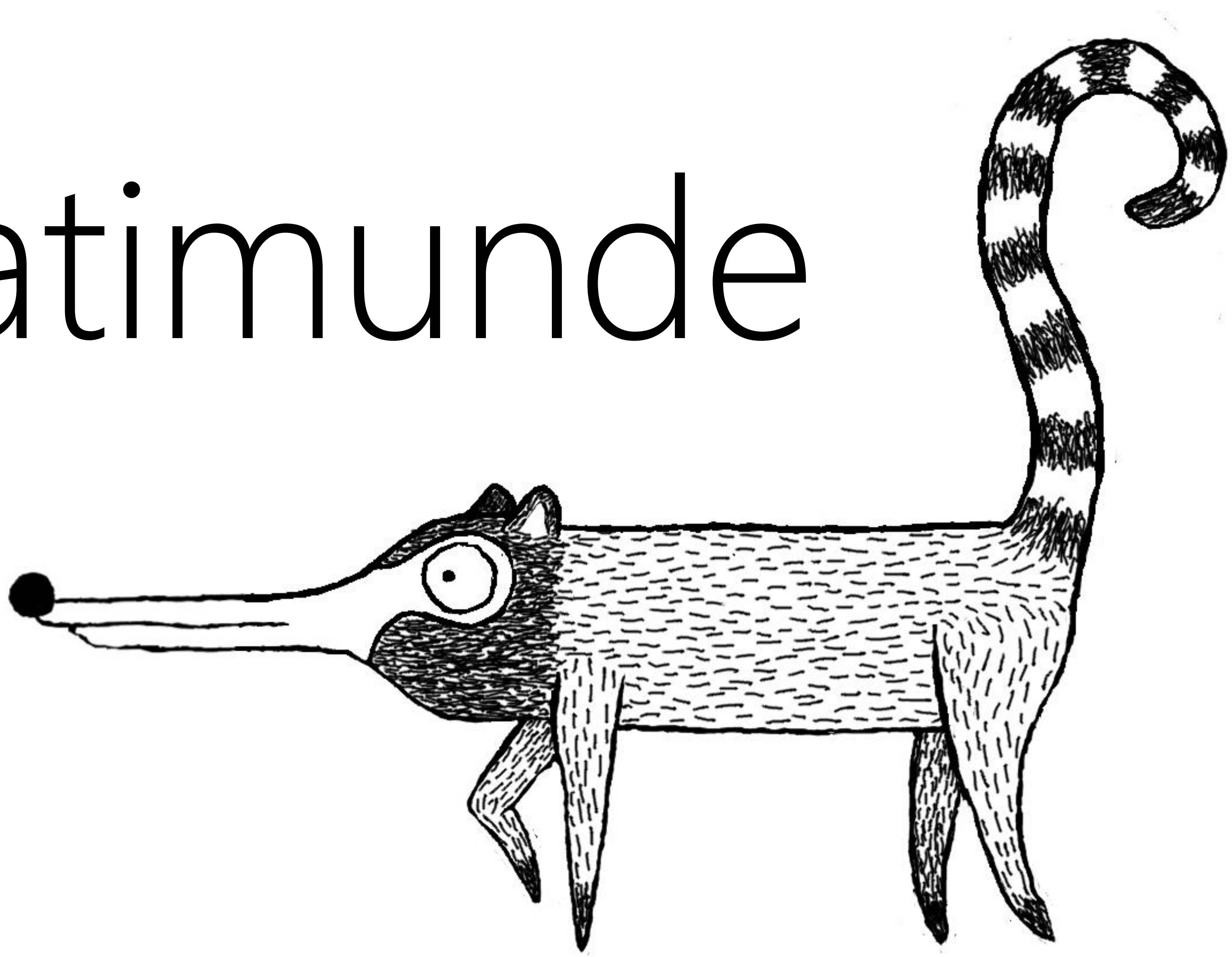
coatimunde



Design Overview

By Kara Stephan and Navarre Hebb

Overview

- Problem Description
- Design Overview
- Scheduling
- Risks & Issues

coatimunde

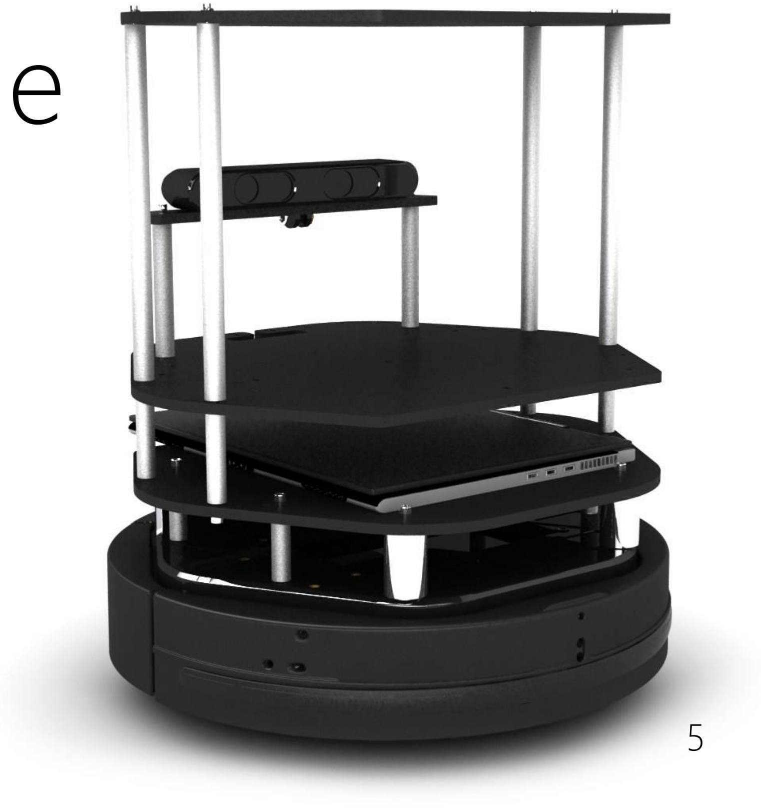
- Computer Optics
- Analyzing Trajectories In
- Mostly Unknown,
- Navigation Denied, Environments

Background

- Obstacle Avoidance
- Unmanned Aircraft Systems
- Computer Vision
- Robot Operating System (ROS)

Requirements

- Movement Toward Target
- Identification of Target
- Identification of Obstacle
- Avoiding Obstacles
- Multiple Obstacles

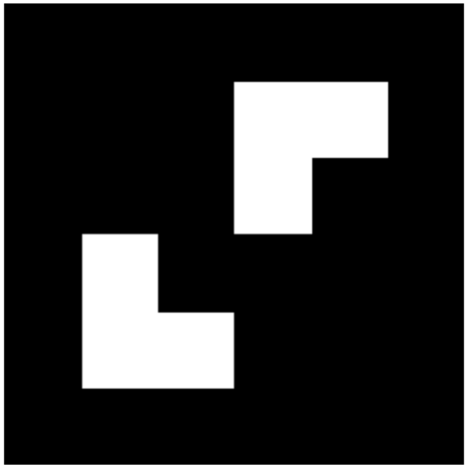


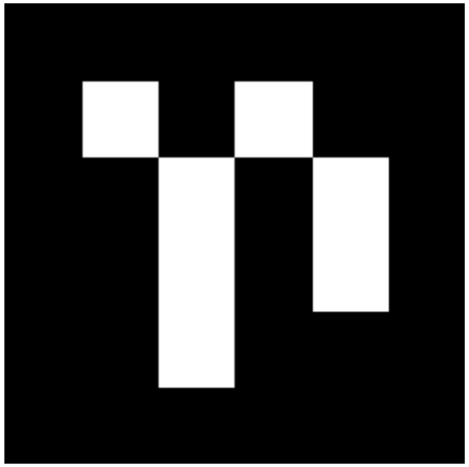
Scope

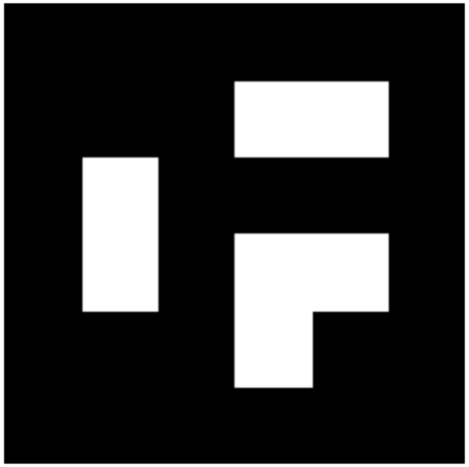
- Small Robotics Laboratory
  - Lower Speeds
  - Smaller Obstacles
- ArUco Marker
- Require only input to commence

Design Overview

- Existing Software
  - Robot Operating System
  - Many ROS Nodes
  - Gazebo
  - OpenCV
  - ArUco

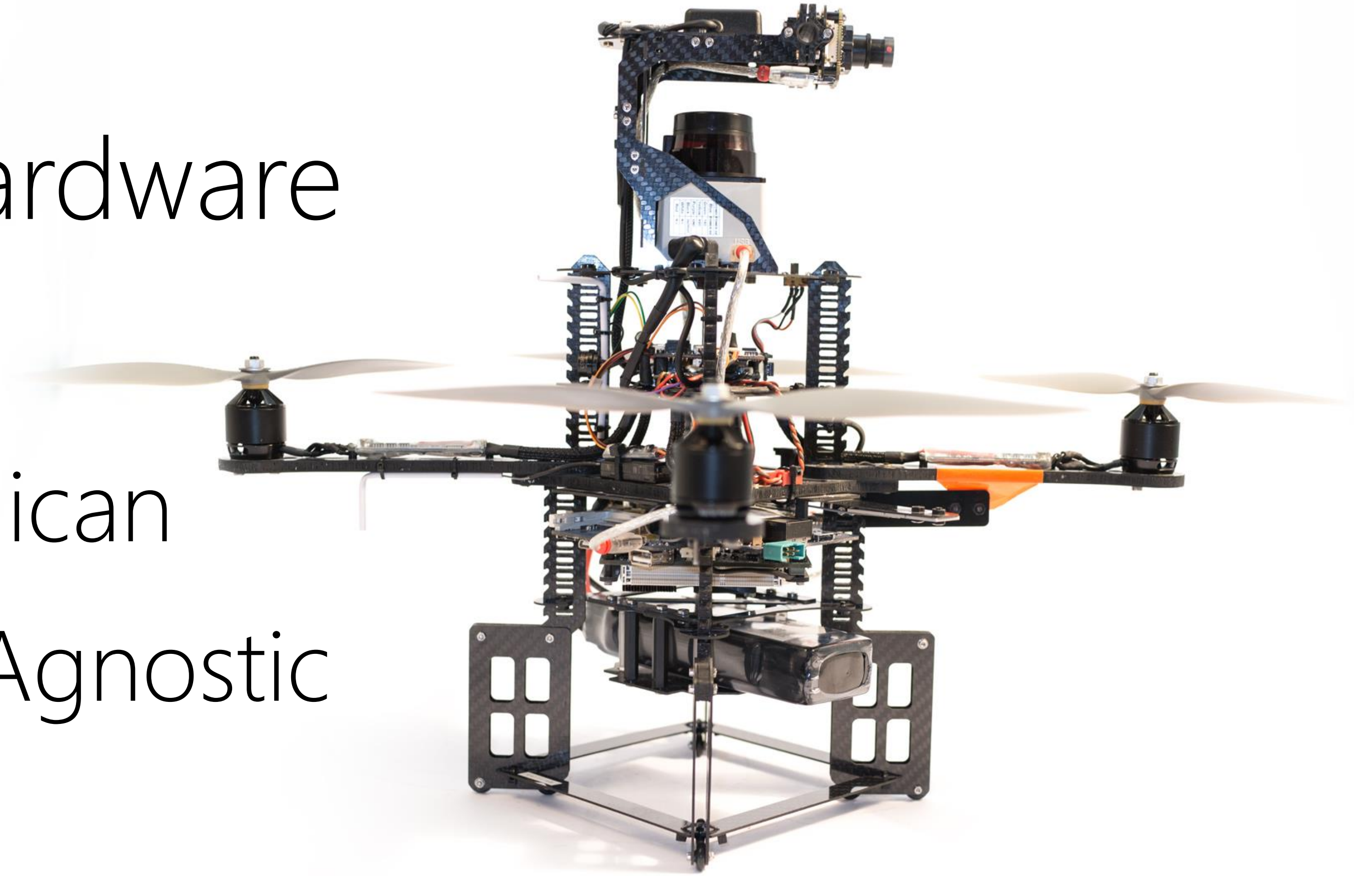

ArUco 42


ArUco 27

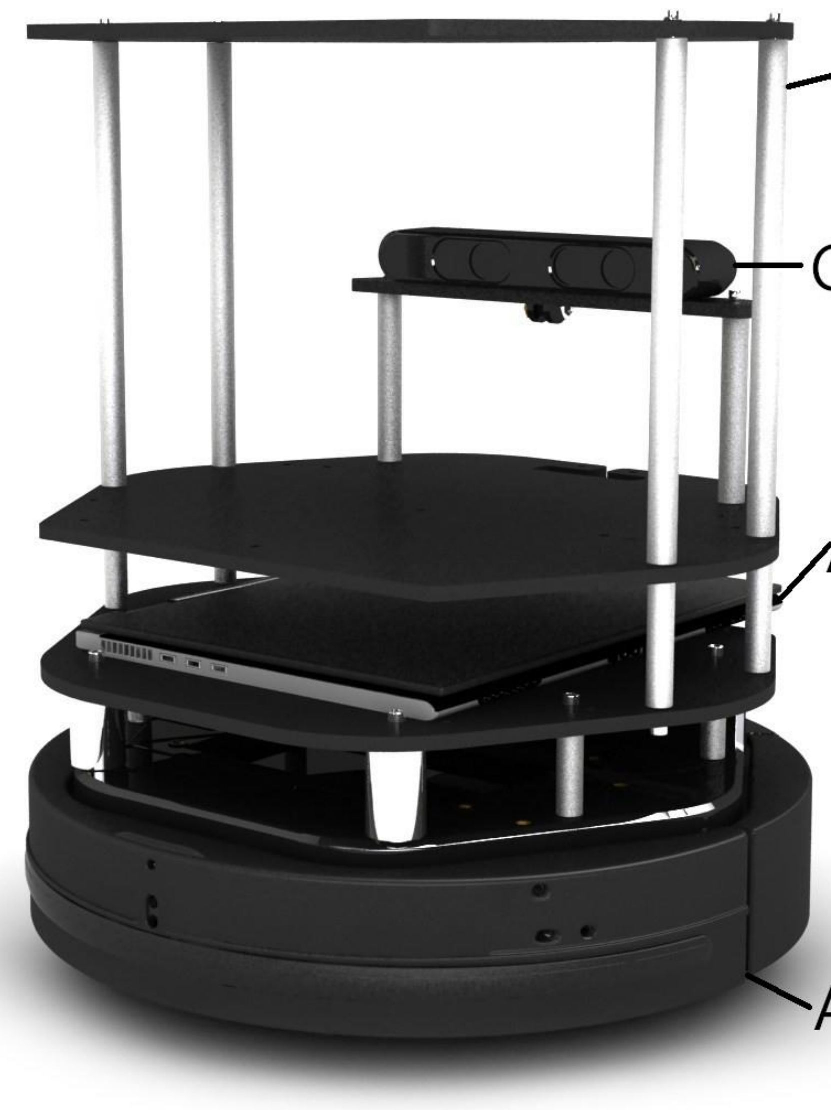

ArUco 43

Design Overview

- Existing Hardware
  - TurtleBot
  - AscTec Pelican
  - Platform Agnostic



Equipment List



AAA - Turtlebot

CAA - Orbbec Camera

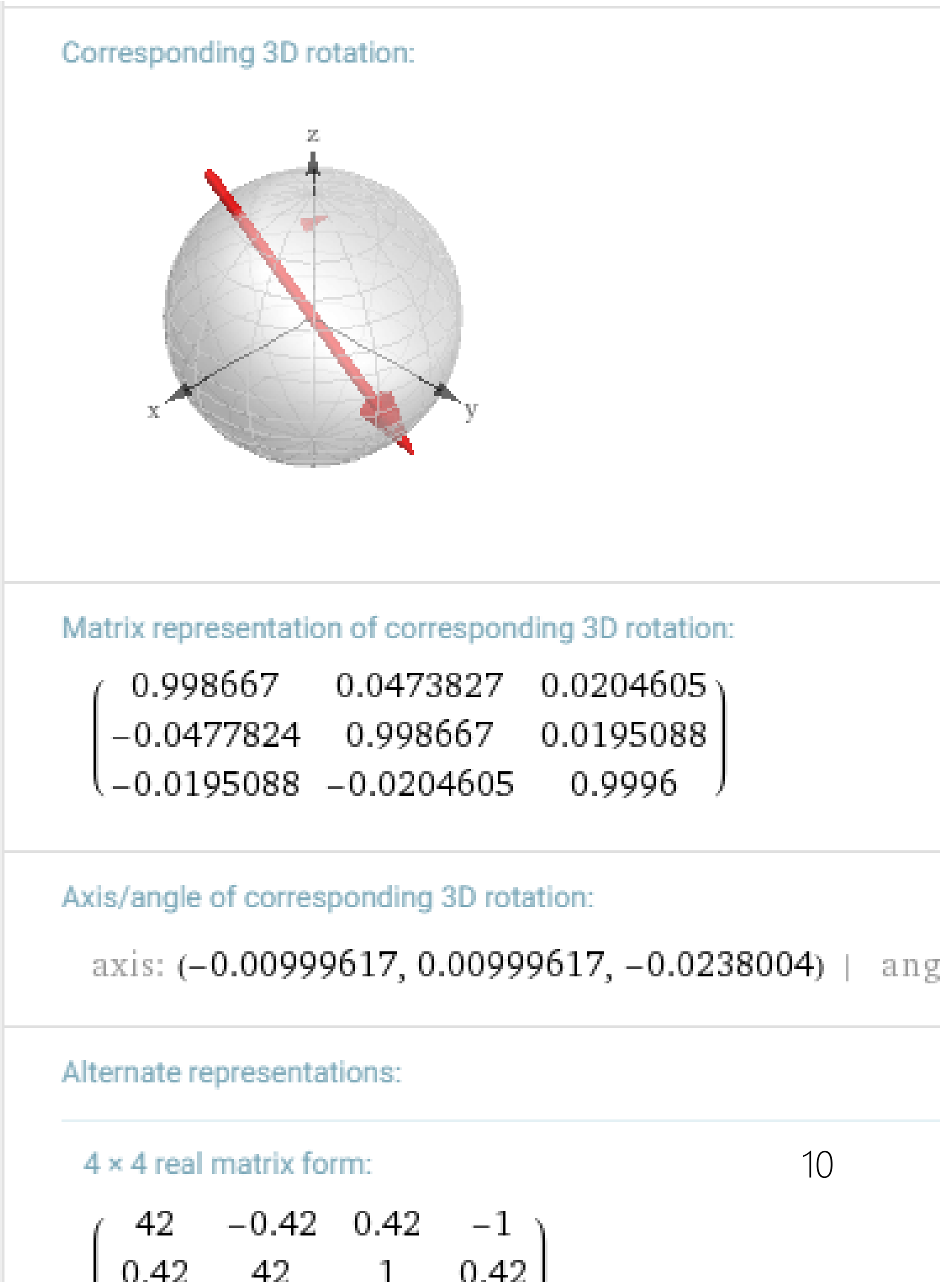
ABA- Acer Laptop

- SAB - Ubuntu 14.04
- ZAA - ROS Indigo
- XA? - Movement
- XB? - Vision

ACA - Kobuki Base

Mathematical Modeling

- Flying Robot
- Robot in 3D Space
- Pose Estimation



Corresponding 3D rotation:

Matrix representation of corresponding 3D rotation:

$$\begin{pmatrix} 0.998667 & 0.0473827 & 0.0204605 \\ -0.0477824 & 0.998667 & 0.0195088 \\ -0.0195088 & -0.0204605 & 0.9996 \end{pmatrix}$$

Axis/angle of corresponding 3D rotation:

axis: (−0.00999617, 0.00999617, −0.0238004) | ang

Alternate representations:

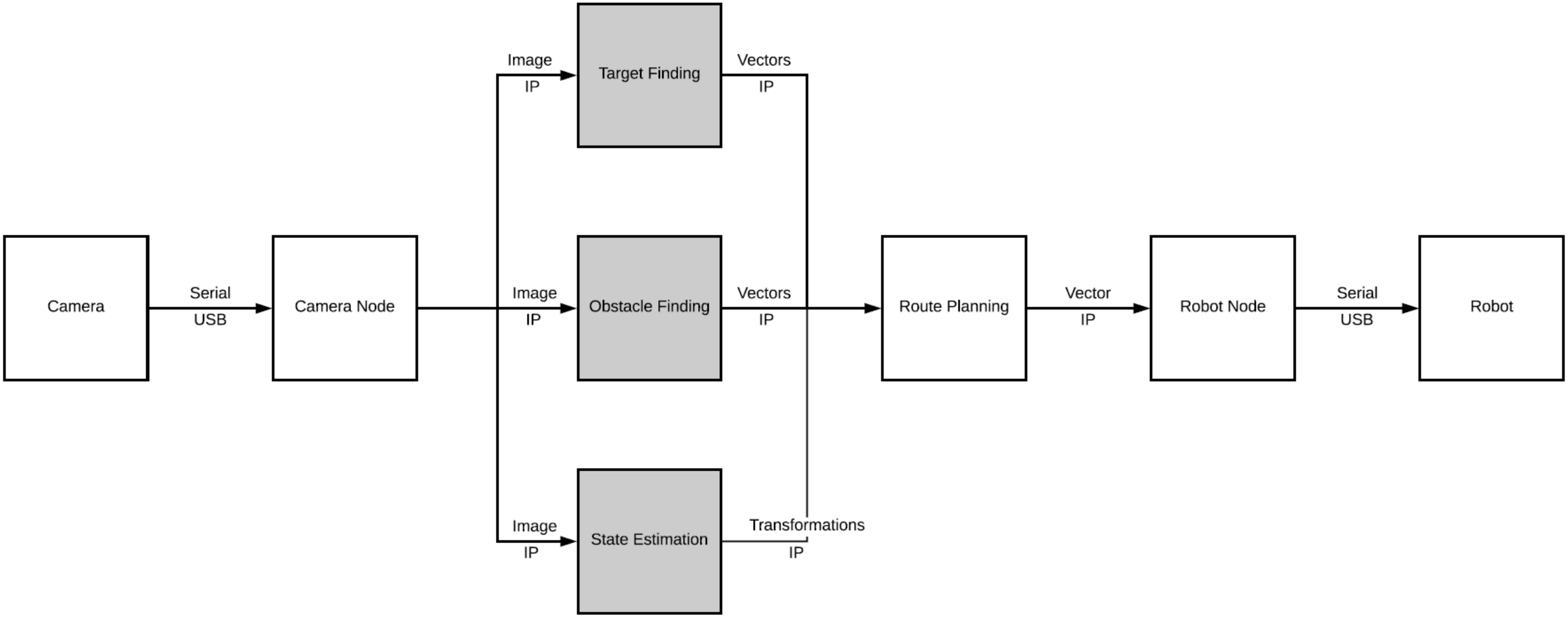
4 × 4 real matrix form:

$$\begin{pmatrix} 42 & -0.42 & 0.42 & -1 \\ 0.42 & 42 & 1 & 0.42 \end{pmatrix}$$

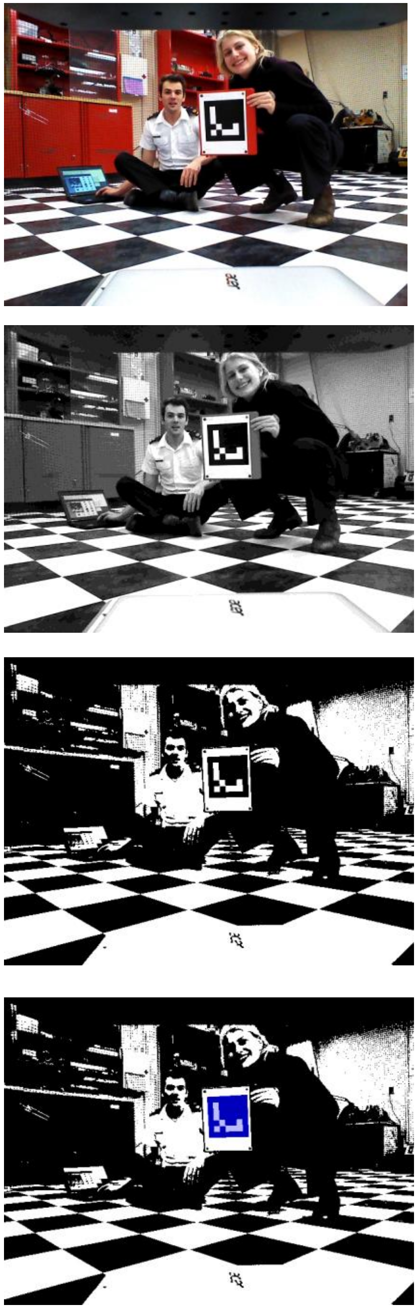
System Architecture

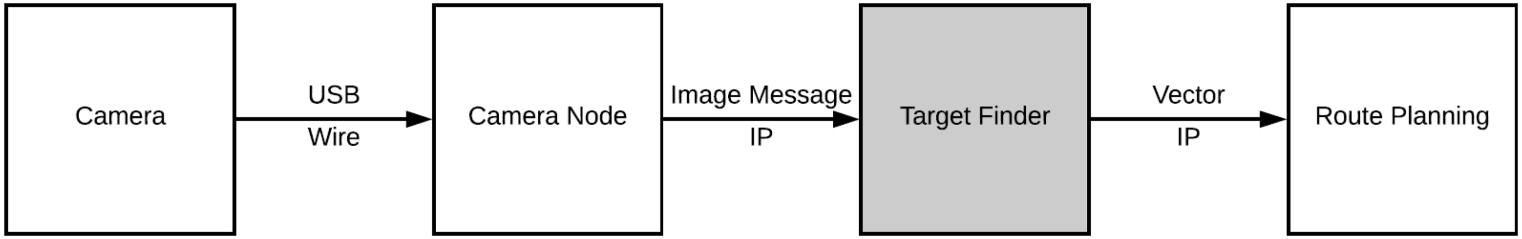
- ROS/Patterns
  - ROS Enhancement Proposals
- ROS/BestPractices
  - Needed Best Practices
  - Package Organization
  - Custom Nodes, Messages, and Services

Simplified Diagram



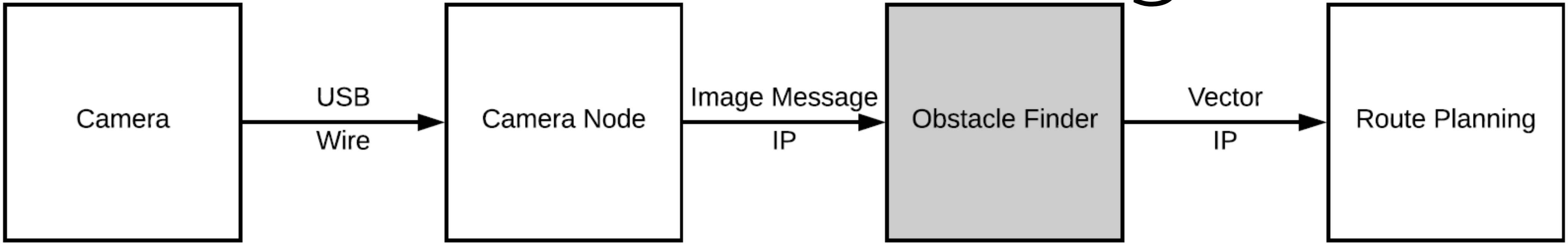
Target Finding



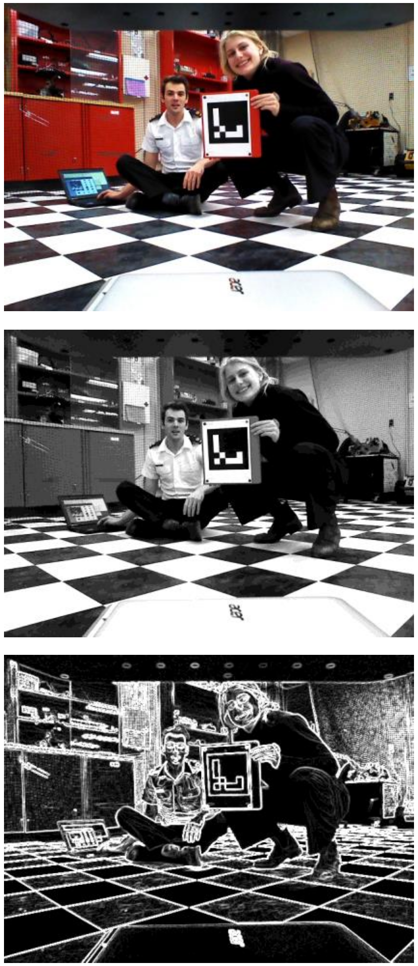


- ArUco Library
- Written in Python and C++
- Input: Camera Information
- Output: Vector to Marker

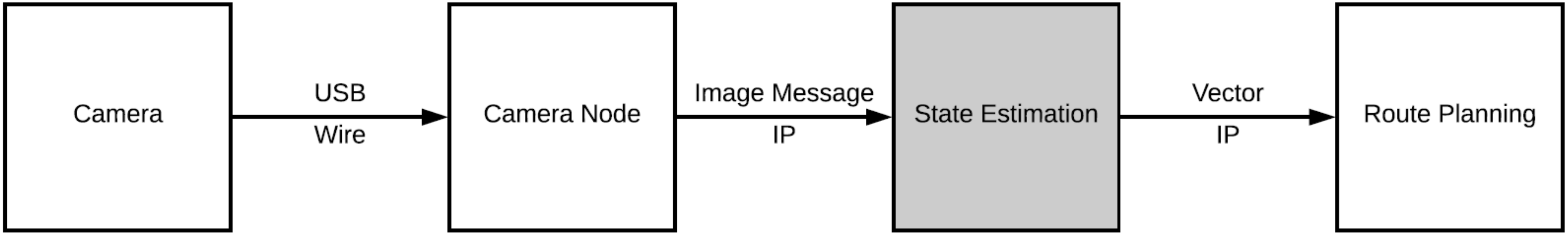
Obstacle Finding

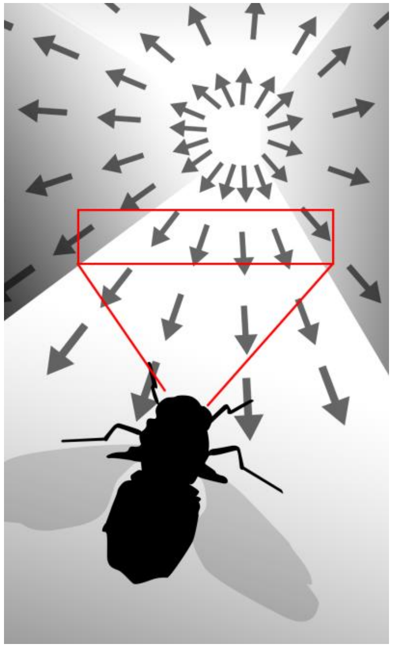


- OpenCV
  - Sobel, Basic Math, Parallax Shift
- Written in Python
- Input: Camera
- Output: Vectors to Edges



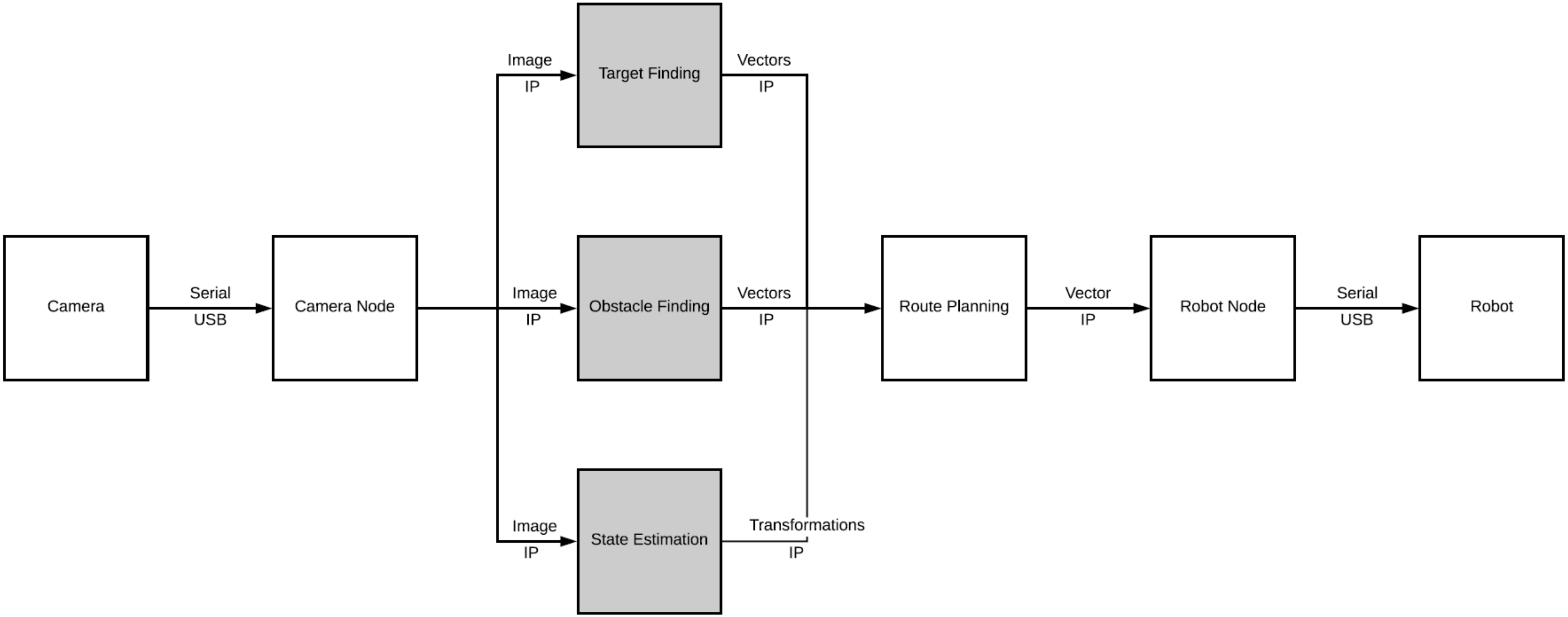
State Estimation

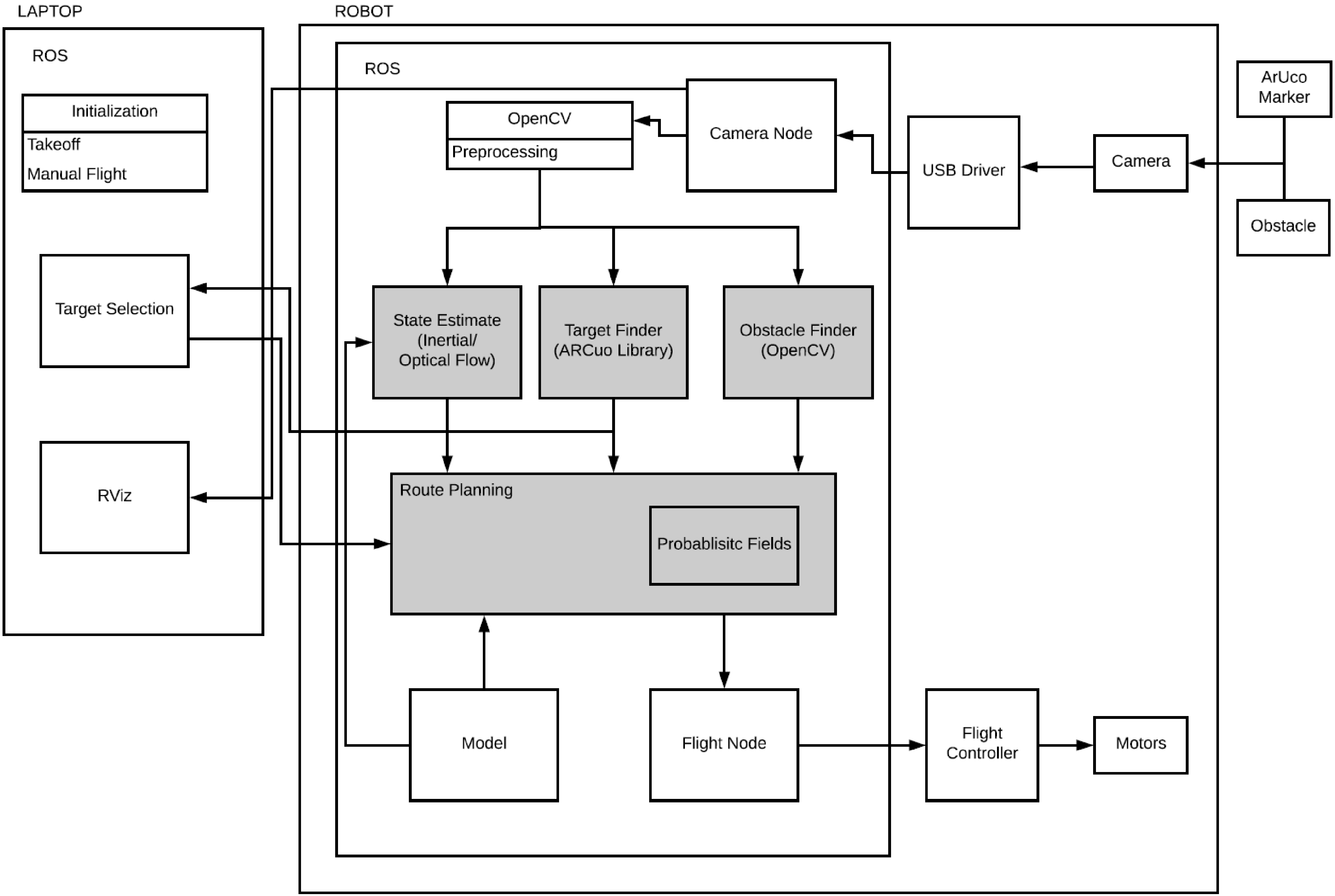




- Written in Python
- Input: Initially Sensor Data
  - Ideally Camera Data
- Output: Transformation Vectors

Simplified Diagram




LAPTOP
ROS
Initialization
Takeoff
Manual Flight
Target Selection
RViz
ROBOT
ROS
OpenCV
Preprocessing
Camera Node
USB Driver
Camera
ArUco Marker
Obstacle
State Estimate (Inertial/ Optical Flow)
Target Finder (ARCuo Library)
Obstacle Finder (OpenCV)
Route Planning
Probablisitc Fields
Model
Flight Node
Flight Controller
Motors

Verification and Validation

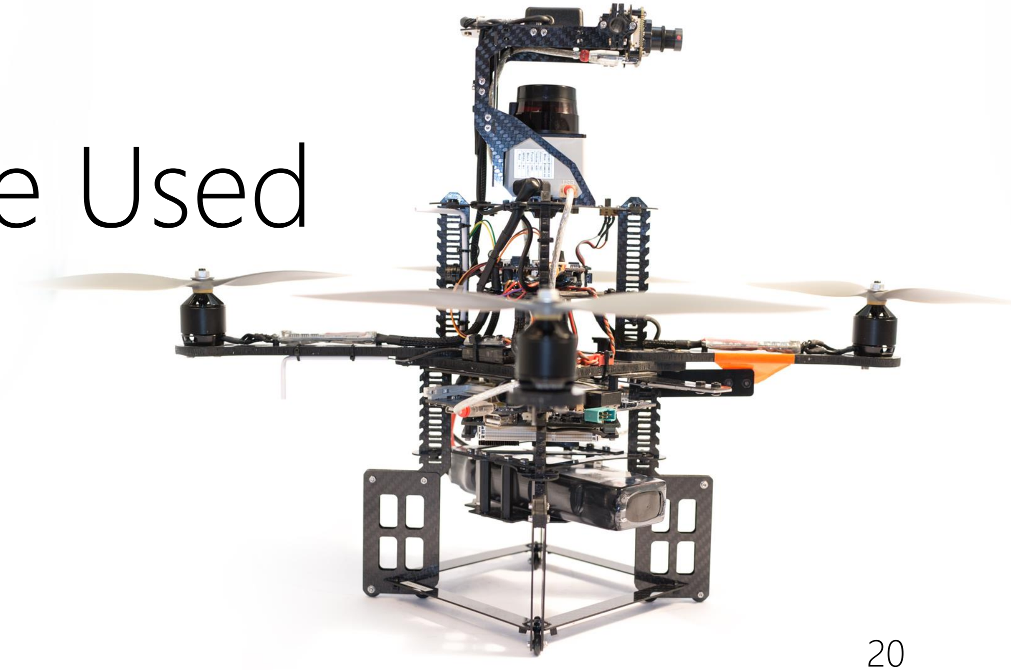
- Individual Nodes
- Nodes Interacting
- Gazebo
- Observing Robot in Lab
- Tracking Robot in Lab

Schedule Milestones

- Turtlebot Finished – Feb 20th, 2019
  - Move towards Target - Nov 30th, 2018
  - Avoid Obstacles – Jan 10th, 2019
  - Obstacle Memory – Feb 20th, 2019
- Port to UAV – April 28th, 2019

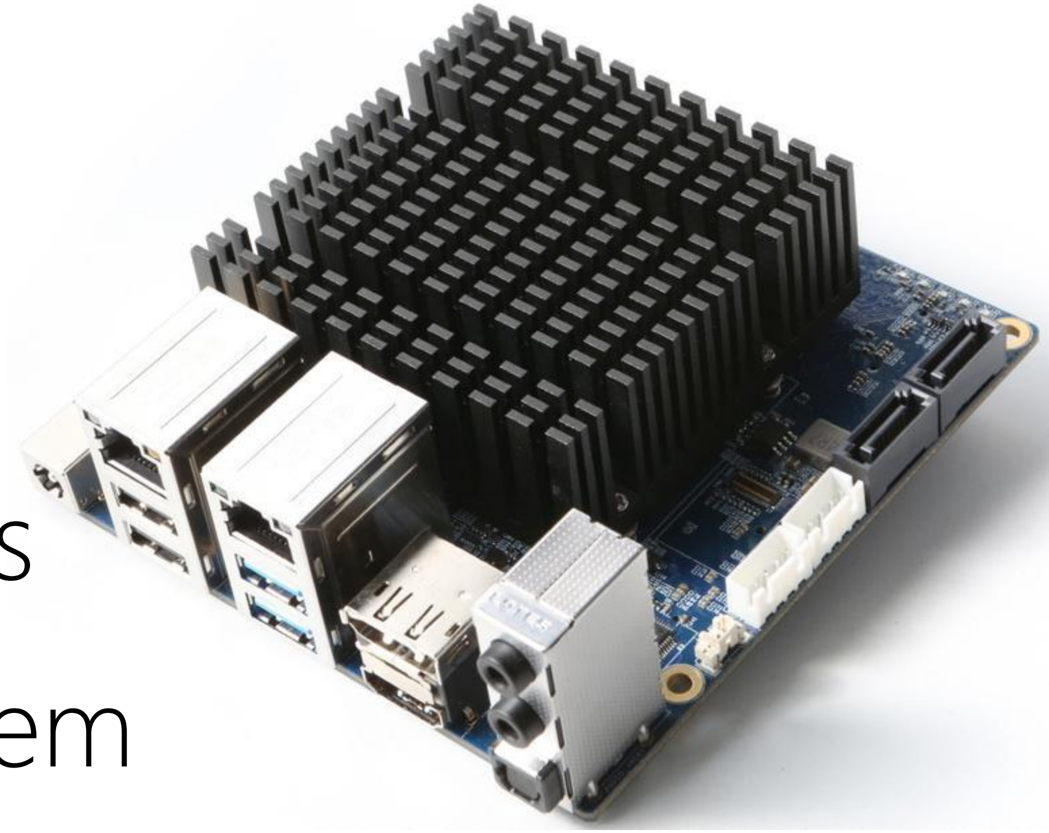
Porting to Flying Robot

- Turtlebot is not Quadcopter
- ROS Nodes are Platform Agnostic
- UAV Flying in Plane
- Same Nodes can be Used



Risks and Issues

- Small Lab
- Identifying Markers
- Flight Control System
- Computer Hardware Limitations



Summary

- Problem Description
- Design Overview
- Scheduling
- Risks & Issues

Thank You!
Questions?