xbox 360 Kinect Power Supply
- 1 x Xbox ONE Kinect

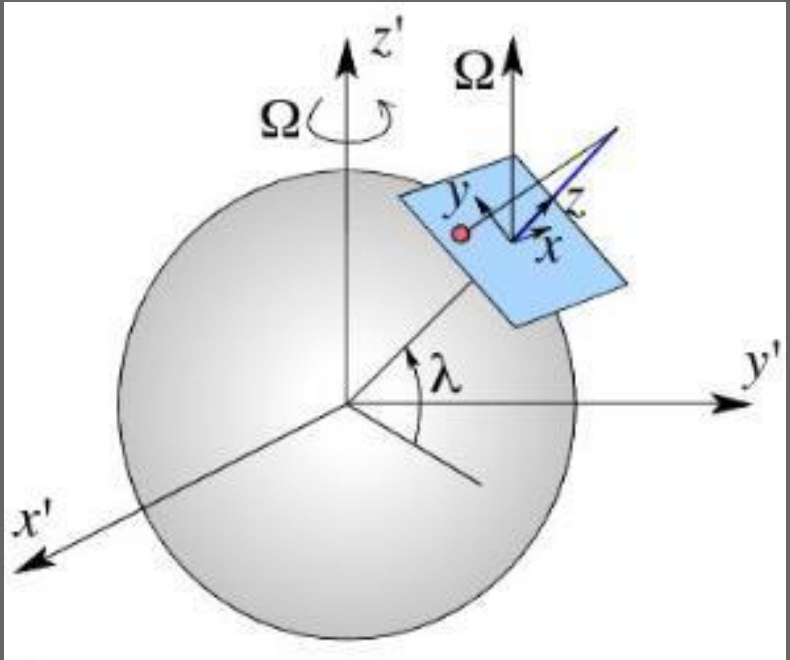
Laptop Running Ubuntu 14.04 Trusty

Total Cost: $2265

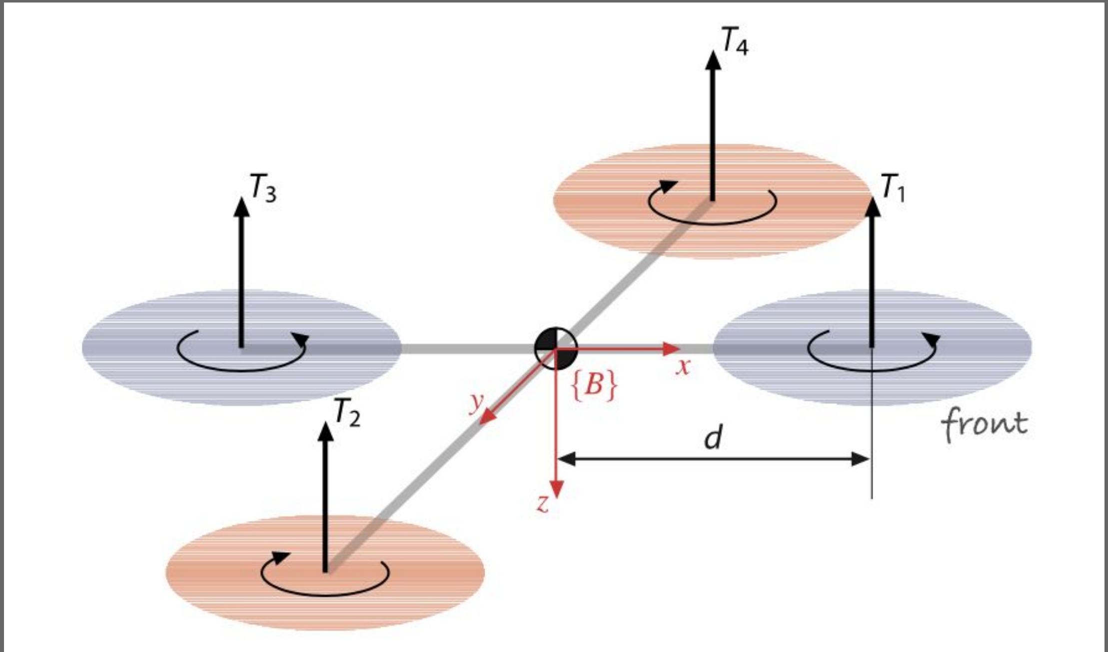
RESEARCH TASK 1: MODELING AND CONTROL DESIGN

Quadrotor Coordinate System

- The inertial frame designates Z to be any direction coming out of the earth
- The body frame of the quadrotor designates Z to be into the earth.



Quadrotor Body Frame



Model Equation

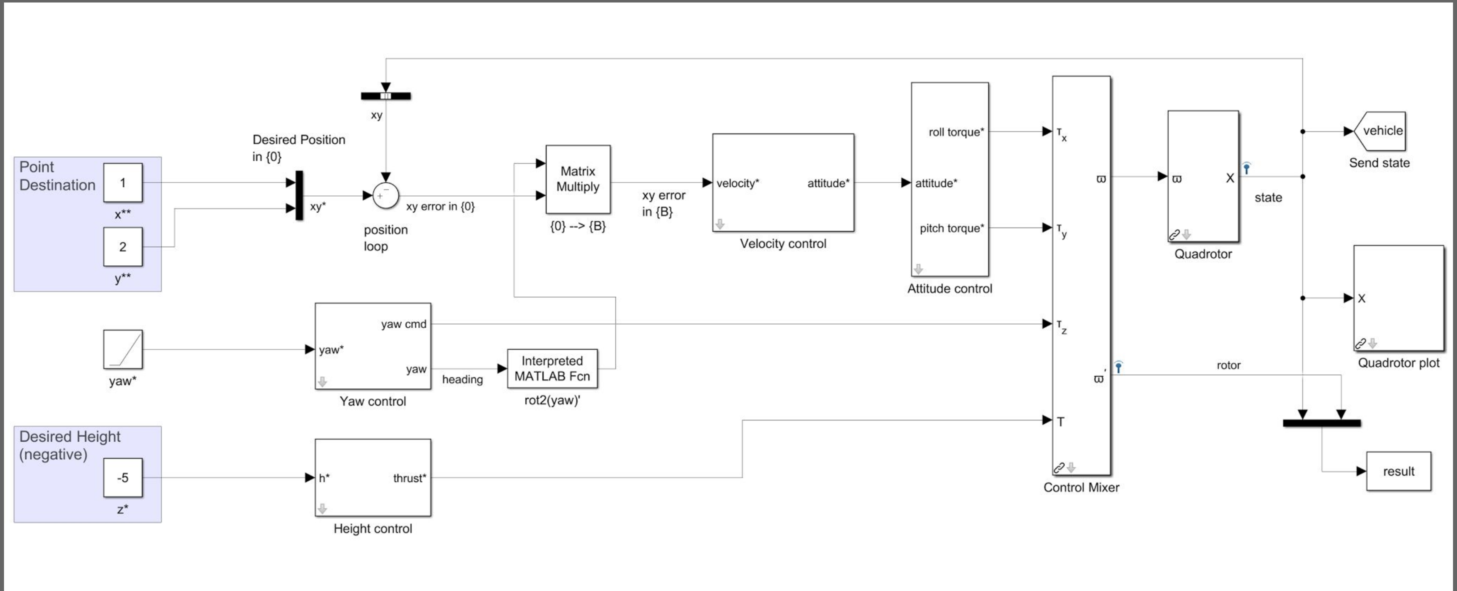
- *Adaptive Control of Quadrotor UAVs: A Design Trade Study With Flight Evaluations*
  - By Zachary Dydek
- Using small angle approximations Eqn. 1 becomes Eqn. 2

$$
\begin{aligned}
\ddot{x} &= (\cos\phi \sin\theta \cos\psi + \sin\phi \sin\psi)\frac{U_1}{m} \\
\ddot{\phi} &= \dot{\theta}\dot{\psi}(\frac{I_y - I_z}{I_x}) - \frac{J_R}{I_x}\dot{\theta}\Omega_R + \frac{L}{I_x}U_2 \\
\ddot{y} &= (\cos\phi \sin\theta \sin\psi + \sin\phi \cos\psi)\frac{U_1}{m} \\
\ddot{\theta} &= \dot{\phi}\dot{\psi}(\frac{I_z - I_x}{I_y}) - \frac{J_R}{I_y}\dot{\phi}\Omega_R + \frac{L}{I_y}U_3 \\
\ddot{z} &= -g + (\cos\phi \cos\theta)\frac{U_1}{m} \\
\ddot{\psi} &= \dot{\phi}\dot{\theta}(\frac{I_x - I_y}{I_z}) + \frac{1}{I_z}U_4
\end{aligned}
\tag{1}
$$

$$
\begin{aligned}
\ddot{x} &= g\theta \\
\ddot{\phi} &= \frac{L}{I_x}U_2 \\
\ddot{y} &= -g\phi \\
\ddot{\theta} &= \frac{L}{I_y}U_3 \\
\ddot{z} &= \frac{\Delta U_1}{m} \\
\ddot{\psi} &= \frac{1}{I_z}U_4
\end{aligned}
\tag{2}
$$

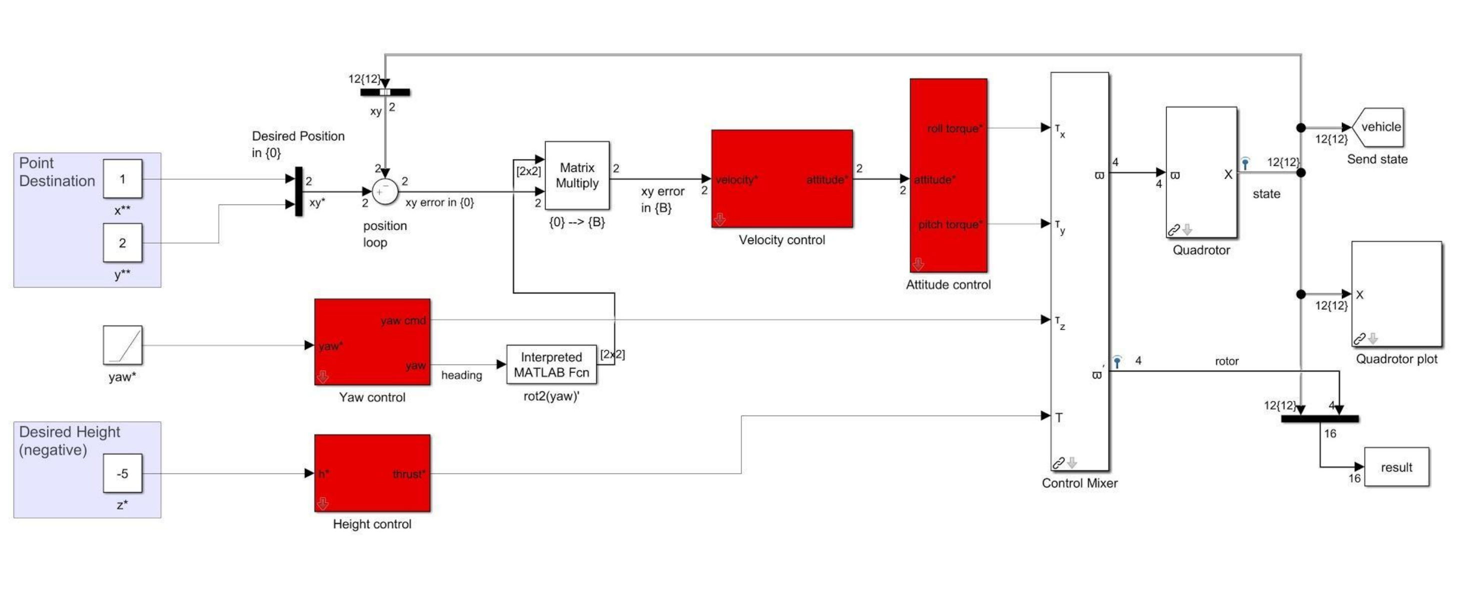
Simulink Modeling

- Installed MATLAB Robotics, Vision and Control Toolbox developed by Peter Corke
- Explored the Quadrotor model that they created



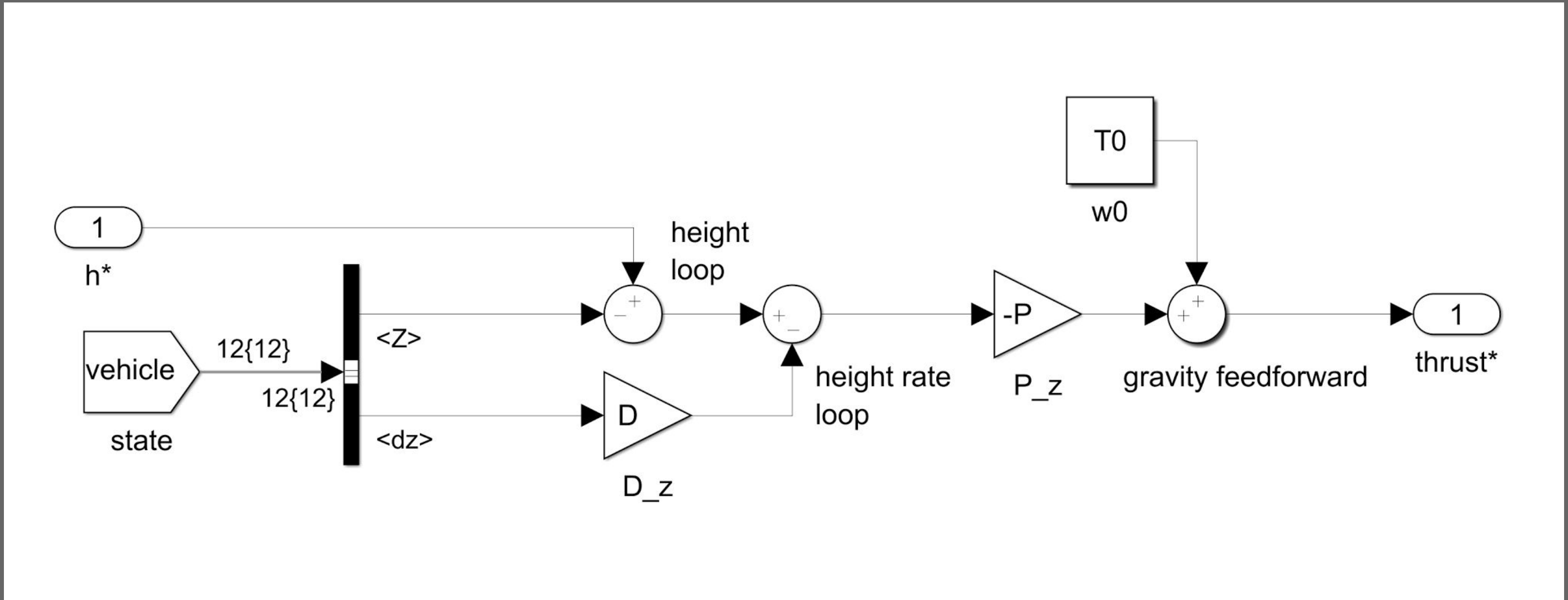
PD Controllers

- 4 PD Controllers: Height, Velocity, Yaw and Attitude



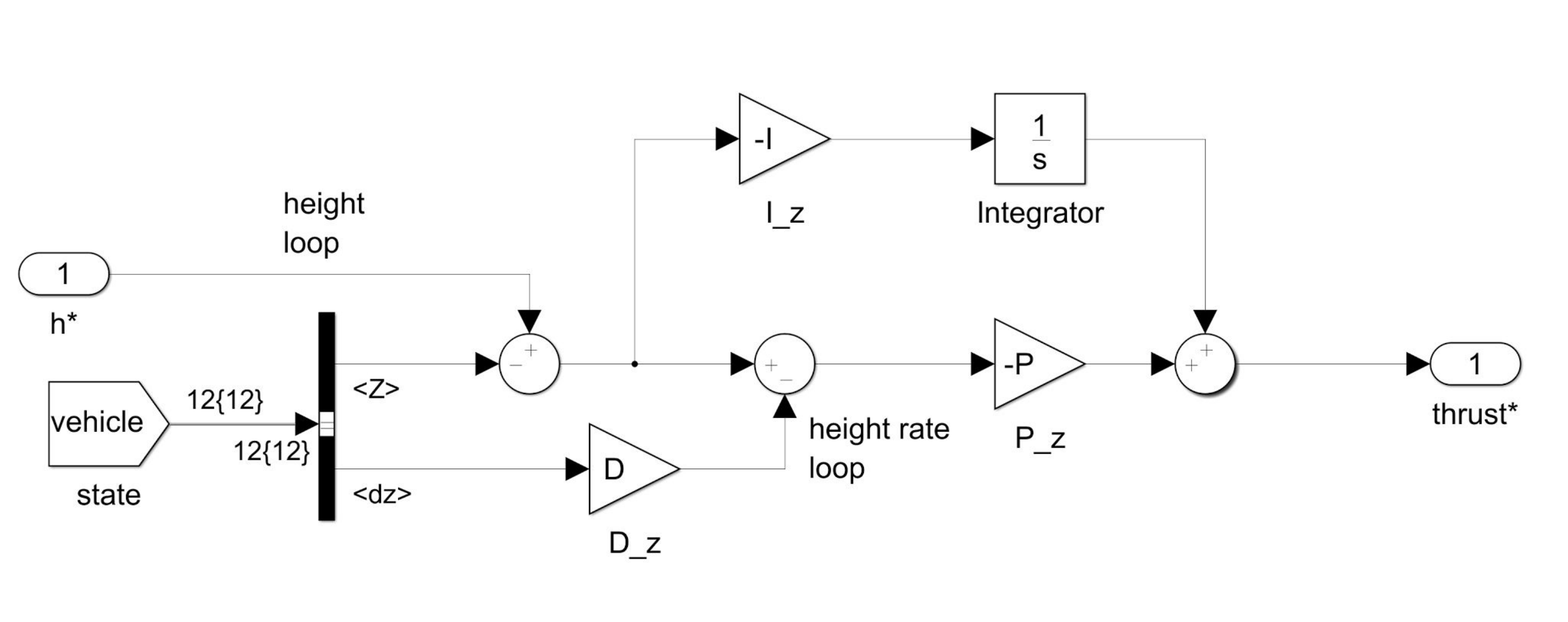
Height PD Controller

- Old model used a feed-forward thrust constant

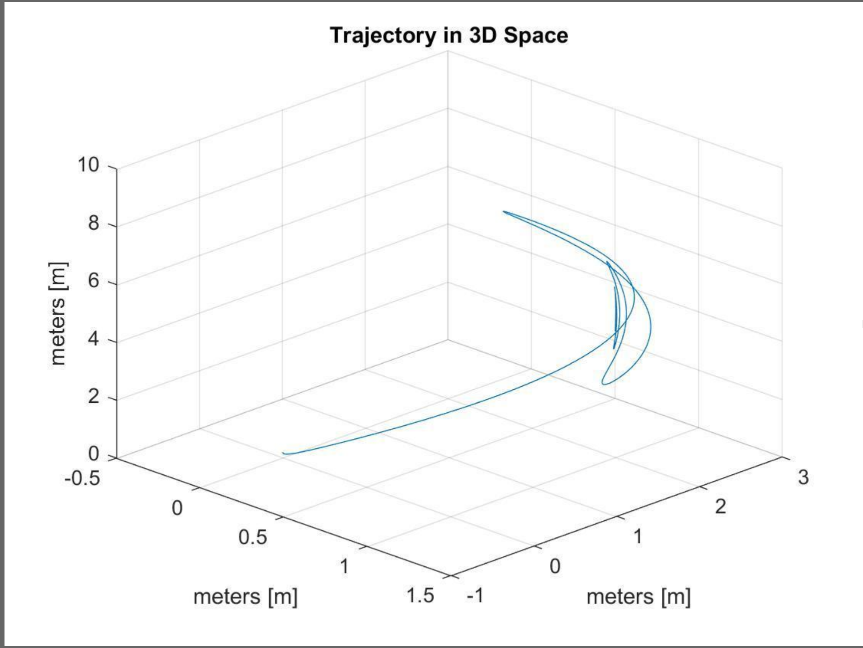


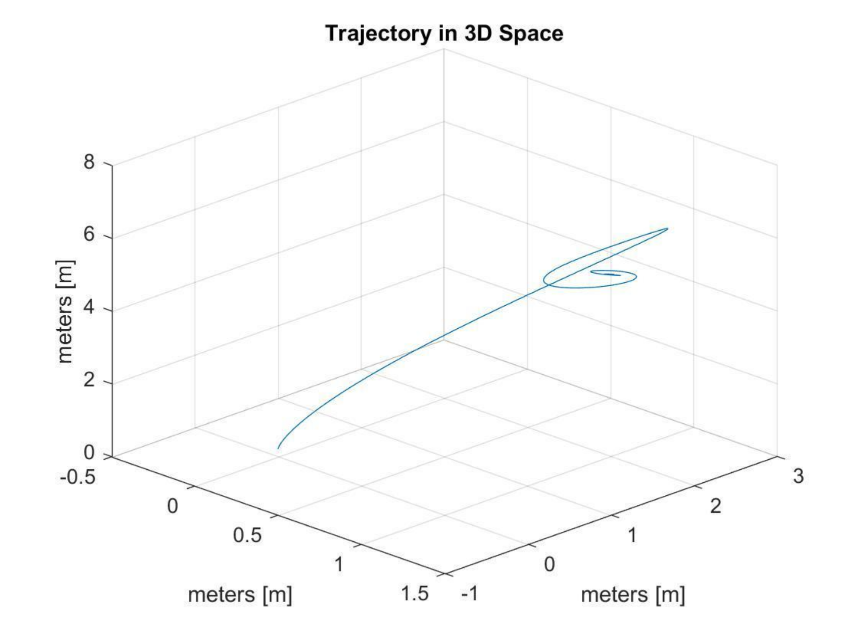
Height PID Controller

- New model replaces feed-forward term with an integral controller

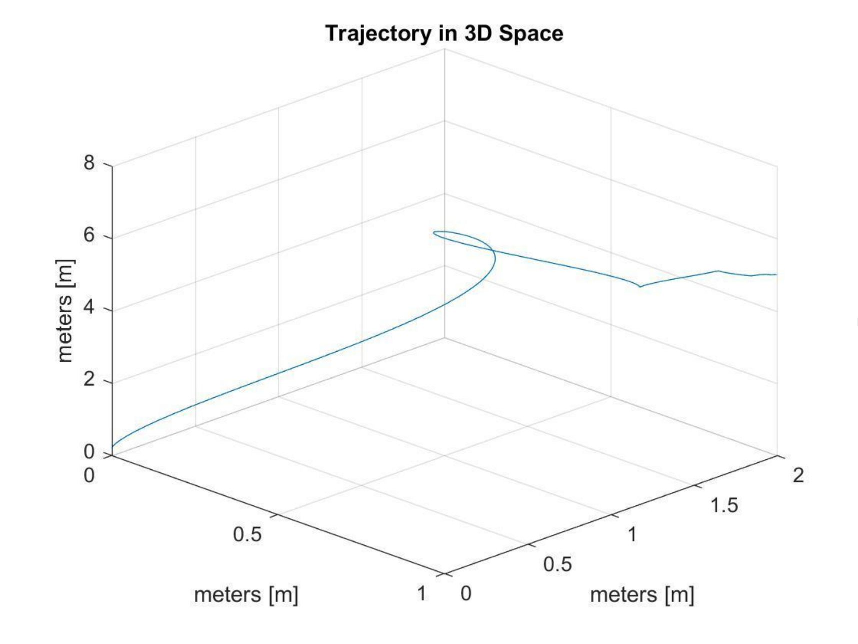


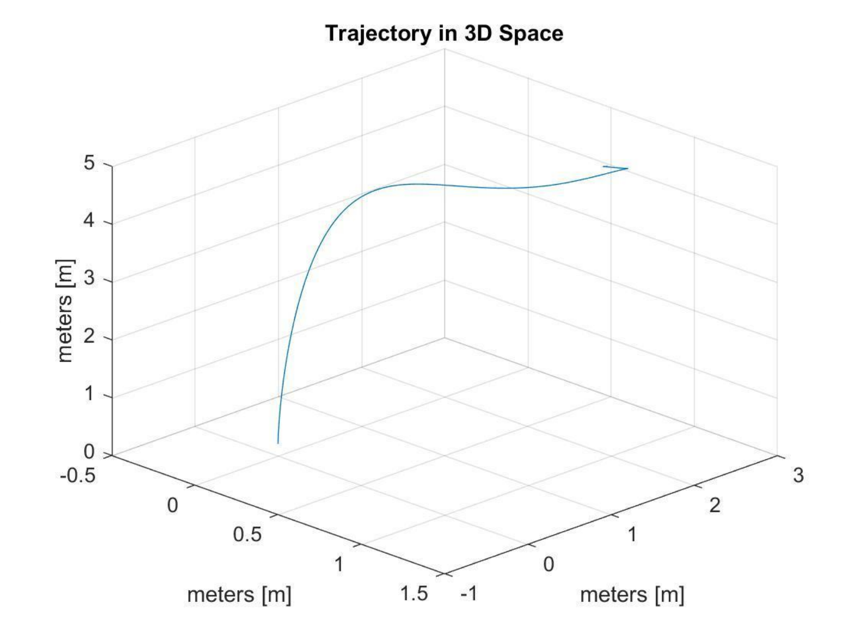
Adjusting Height and Velocity Controls





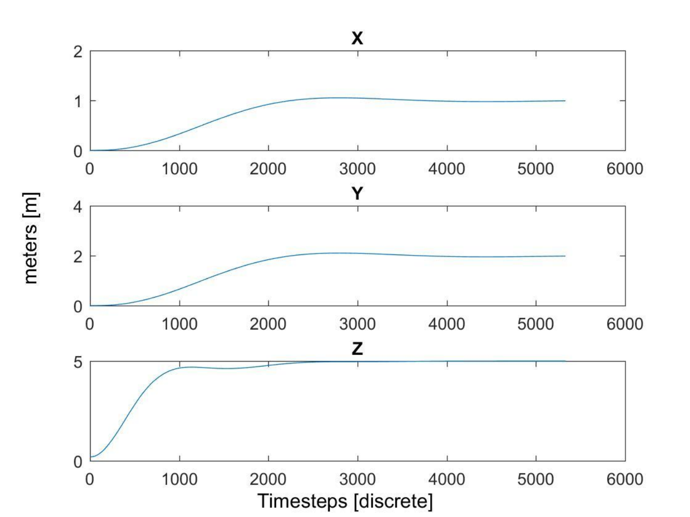
Adjusting Height and Velocity Controls





Simulation Results

- Tuning the controllers allowed us to reduce the overshoot to 6% for X and Y
- We decided for Z to have 0% overshoot
  - Critically damping the system
- We don't want the crazyflie ever crashing into a ceiling was our reasoning
- Z is able to settle within 6.5 seconds



Trajectory Control

- Circle
  - sine and cosine inputs to simulate a circle
    - 3sin(t/8)
    - 3cos(t/8)
- Figure 8
  - 2 sin inputs that create a figure 8
    - 3sin(t/10)
    - 3sin(t/20)
- Square
  - 4 step inputs
  - Each activating after ‘x’ seconds

Circle Trajectory

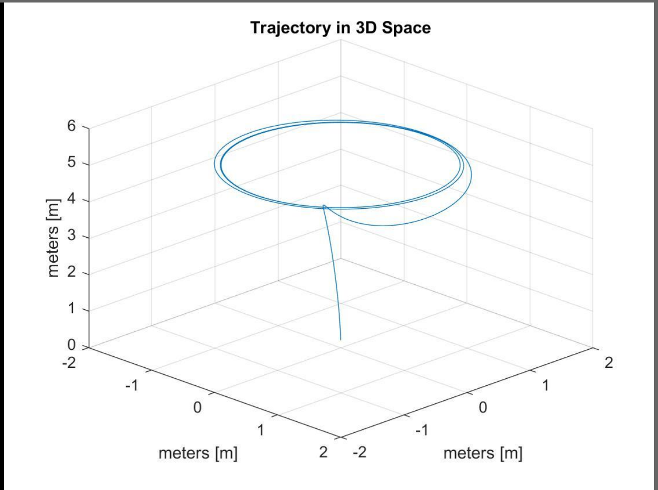
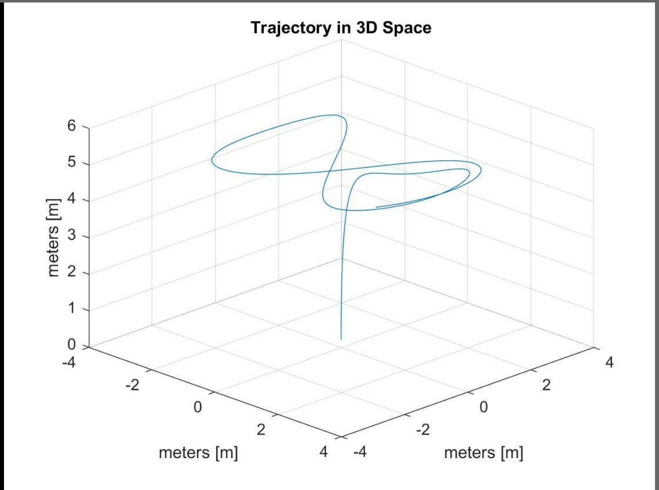
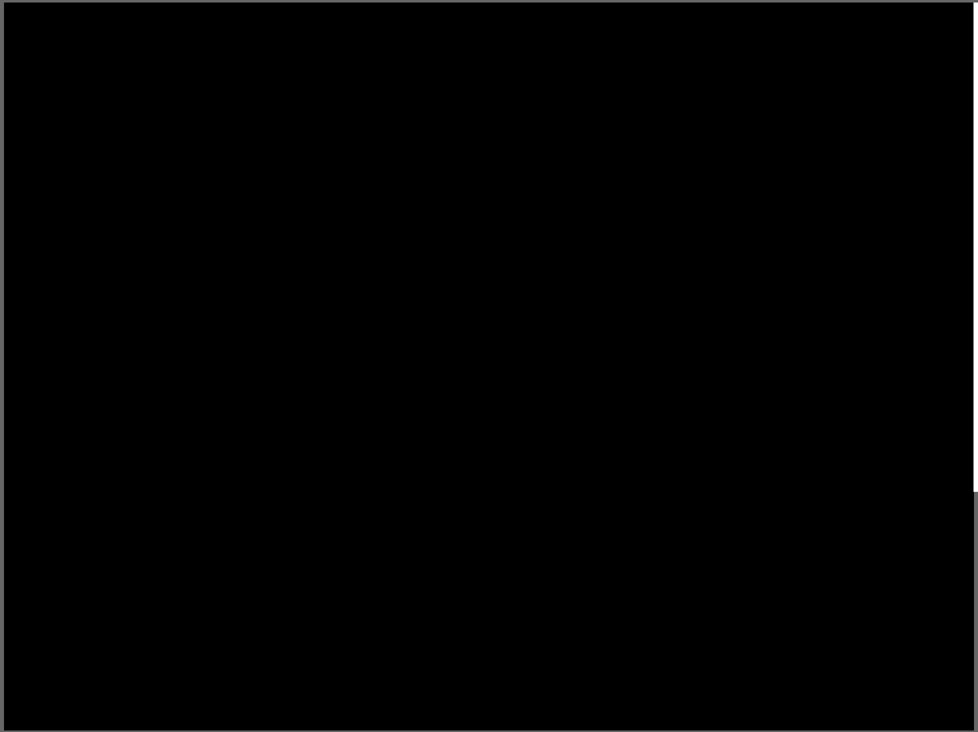
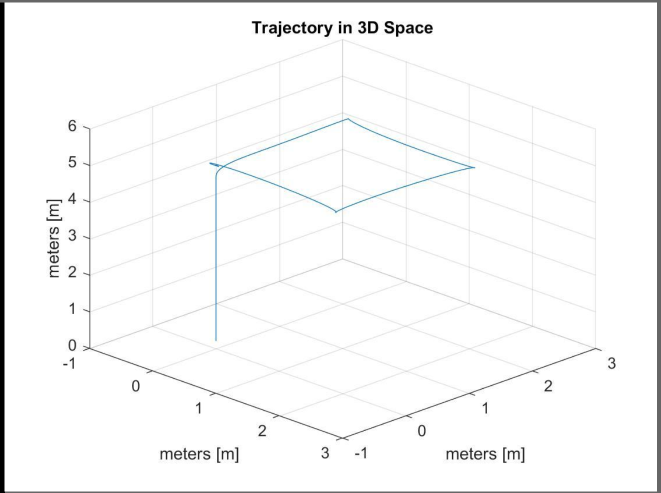


Figure 8 Trajectory



Square Trajectory





RESEARCH TASK II: LOCALIZATION USING ROS

ROS

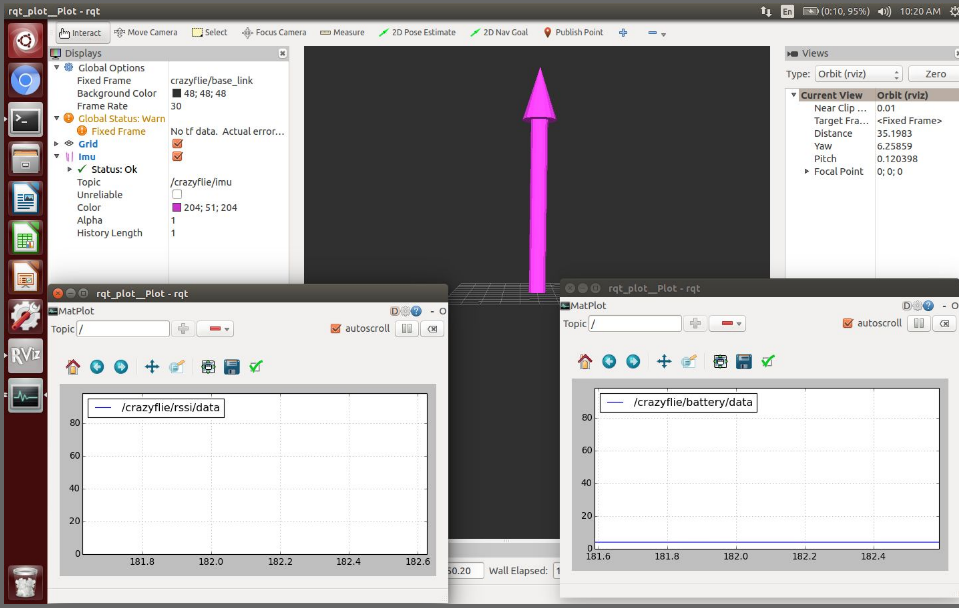
- First thing to do was to research the ROS environment
  - Read through textbook provided by Dr. Wang
- Basics
  - Packages: Contain the information needed to compile the executables
  - Nodes: Small modules of processes that are designed to do a few tasks of the larger more complex program
  - Master: Nodes are able to communicate with other nodes on the same computer and on a different networked computer using the Master. Master provides naming and registration services for the nodes. Master is required to be running for the ROS environment to operate.

Open-Source ROS

- Honig Paper
  - Has developed a custom driver for the Crazyflie 2.0 using ROS
  - However, the driver was designed to use an expensive VICON system for localization
  - We used his driver code to be able to fly the Crazyflie using a PS3 remote
- Complications
  - One of the benefits is also a downside
    - The ROS system can be made extremely modular
    - This modularity can also become overly complex and hard to follow
      - This is the case with the Honig driver
  - The toughest part of the project in terms of ROS will be figuring out how to develop programs, or edit current programs, that will control the Crazyflie

PS3 Control

- Flying with the PS3 remote and Crazyradio PA
  - Used the files from the Honig paper GitHub
  - The demo creates numerous nodes to control different aspects of the Crazyflie
  - The demo opens up RViz to show the status of the Inertial Measurement Unit (IMU) on the Crazyflie
  - There is a possibility of adding more crazyflies to the demo, but that hasn't been tested yet.



Live PS3 Control

Live PS3 Control

RESEARCH TASK III: LOCALIZATION USING KINECT AND LOCO

Kinect

- The Crazyflies have onboard gyroscopes and accelerometers, but incapable of determining their location in 3D space
- We will be using Xbox 360 Kinects to solve the localization issue

Kinect Background

- The Kinect itself has 3 I/O devices: cameras, audio, and motors
- We will only use the camera functions, not audio output or motor input
  - The camera has 2 outputs to ROS
    - RGB camera
    - IR Blaster and monochrome CMOS sensor
- The Kinect operates with 640 x 480-pixel resolution and runs at 30 FPS

Kinect -ing to ROS

- Communication between ROS and the Kinect consists of camera images and position data.
- Kinect will use a color threshold program to locate the Crazyflie in 3D space
- Depth sensor - an infrared projector and a monochrome CMOS (complementary metal-oxide semiconductor) sensor work together to "see" the room in 3-D regardless of the lighting conditions.

Kinect -ing to ROS

- Need ROS to recognize Kinect 360 in order to extract information from the module
- ROS textbook examples used Kinect V2.0
  - Department’s Kinect modules are Kinect 360
  - Has a lower framerate and resolution
    - We will have to determine if it is still a viable way to track the drones
    - We will also have to use older libraries and programs to connect the Kinect to ROS.

Kinect Libraries

- After testing the most recent libraries, it was found that they (libfreenect2) do not communicate with the Kinect 360.
- The older library, libfreenect, will connect to the Kinect 360.
- This, along with fakenect, crazyflie_ws, and ROS we will be able to analyze the data sent through the Kinect and use the data to control our Crazyflie.

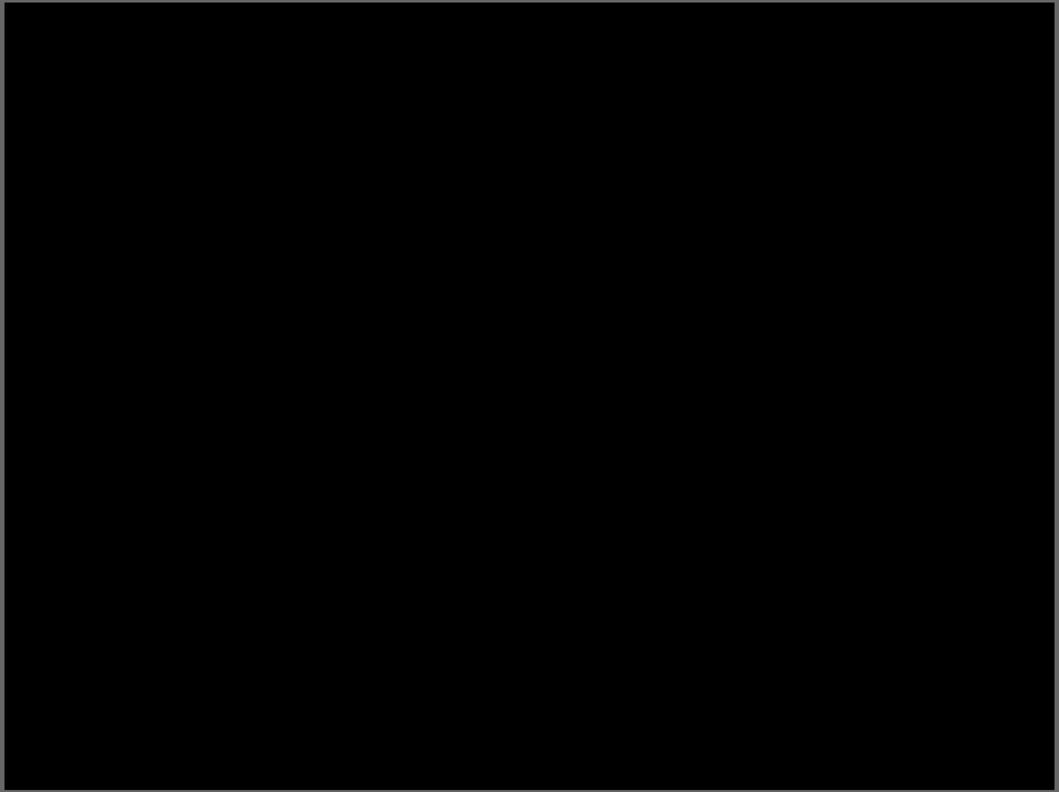
Kinect Localization

- The control algorithm will feed the estimated position coordinates to ROS, which will implement our simulation PD control algorithm. This will create a 3D environment where ROS will estimate where the Crazyflie(s) exist and through different input methods, will fly to where they are told.
    - The position estimation can be expanded to multiple Crazyflies, using different colored markers or a numbering system with the same colored markers.

Kinect Workspace

- Dedicated space in Robotics Lab for Crazyflie operation
- Kinect(s) will be on a stand(s)
  - Minimum distance: 0.5 m
  - Maximum distance: 4.5 m
  - Height: 0-3 m
- Flight area will be defined by the line of sight of the Kinect(s)
- The position estimation can be expanded to multiple Crazyflies, using different colored markers or a numbering system with the same colored markers.

Kinect Control Demo

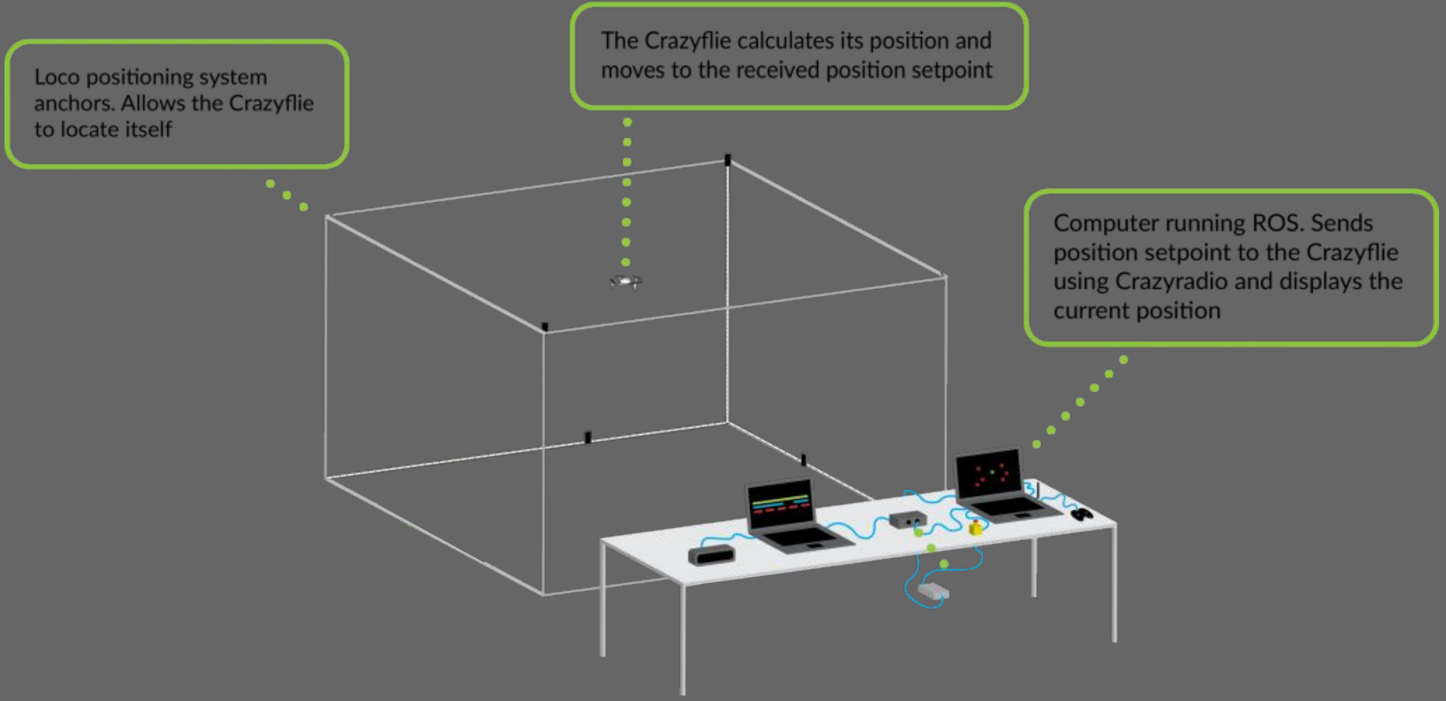


LOCO Positioning System

LOCO Positioning System (cont.)

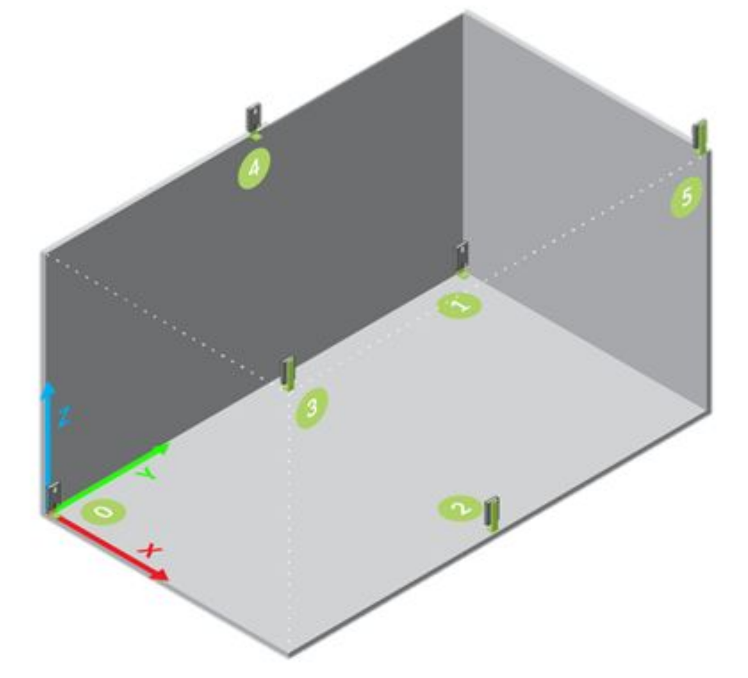
- In essence, a small indoor GPS-like setup
- Anchors around the room (at least 4-6) act as position markers to set up the 3D space
- Nodes are what get tracked through the 3D space
  - One of these nodes goes on the Crazyflie
- This setup might be able to replace the need for camera feedback as the system is accurate to 10 cm
  - Won’t work for tight maneuvers
- Limitation: May only work for one crazyflie at this time. We may have to develop the capability to track more than one crazyflie.

Bitcraze Demo

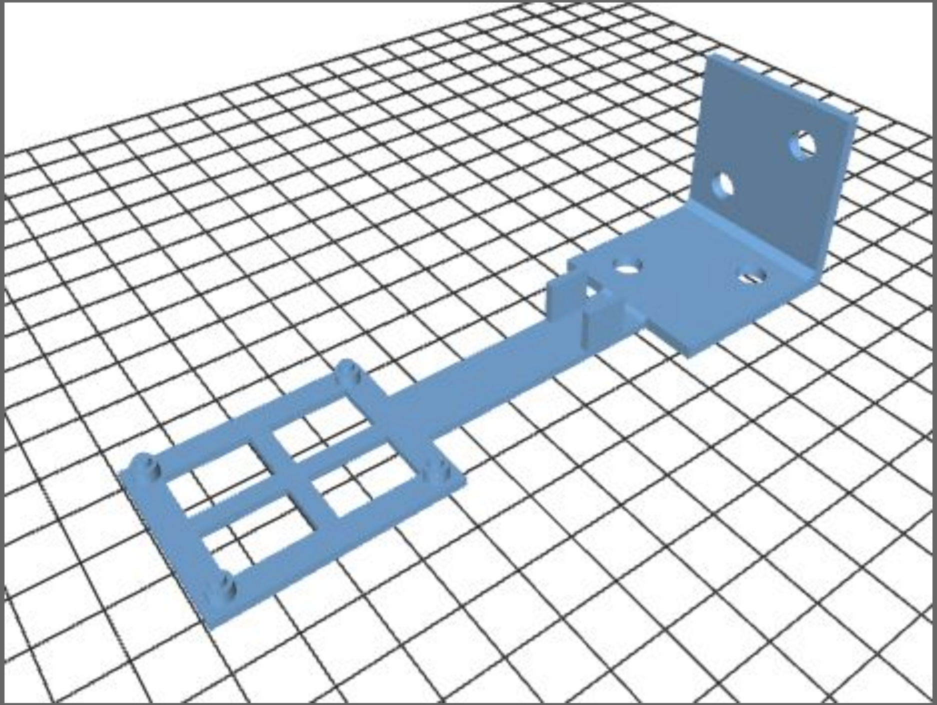


LOCO Setup

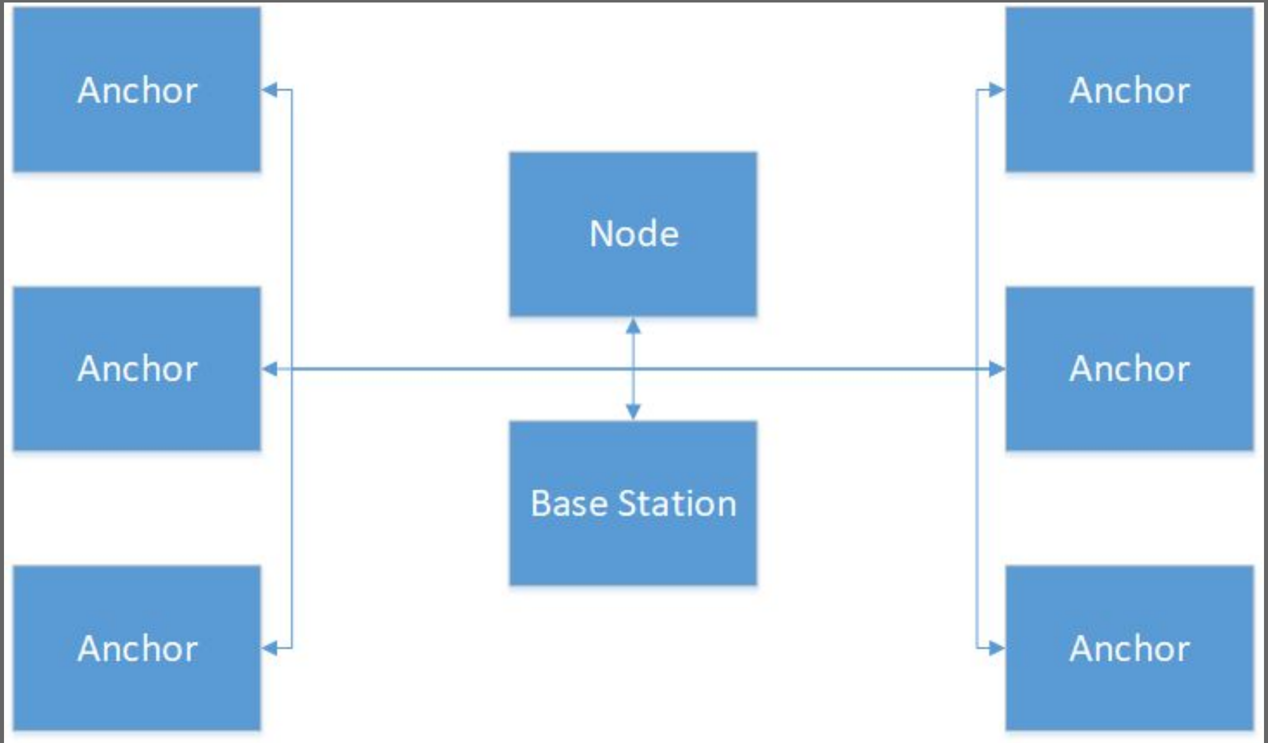
- Workspace will be setup with the anchors in triangle patterns at the top and bottom of the space
- This setup will give us the best accuracy when it comes to 3D localization.



3D-Printed Anchor Bracket



LOCO System Diagram



LOCO vs Kinect

LOCO

LOCO Pros

- Can be used in visually “noisy environments”
  - Open space is not required
- Accurate to 10 cm
- More points of reference to increase general accuracy

LOCO Cons

- Susceptible to radio interference in the populated 2.4 GHz range
- Currently setup for only one Crazyflie
- Expensive if we want to introduce more Crazyflies

Kinect

Kinect Pros

- More cost effective than the LOCO position sensors
- Very minimal physical system requirements
- Random error accuracy: 5mm-4cm (0.5m to 5m)
  - Depth accuracy: 2mm-7cm (1m-5m)

Kinect Cons

- Difficult to implement
  - Little documentation on libraries and packages needed to connect Kinect to ROS
- Multiple programs and systems are needed to run Kinect through ROS
- System has to be built by us
- Unknown if data can be interpreted through ROS

How We Will Proceed

- Start with using the LOCO system
  - Easier to setup
- Progress with Kinect
  - More difficult to integrate with ROS
- Kinect can detect multiple Crazyflies

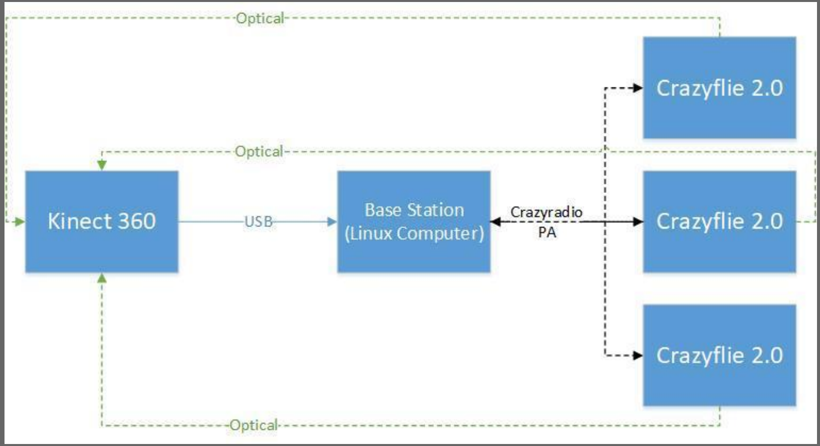
Goals

- Use LOCO to control the “lead” Crazyflie
- Use Kinect to detect the other Crazyflie positions
- ROS will interpret localization data and send Crazyflies new desired positions

Research Task IV: System Integration and Formation Control Implementation

System Integration

- LOCO and Kinect detect position of Crazyflies
- ROS environment accepts inputs from Crazyflie and LOCO/Kinect
- Crazyflies will execute our control algorithm
- ROS will execute distributed control algorithm
- Base station is merely there to provide localization data
    - Crazyflies take data and move to their updated position



Formation Control

- Control will be based on cooperative control theory
- Will begin with modeling in Simulink
  - We can simulate multiple Crazyflies using current model
- Equation below will be run in ROS
  - Base station will provide desired positions for each Crazyflie

$$u_i = \sum_{j=1}^{n} a_{ij}(x_j - x_i) + \sum_{j=1}^{n} a_{ij}(\dot{x}_j - \dot{x}_i)$$

Work Division

Bryce Mack

- Simulink Modeling
  - Create new model that will simulate multiple Crazyflies
    - ➢ Create a subsystem of current model
    - ➢ Implement theory from Cooperative Control class
- Simulations
  - Continue to create videos for simulations
  - Record trajectory control
    - ➢ Settling time
    - ➢ Overshoot
    - ➢ etc.
- Website
  - Presentable
  - Deliverables included on website
  - Easy to navigate

Chris Noe

- ROS Research
  - Reading through provided papers and textbook
  - Searching Bitcraze and other online forums to better understand ROS
- LOCO Positioning Research
  - Research the basics of the LOCO system
  - Create a high-level diagram
  - Print the anchor brackets for our use
  - Setup and test the LOCO system with one drone
  - Possibly expand to use multiple drones
- ROS Implementation
  - Create a system diagram
  - Work to compile and implement the control code when finished
  - Test the control system thoroughly

Trevor Rice

- Kinect Implementation
  - Management and setup of programs needed to connect the Kinect to ROS
    - Researching older libraries and programs used on more recent versions of the Kinect
  - Setup and calibration of Kinect system(s)
  - Color threshold algorithm research
  - Physical setup of Kinect system and 3D space for control

Schedule

November:
- 28: Final Proposal and Presentation due

December:
- 5: LOCO system setup
- 7: Website with deliverables due

January:
- 31:
  - Single Crazyflie Control Operational
  - Multiple Quadrotor Model & Simulations

February:
- 16:
  - Kinect control operational
  - LOCO system operational

March:
- 29: Final Report draft due

April:
- 10: Student Expo
- 26: Presentation ready

May:
- 1: All deliverables due

Deliverables

- Project Proposal
- Proposal Presentation
- Project Website (with pdfs for Presentation and Proposal)
- Project Midpoint Progress Update
- Student Expo Presentation
- Final Report
- Final Presentation

Project Goals

- We have simulations with smooth flight for a quadrotor
  - Duplicate simulation results with Crazyflies
- Apply formation control to multiple Crazyflies
  - Through modeling and simulations
  - Duplicate with Crazyflies
- System integration with LOCO/Kinect and ROS

Questions?