EZ-RASSOR 2.0: Black Team

Group 12

Team Members

John (Jordan) Albury
Shelby Basco
John Hacker
Michael Jimenez
Scott Scalera

Sponsor

Michael Conroy

Mentor

Kurt Leucht

User Story

- Rover is at home base
- Needs to travel to digsite
  - Has no GPS
  - Boulders and craters

Regolith on the moon [1]

Background: EZ-RASSOR 1.0

- Previous team created the base software
  - EZ-RASSOR meant for the Mini-RASSOR
- Needs better autonomy
  - Dependent on navigating without a GPS
- Two EZ-RASSOR 2.0 teams
  - Black: GPS-Denied Navigation
  - Gold: Swarm Control

The RASSOR [2]

Technologies

Programming Language: Python2

Robotics Framework: Robot Operating System (ROS)

Testing Environment: Gazebo, RVIZ

Version Control: Git




Specifications

Objectives

- Add autonomous navigation capabilities to the EZ-RASSOR
- Test these capabilities in a simulated, moon-like environment
- Provide thorough documentation for future teams

Requirements

- Determine the location of the robot in a GPS-denied environment
  - Given a known starting location
- Route the robot to a target location
- Detect and avoid obstacles

Components

Absolute Localization: Estimates the coordinates of the robot

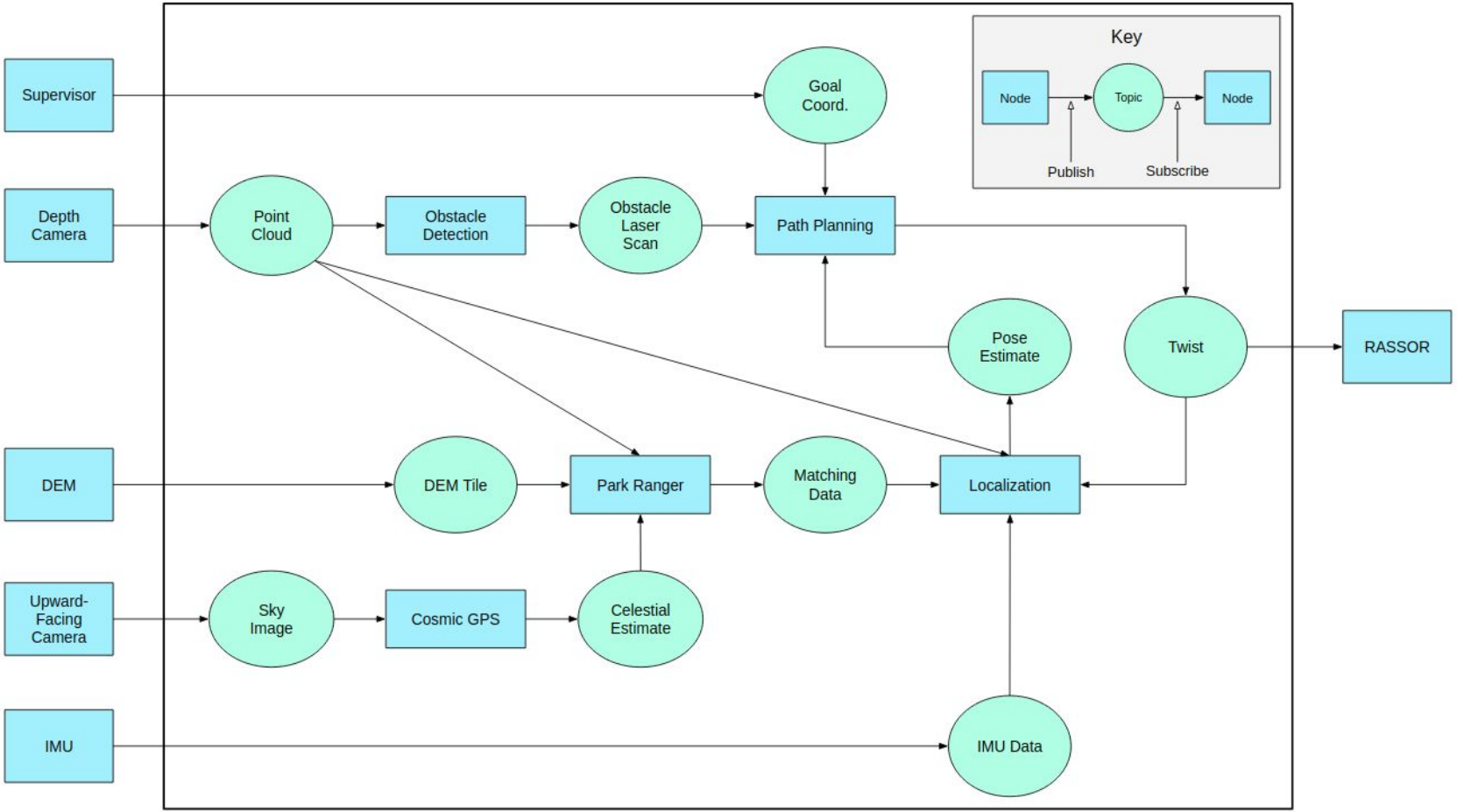
- ★ **Cosmic GPS:** Derives location based on positions of stars
- ★ **Park Ranger:** Derives location by comparing surroundings to a digital elevation model

Odometry: Estimates the motion of the robot

Obstacle Detection: Detects obstacles in front of the robot

Path Planning: Routes the robot to a target location

EZ-RASSOR 2.0 Black Team Software Architecture



Roles

Shelby: Project Manager, Absolute Localization, Odometry

Jordan: Odometry Lead, Obstacle Detection, Path Planning

John: Obstacle Detection Lead, Path Planning

Mike: Path Planning Lead, Obstacle Detection

Scott: Absolute Localization Lead, Path Planning

Necessary Hardware

- Intel Realsense D435i depth camera
  - Horizontal FOV: 74 degrees [3]
  - Range: 0.105m to 10m [4]
  - BMI055 IMU [3]
- Camera for taking pictures of sky

90 mm x 25 mm x 25 mm

The Intel Realsense D435i depth camera [4]

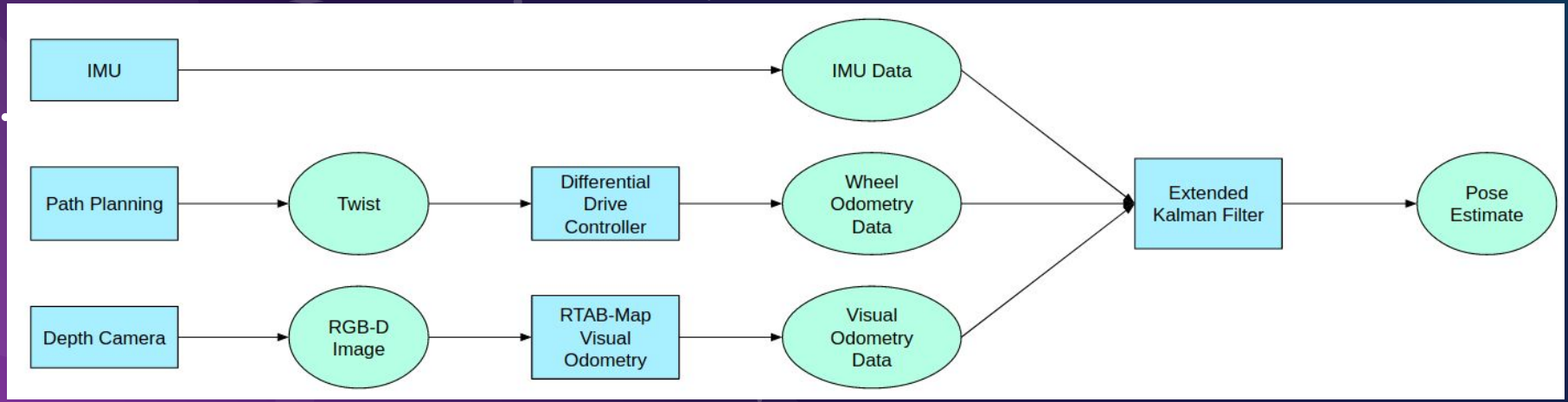
Budget

- Camera for testing Cosmic GPS: **$55**
- All software used is open source (free)


Relative Localization
Odometry

Odometry

- ★ Provides frequent pose (position and orientation) updates for use in Path Planning
- ★ Combines IMU Odometry, Wheel Odometry, and Visual Odometry via an extended Kalman filter



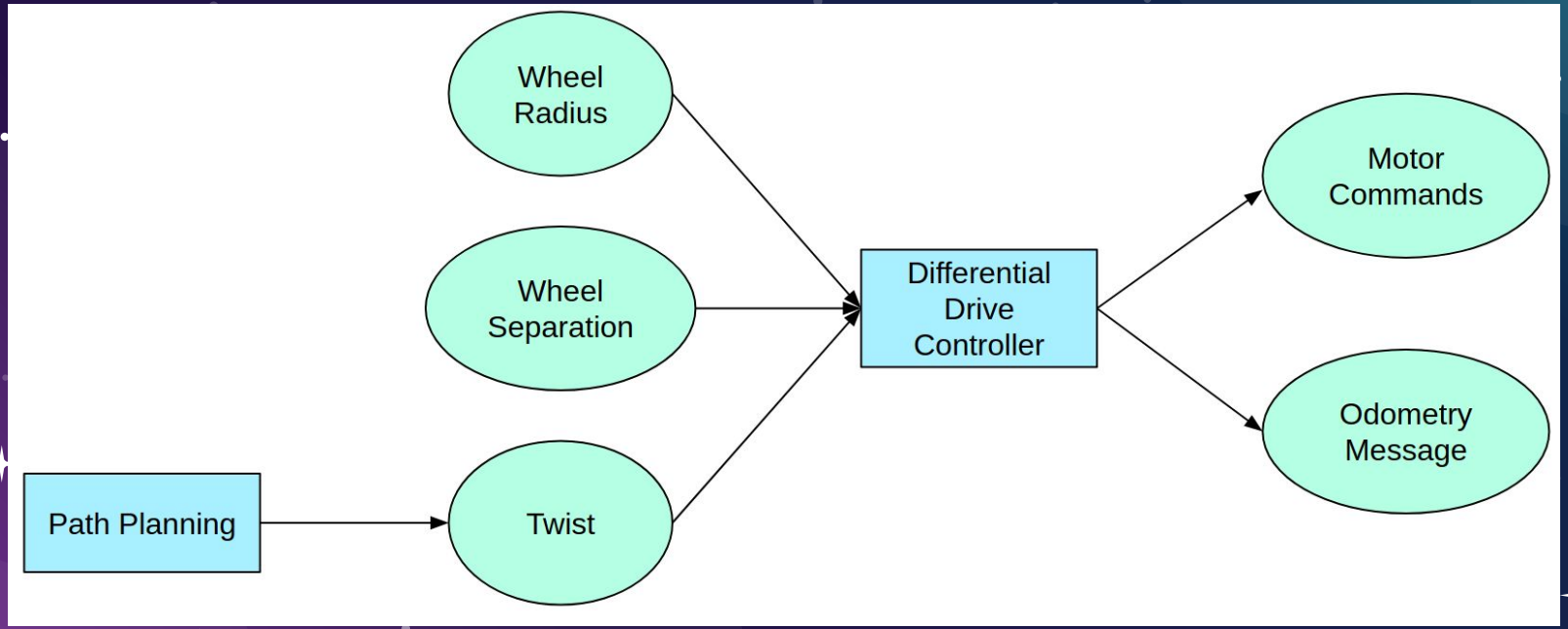
IMU Odometry

- Added simulated IMU sensor with Gaussian noise

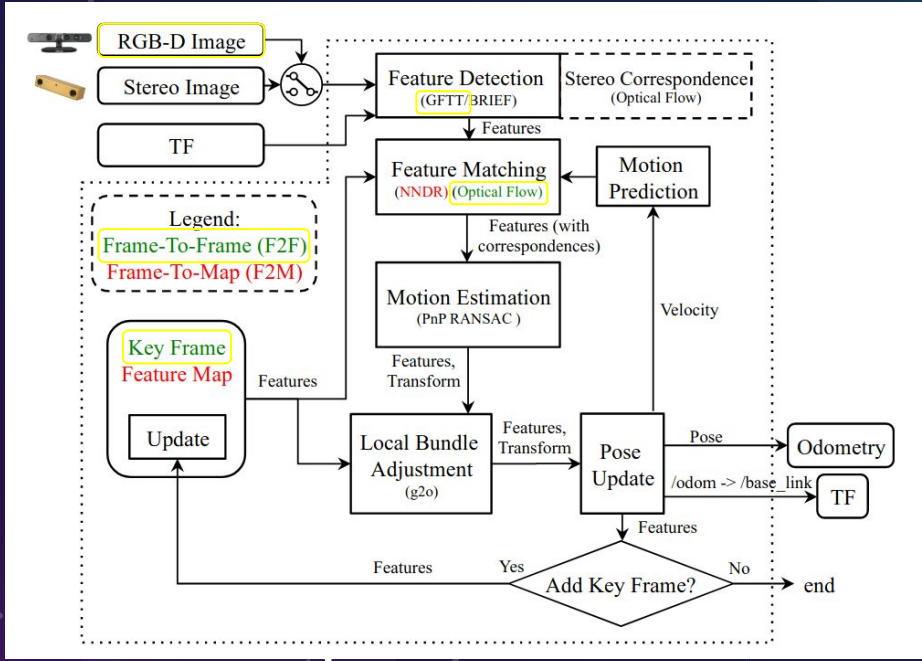
```
# This is a message to hold data from an IMU (Inertial Measurement Unit)
#
# Accelerations should be in m/s^2 (not in g's), and rotational velocity should be in rad/sec
#
# If the covariance of the measurement is known, it should be filled in (if all you know is the
# variance of each measurement, e.g. from the datasheet, just put those along the diagonal)
# A covariance matrix of all zeros will be interpreted as "covariance unknown", and to use the
# data a covariance will have to be assumed or gotten from some other source
#
# If you have no estimate for one of the data elements (e.g. your IMU doesn't produce an orientation
# estimate), please set element 0 of the associated covariance matrix to -1
# If you are interpreting this message, please check for a value of -1 in the first element of each
# covariance matrix, and disregard the associated estimate.

Header header

geometry_msgs/Quaternion orientation
float64[9] orientation_covariance # Row major about x, y, z axes

geometry_msgs/Vector3 angular_velocity
float64[9] angular_velocity_covariance # Row major about x, y, z axes

geometry_msgs/Vector3 linear_acceleration
float64[9] linear_acceleration_covariance # Row major x, y z
```

ROS IMU message definition [5]

Wheel Odometry



Visual Odometry



RTAB-Map RGB-D Visual Odometry Pipeline [6]

Extended Kalman Filter

- Uses estimates and variances from input sources to produce an estimate that tends to be more accurate than any of the input sources

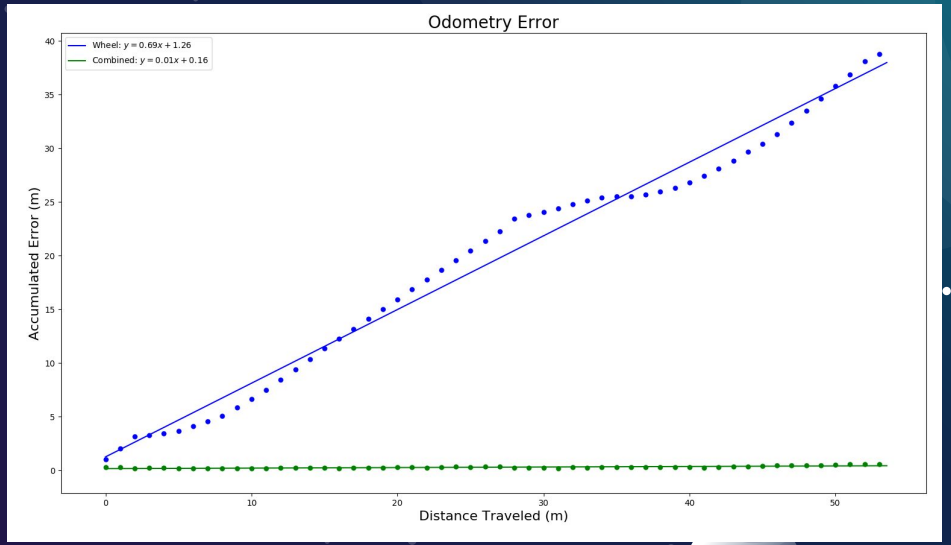
	x	*y*	*z*	*roll*	*pitch*	*yaw*	*x'*	*y'*	*z'*	*roll'*	*pitch'*	*yaw'*	*x''*	*y''*	*z''*
IMU						■							■		
Wheel							■					■			
Visual							■	■				■			

Challenges

- Configuring differential drive controller
  - Replaced manual control system with differential drive controller
- Simulated IMU
  - Switched from *GazeboRosImuSensor* to *GazeboRosImu*

Results

- Combined odometry performs much better than individual components
- Combined odometry error accumulates at a rate of approximately 1 centimeter per meter traveled

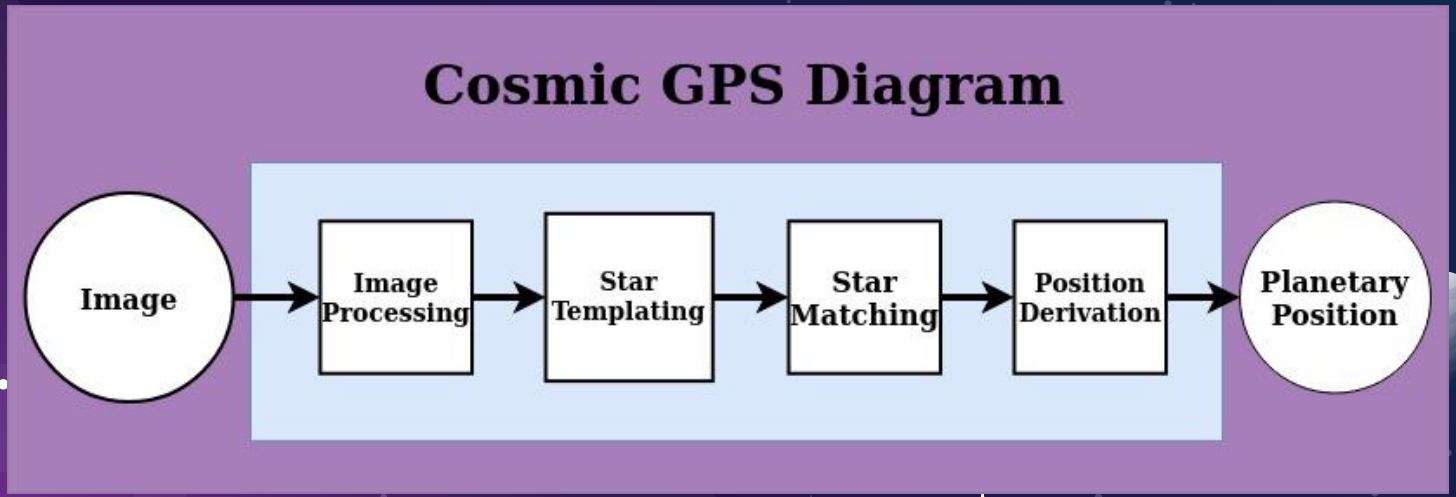


Absolute Localization

Cosmic GPS

Cosmic GPS

- Produces a coarse location estimate
- Location estimate and error is then fed into Park Ranger to refine the estimate



Cosmic GPS Overview

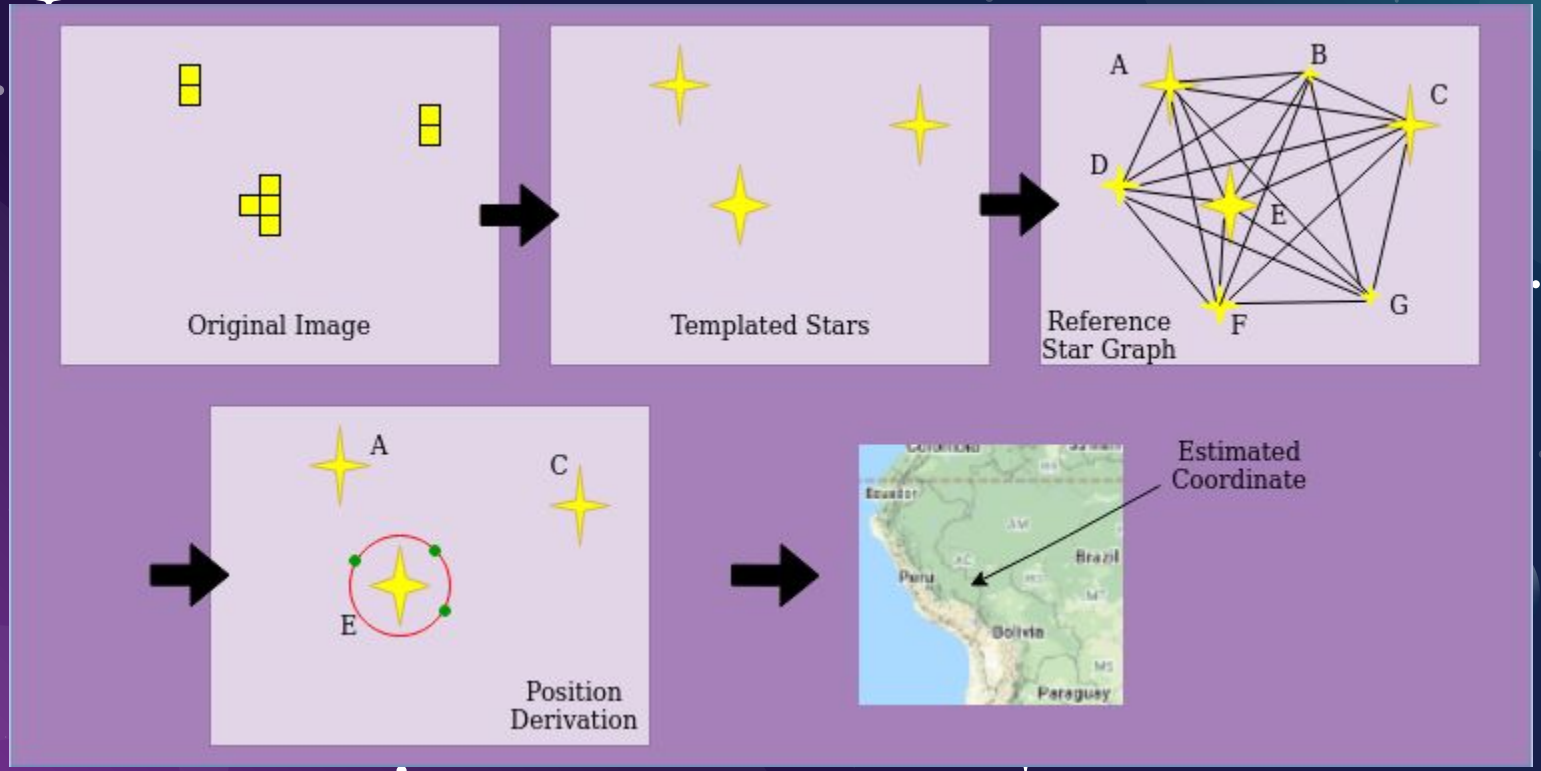
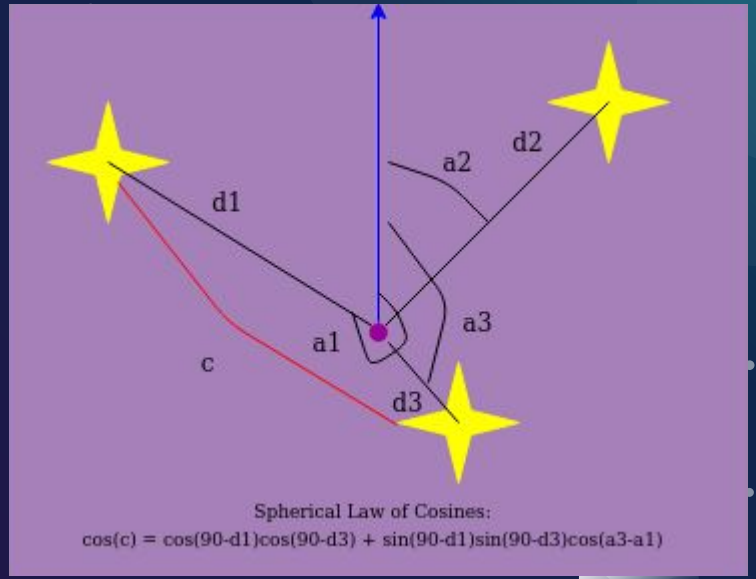


Image Processing

- Master bias frame
  - Base level readout noise
- Master dark frame
  - Hot pixels and dark current
- Master flat frame
  - Variation in pixel sensitivity
- Anisotropic filter
  - Smooths while preserving edges

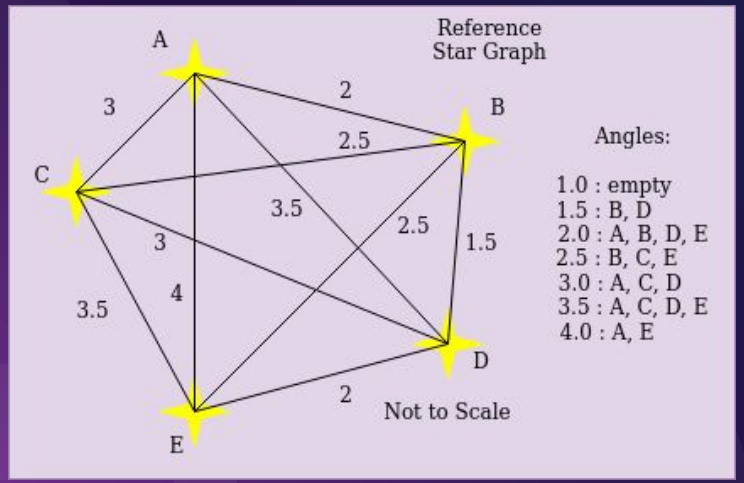
Star Templating

- Thresholding (Local Adaptive)
  - Select “important” pixels
- Clustering
  - Group neighboring pixels
- Centeroiding
  - Determine star centers
- Angles (for each star)



Star Matching

- ★ Take the image's five brightest stars
- ★ Measure distance between each star
- ★ Match these stars into a reference star graph

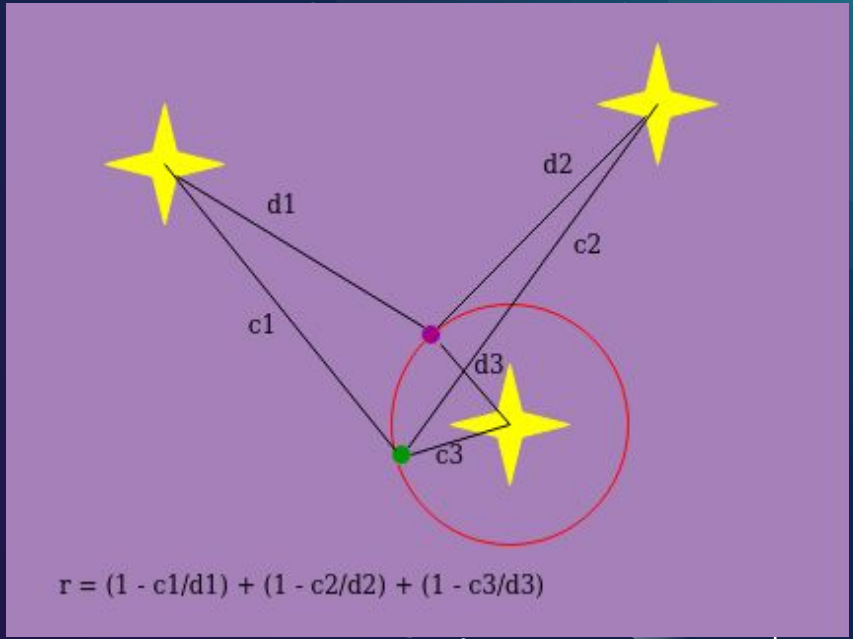


Count Arrays

	A	B	C	D	E
1st :		II	I	I	I
2nd:	I	I	II	I	I
3rd:	I	I	I	II	

Position Derivation

- Based on the celestial positions of the identified stars, search for the celestial and then planetary position of the image's center

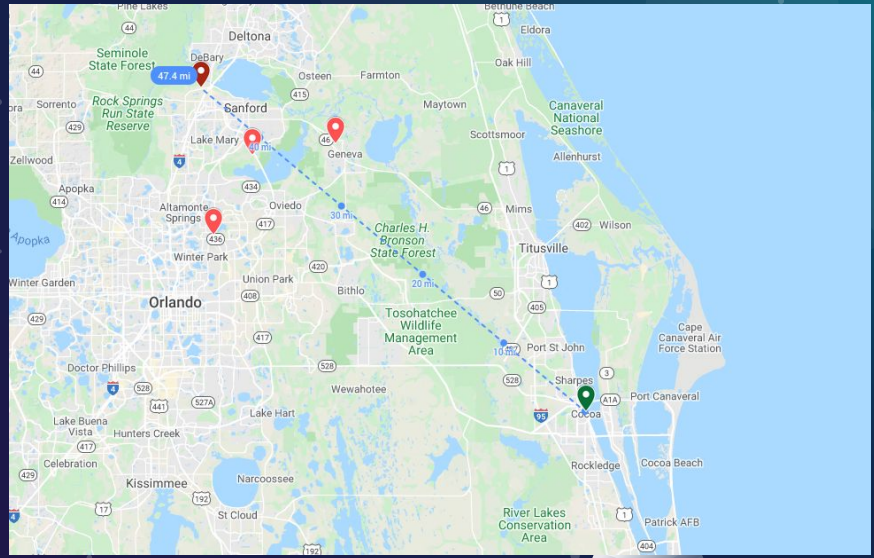


Challenges

- Writing software for hardware that does not exist
- Having to test the software on real data

Results

- Position estimate within 60 miles (less than 1 degree lat/long) of true position
- Would work better on moon
  - Higher curvature



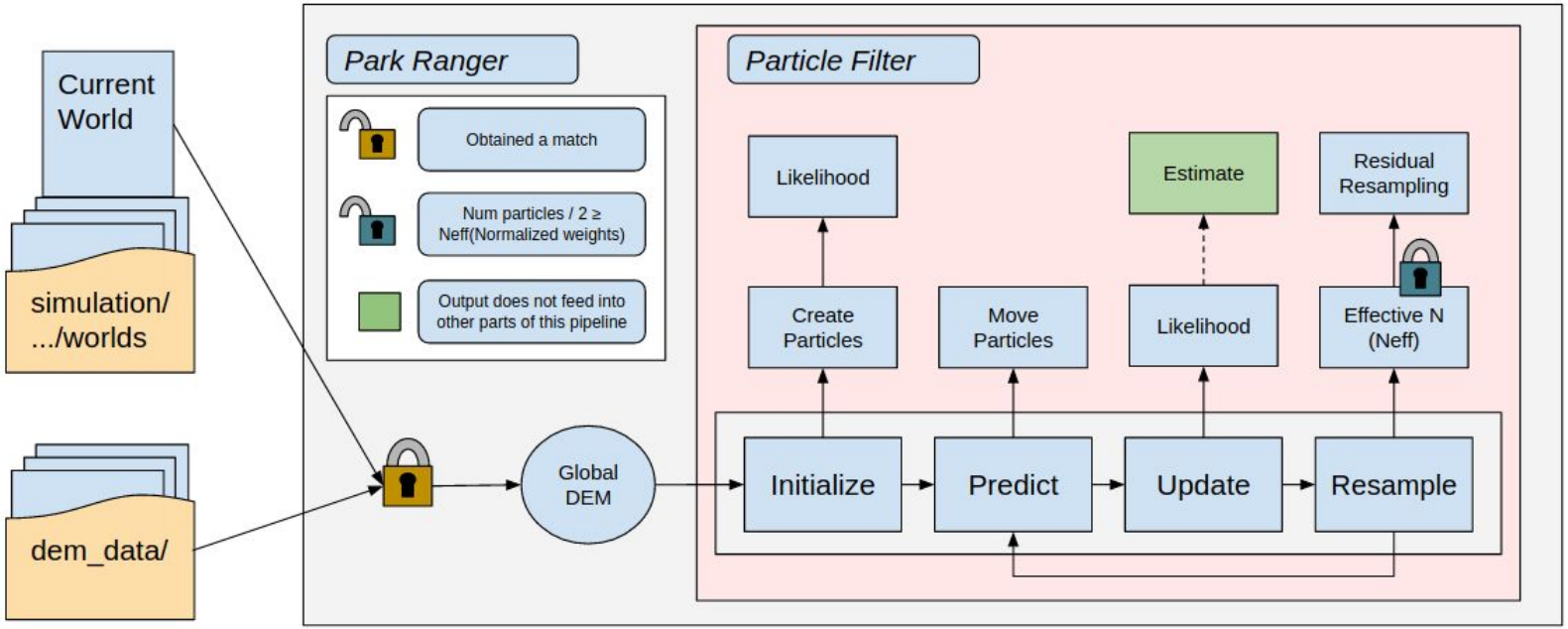
Absolute Localization

Park Ranger

Park Ranger

- Compare direct surroundings with an overhead map
  - Overhead map: Digital Elevation Model (DEM) of the area
- Use particle filter
  - Particle: represents potential state of robot

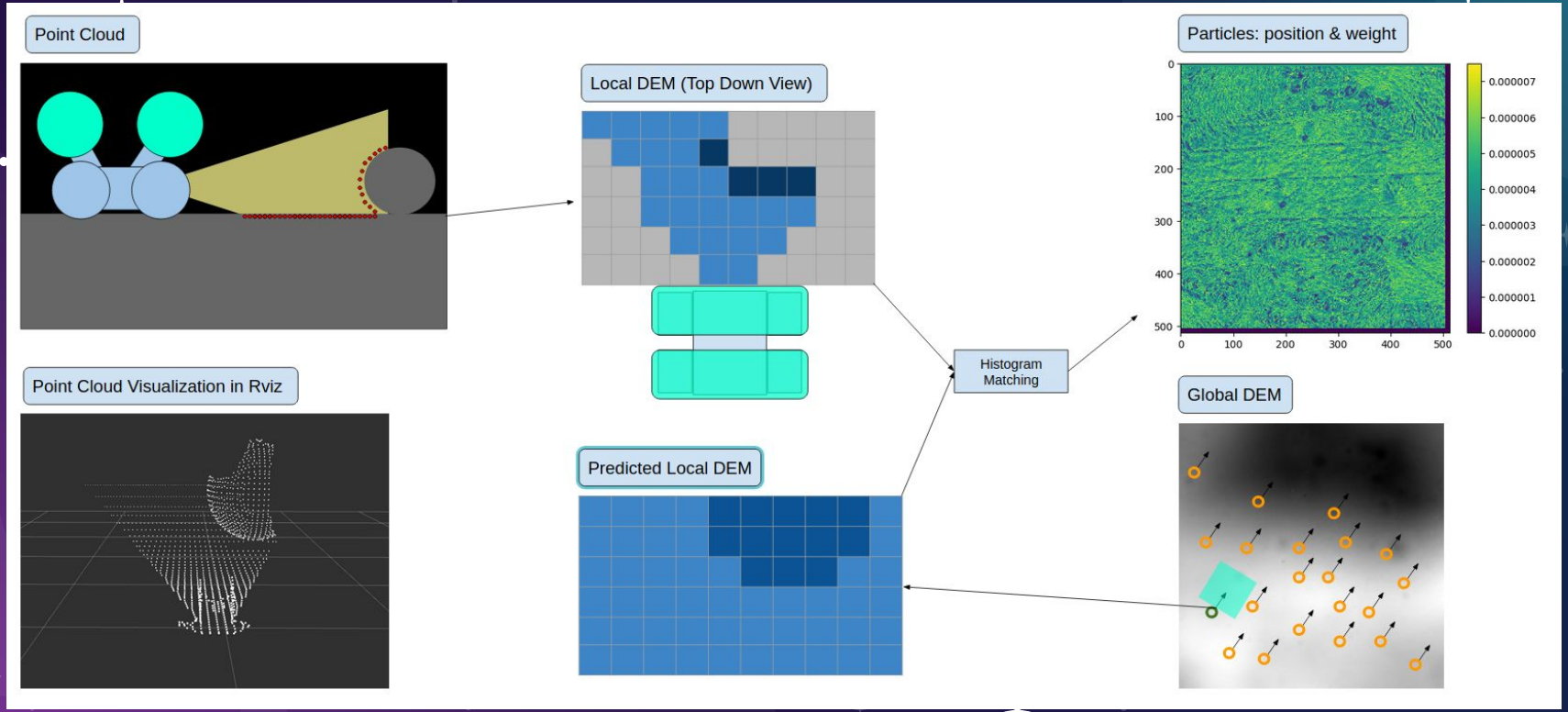
Park Ranger Pipeline



Likelihood

- **Local DEM**: Top down “bucketed” view of point cloud
- **Predicted Local DEM**: Expected local DEM at some position
- Histogram Matching

Likelihood

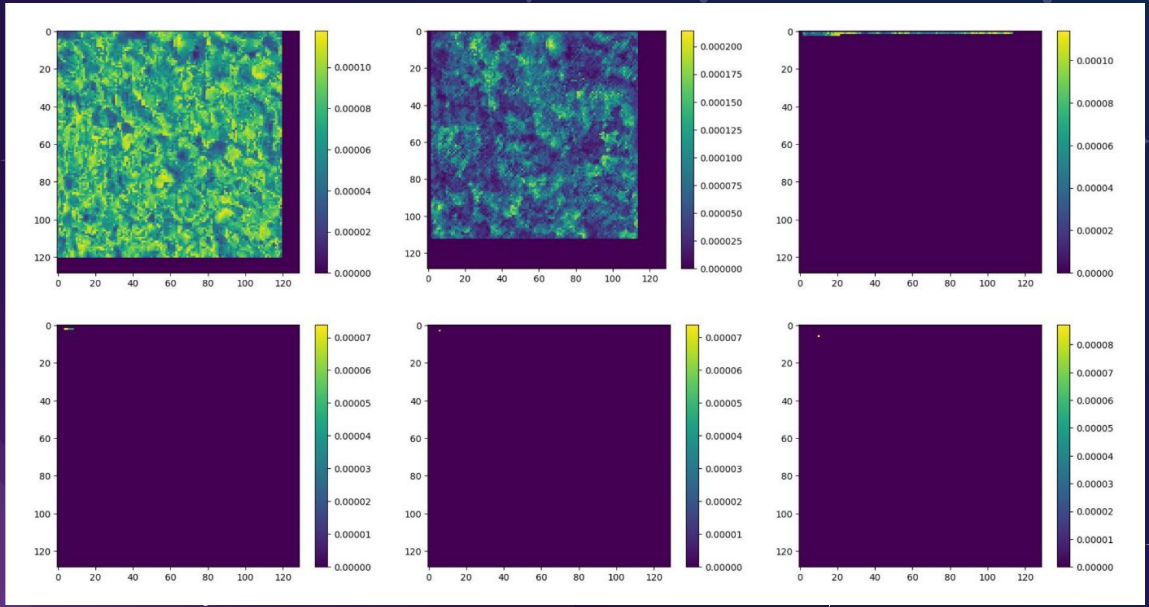

Point Cloud
Local DEM (Top Down View)
Particles: position & weight
Point Cloud Visualization in Rviz
Histogram Matching
Predicted Local DEM
Global DEM

Challenges

- Particle Filter
  - Camera can't see over many hills
    - Flatter terrain gives longer range of view
    - Varying terrain gives more unique predicted local DEMs
  - Real-time plotting for debugging

Results

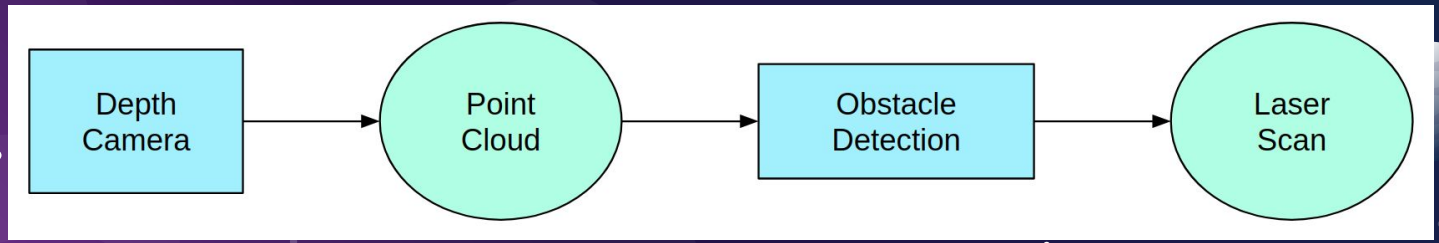
- Converging estimate
- Inaccurate/inconsistent estimate



Obstacle Detection

Obstacle Detection

- Local environment data for path planning
- Laser scan of detected obstacles
  - Array of distances at consecutive angles
- Rocks, craters, and more
- Uneven ground on moon



Point Cloud Processing

- Depth camera produces point cloud
- 3D Cartesian coordinate system
- Right, Down, Forward values for each point
- $\text{Angle} = \arctan\left(\frac{\text{Right}}{\text{Forward}}\right)$
- $\text{Step} = \frac{\text{Angle}_i - \text{Angle}_{min}}{\text{Angle}_{increment}}$
- $\text{Distance} = \text{sqrt}(\text{Forward}^2 + \text{Right}^2)$

Detection Algorithm

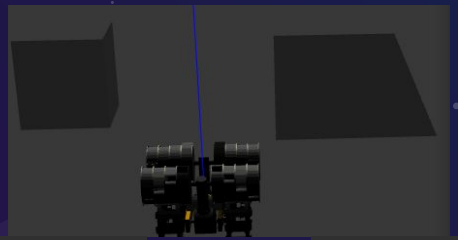
- Group by Step, sort by Distance
- Iterate groups in ascending order
- Hike = ΔDistance
- Slope = $\frac{\Delta\text{Down}}{\text{Hike}}$
- Obstacle if Hike $\geq \text{Threshold}_H$ or Slope $\geq \text{Threshold}_S$
- Min. distance to detected obstacle in each step

Challenges

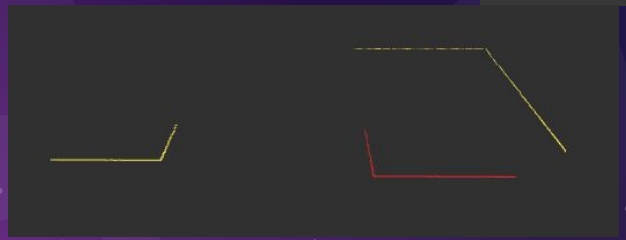
- Detecting cliffs
- Detecting holes
  - Gradients and Hike
- Optimization
  - NumPy vectorization

Results

- ★ Quickly makes admissible laser scan
- ★ Strong with above-ground obstacles
- ★ Moderate hole detection



Hike and Slope



Combined

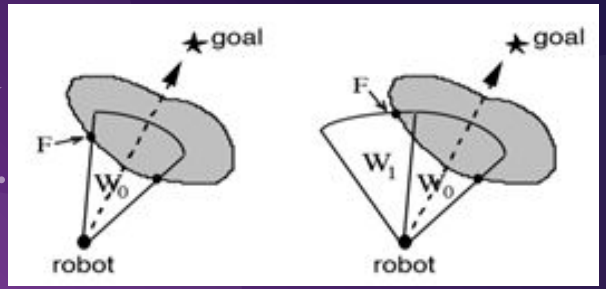
Path Planning

Path Planning

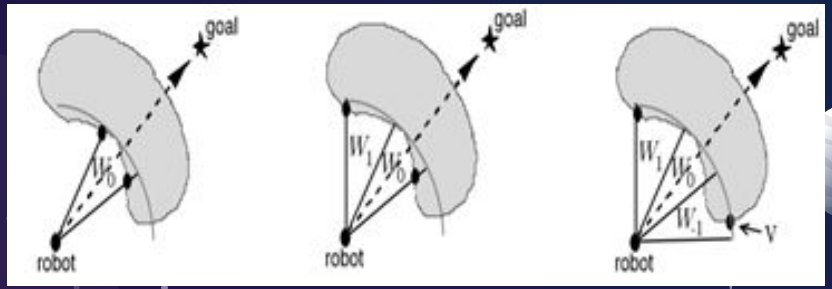
- Get from point A to point B while avoiding obstacles like rocks and craters
  - Intermediate path planning between waypoints
- Use the Wedgebug algorithm
  - Use only sensor data (no map) to avoid obstacles
- Twist message to controllers
  - Linear and angular velocity

Wedgebug Algorithm

- Inspired by how bugs path plan
- If there are no obstacles, continue towards the goal
- If there are obstacles detected (laser scan), then pick the point that will avoid the obstacle and minimize the distance to the goal



Example of virtual motion to goal [7]



Example of virtual boundary following [7]

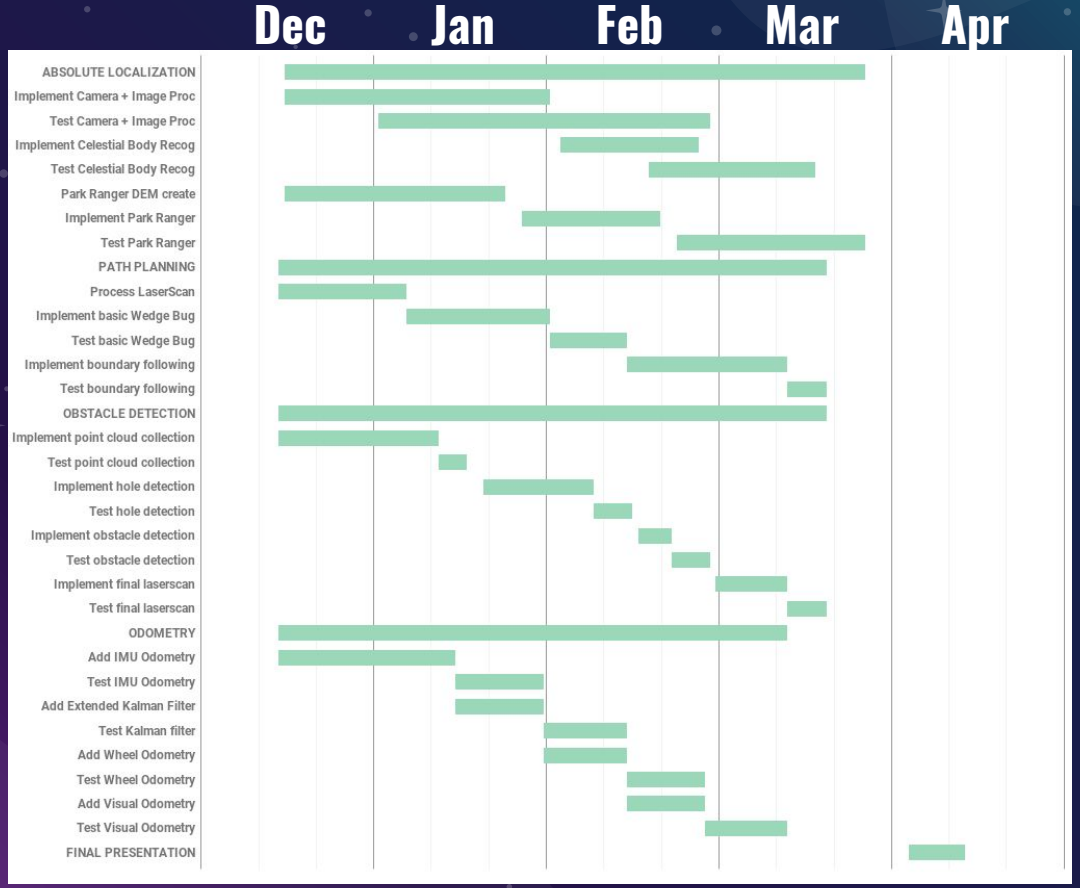
Challenges

- Making sure that the robot doesn't hit obstacles that are out of view of its sensors
  - Use larger buffer when checking safety

Results

- Successful autonomous navigation in simulation
  - Moon (hills and craters)
  - Obstacle course (blocks)
- Fluid movement with ramping function

Milestones


Dec
Jan
Feb
Mar
Apr
ABSOLUTE LOCALIZATION
Implement Camera + Image Proc
Test Camera + Image Proc
Implement Celestial Body Recog
Test Celestial Body Recog
Park Ranger DEM create
Implement Park Ranger
Test Park Ranger
PATH PLANNING
Process LaserScan
Implement basic Wedge Bug
Test basic Wedge Bug
Implement boundary following
Test boundary following
OBSTACLE DETECTION
Implement point cloud collection
Test point cloud collection
Implement hole detection
Test hole detection
Implement obstacle detection
Test obstacle detection
Implement final laserscan
Test final laserscan
ODOMETRY
Add IMU Odometry
Test IMU Odometry
Add Extended Kalman Filter
Test Kalman filter
Add Wheel Odometry
Test Wheel Odometry
Add Visual Odometry
Test Visual Odometry
FINAL PRESENTATION

Works Cited

[1] B. Aldrin, “Apollo 11 bootprint,” July 1969.

[2] K. Kessel, “Regolith Advanced Surface Systems Operations Robot (RASSOR) Excavator.”

[3] “Intel Realsense D400 Series Product Family Datasheet,” pages 57 and 68, January 2019.

[4] “Intel Realsense Depth Camera 435i,” https://www.intelrealsense.com/depth-camera-d435i/.

[5] “sensor_msgs/Imu Documentation,” http://docs.ros.org/melodic/api/sensor_msgs/html/msg/Imu.html.

[6] M. Labbe, and F. Michaud, “RTAB-Map as an Open-Source Lidar and Visual SLAM Library for Large-Scale and Long-Term Online Operation,” *Journal of Field Robotics*, pages 8-10, 2019.

[7] S. L. Laubach, and J. W. Burdick. “An Autonomous Sensor-Based Path-Planner for Planetary Microrovers,” *Proceedings of the IEEE International Conference on Robotics and Automation*, May 1999.

Demonstration