# Roverside Assistance

Final Report

Team I (Moon Wreckers)

Daniel Arnett
Karthik (Venkata Rama) Paga
Dicong (David) Qiu
Matthew Swenson
Abdul Zafar

May 10, 2018

## Abstract

Planetary exploration robots have to date operated alone. Groups of two or more rovers can cover more terrain, take more risks, accomplish more planned goals, localize more effectively, and provide third-person views never before possible with solitary robots. This project explores rover entrapment and extrication by a companion rover.

This report documents the progress and final system evaluation results of the project by the MRSD Team I, the Moon Wreckers, towards a system capable of autonomous extrication of entrapped rovers. This report details the specific requirements that the system set out to meet, lays out solutions and subsystems designed to satisfy those requirements and the results of our final validation experiments.

# Contents

## 1 Project Description

In this project explores different scenarios of rover entrapment and proposes a systematical solution to address the problem of rescuing an entrapped rover with a companion rover.

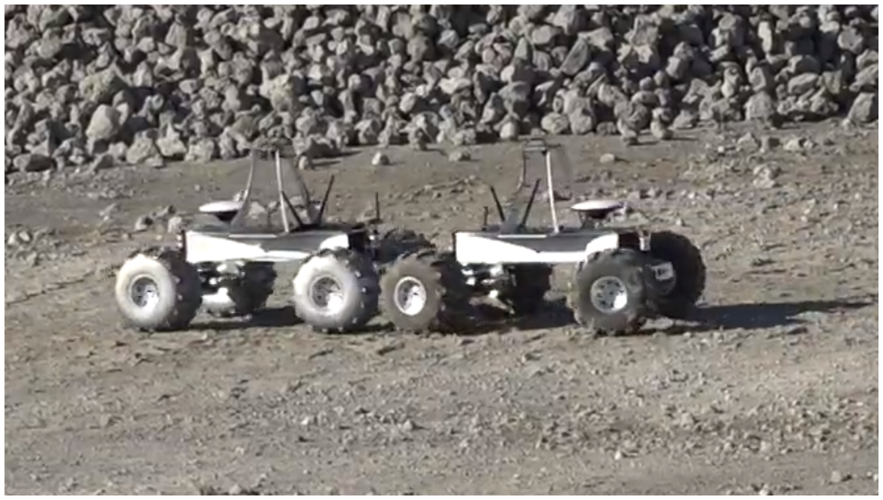


**Figure 1: Two rovers docking in a simulated lunar environment.**

More specifically, the goal of this project is to develop a symbiotic rover system with one rover assisting the other rover in diagnosis and resolution of terrain traversal problems. The system developed in the project gives these costly machines the capacity to extricate one another when high-centered. Such capacity shall enable exploration of terrain too steep and risky for a single rover. Our proposed system provides an additional safety guarantee for the future planetary rovers. With that, a higher level of autonomy can be built into these rovers systems so that direct human control will become less necessary.

The experimental platforms are adapted from a pair of AutoKrawler rovers developed in the Planetary Robotics Lab at Carnegie Mellon University, each of which utilizes double Ackermann steering drives and is equipped with an inertial measurement unit (IMU), wheel encoders, an on-board WiFi router, an HTC Vive Tracker and Base Station, a set of docking mechanism and a docking tactile force sensor. The software system is built based on the Robotics Operating System (ROS) and consists of nodes for Ackermann steering control, robot localization and relative pose estimation, dynamic mapping, auto-navigation with obstacle avoidance, entrapment detection, claw manipulator control for docking and abstract-level symbolic tasks planning with a state machine.

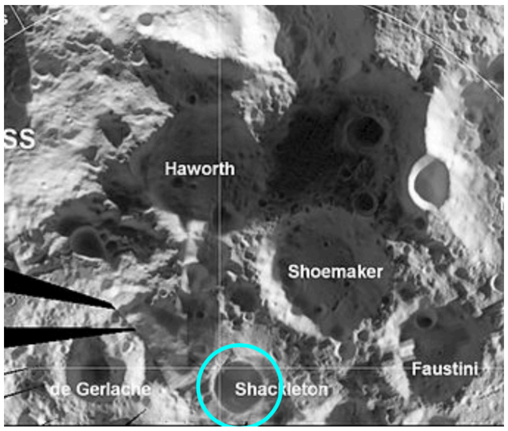
In the validation experiment, it was demonstrated that the symbiotic rover system is capable of autonomously detecting entrapment and rescuing the entrapped rover in different scenarios and difficulty levels of high-centered entrapments. In addition, component-level utilities have also been demonstrated, such as high-accuracy short-range localization and pose estimation with HTC Vive tracking system, dynamic construction of the environment cost map, path planning and autonomously navigation to destinations, and claw manipulator control for docking.

## 2 Use Case

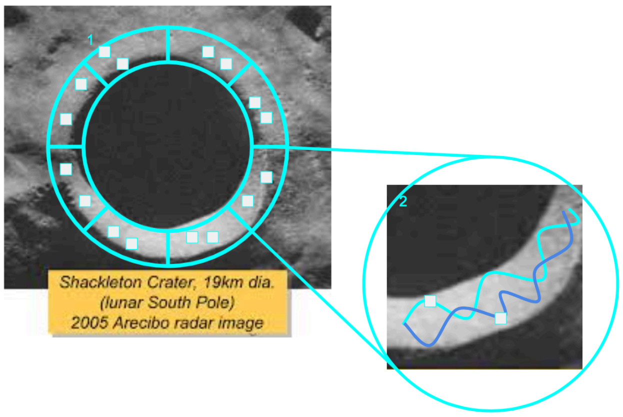
The year is 2050.

NASA is (finally) in the last stages of planning its first manned lunar base. They have a rough idea of where to Setup the base, but need dense surface maps to determine the best suitable location for the base, and to give its astronauts a good idea of their immediate surroundings when they arrive. To do this as quickly and thoroughly as possible, NASA has deployed a swarm of dozens of rovers to map the 25 km$^2$ potential landing zone. Considering the sheer complexity of having to directly monitor a large fleet of rovers, NASA has paid for Roverside Assistance to reduce the amount of required remote-human intervention with its rovers.

In addition to autonomous rover-rover interaction behaviors specifically for navigating in rough terrain which often lead to extreme dynamics, Roverside Assistance also includes integrated system capabilities to asses operating state of each individual rover in the fleet. Further, in case of distress situation the Roverside assistance system can autonomously queue a rescue mission to liberate an entrapped rover using a second rover as a rescuer. Finally, a pair of robots when operating as one cohesive unit generate 3D footage that is an accurate representation of each of the rover's in-situ interaction with the environment.



**(a) Shakleton Crater.**



**(b) Possible division.**

**Figure 2: NASA 2050 use case, with an image highlighting the location of the Shakleton Crater (left) and sample photo of a possible division of Shackleton Crater into segments to be searched by individual rover teams. Rover starting positions are indicated by squares (right).**

The rovers are landed in the search zone in pairs, the plan being for each pair to map out a few square kilometers in the course of a year. One such pair of rovers, named Block and Tackle, have been assigned a search zone that includes several areas with difficult terrain that is prone to high-centering rovers who cross it. Over the course of a year, Block and Tackle follow a rough route given to them by NASA, autonomously avoiding obstacles like boulders and crevasses while building up an accurate map of their environment by cross-referencing their surroundings with available satellite imagery and their own and relative odometry. While they typically roam out of sight of each other for greater coverage, they periodically meet back up to share map data and use each other's location to calibrate their own positions.

One day Tackle receives an SOS from Block, which reports it has just entered a boulder field and gotten high-centered. The system of rovers initiate the coordinate and rescue mode. With an updated map provided by satellite imagery and the 3D terrain map last constructed by the pair's efforts, Tackle can autonomously route to Block's location. As Tackle approaches the scene, the pair use Block's odometry and information from Tackle's sensing suite to accurately estimate Block's relative pose and decide the best plan of action for freeing Block.

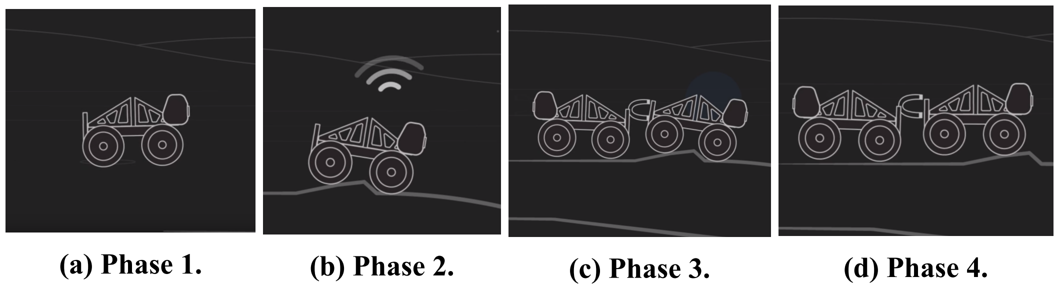

(a) Phase 1. (b) Phase 2. (c) Phase 3. (d) Phase 4.

**Figure 3: A storyboard of Block and Tackle demonstrating our system's in action, with 4 phases: Block exploring its environment (phase 1), Block signaling for help (phase 2), Tackle successfully docked for towing (phase 3), and Block has been dislodged (phase 4).**

The rovers determine that the best plan of action is to pull Block off of the rock against which it's high-centered. Tackle approaches Block along roughly the same path Block used. When it is close, they dock with each other using specialized integrated booms. Once a secure connection has been established, the rovers coordinate and simultaneously move in a predetermined direction to free Block. Within a few seconds, success! Block has been moved off of the rock it was stuck on. As a final step the rovers analyze and update their terrain maps to show that the area they are next to is unsafe for travel, and route around it as they continue on their mission!

## 3 System-Level Requirements

### 3.1 Mandatory Functional Requirements

M.F.0 Rovers shall autonomously navigate to commanded waypoints.

M.F.1 Rovers shall avoid large obstacles when autonomously navigating.

M.F.2 Rovers shall be able to dislodge a stuck rover from a high centering obstacle via a docking and towing maneuver.

M.F.3 Rovers shall autonomously perform entrapment detection.

### 3.2 Mandatory Nonfunctional Requirements

MN.0 Rovers shall be tele-operable.

## 3.3 Desirable Functional Requirements

D.F.0 Rover system shall be able to reason about its environment and know when to link rovers together before entering a hazardous area.

D.F.1 Rover system shall be able to reason about the circumstances and position of a stuck rover in an attempt to dislodge a stuck rover via pushing or other non-prehensile manipulation methods.

D.F.2 Rover system shall be able to purposefully lodge itself behind obstacles to exert greater force when towing.

## 3.4 Desired Non-Functional Requirements

D.N.0 Rover shall detect an unsuccessful docking attempt

D.N.1 Rover shall re-attempt docking in event of an unsuccessful dock

## 3.5 Mandatory Performance Requirements

M.P.0 Rovers will navigate to within 0.5 meters and 15° of the commanded waypoint and orientation.

M.P.1 Rovers will plan a route around any obstacle that is greater than 0.3 meters in height.

M.P.2 Rovers will be able to free a high centered rover in a 3D environment 80% of the time.

M.P.3 Rovers will detect when they are entrapped due to high centering in less than 60 seconds.

# 4 Functional Architecture

A graphical representation of the functional architecture of the project is presented in Figure 4.

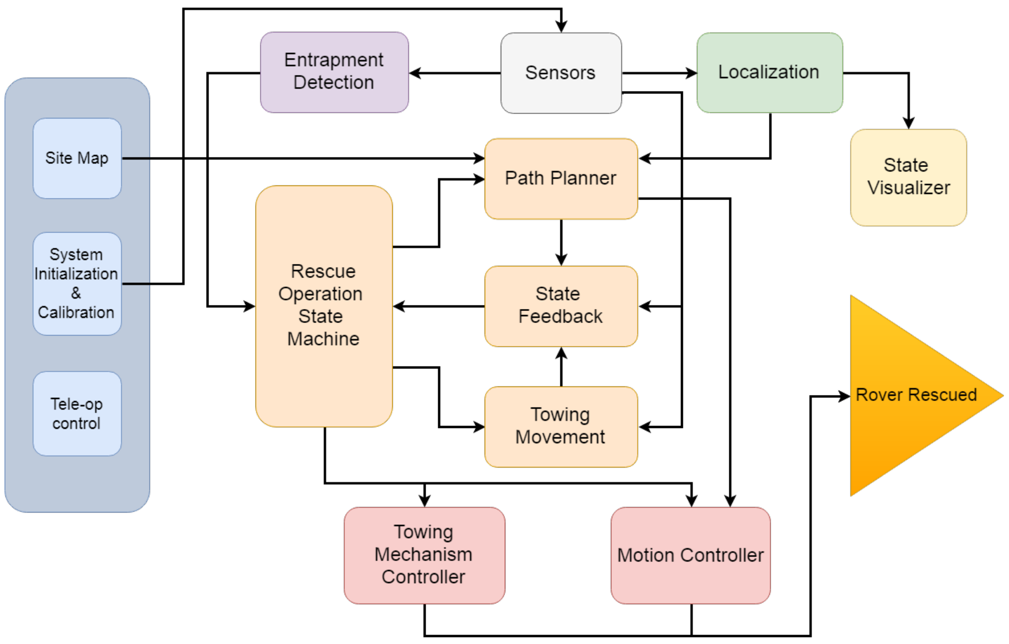


**Figure 4: Functional architecture**

The functional architecture is designed to fulfill all the previously mentioned functional requirements of the system.

The blue boxes signify the input required as the system turns up. ”Site Map” module develops a cost map of the environment which is fed into the ”Path Planner” module to generate motion plans while avoiding obstacles in the environment. ”System Initialization & Calibration” modules ensure proper calibration of the sensors and the initialization of reference frames involved in the project. Lastly, ”Tele-op Control” module provides the functionality of manually navigating the rovers in the environment using joy-stick.

The ”Entrapment Detection” module uses the sensor data to continuously monitor the state of entrapment of the rovers while they are navigating in the environment. On the other side, the ”Localization” module keeps track of the rovers in terms of 6D pose which is continuously fed into the ”Path Planner” module to generate valid plans for navigation of the rovers. At the same time, the rover’s pose information is also fed into the ”State Visualizer” module to visualize the current state of the rovers in simulation.

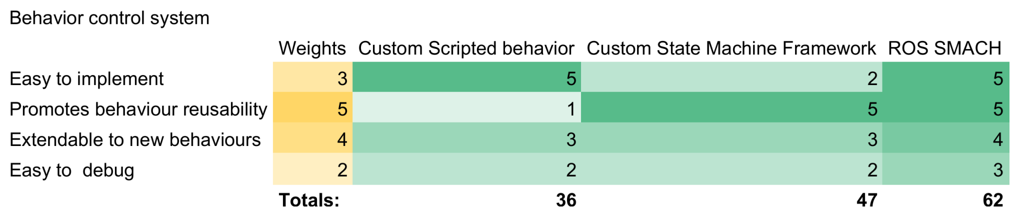
In the event of entrapment detection, the ”Rescue Operation State Machine” initiates rescue operation of the entrapped rover. The state machine module calls the ”Path Planner” module to plan/re-plan a path for the rescue rover to dock with the entrapped rover. The ”Towing Movement” module ensures successful docking as well as coordinated movement of the rovers once the rescue rover docks the entrapped rover. The ”State Feedback” module uses the information from the ”Path Planner”, ”Towing Movement” and ”Sensor” module to determine the appropriate transitions of the rescue operation state machine.

The physical actions determined by the ”Rescue Operation State Machine” module and the planned path generated by the ”Path Planner” module are fed into the controller modules.

”Towing Mechanism Controller” module provides the software-hardware interface for the dock and tow mechanism. The ”Motion Controller” modules provides the interface for driving the wheel motors in accordance with the desired action as determined by the rescue operation state machine. Upon execution of the correct rescue operation, the entrapped rover is liberated which is signified by the ”Rover Rescued” state.

## 5 System Level Trade Studies

### 5.1 Behavior Control System



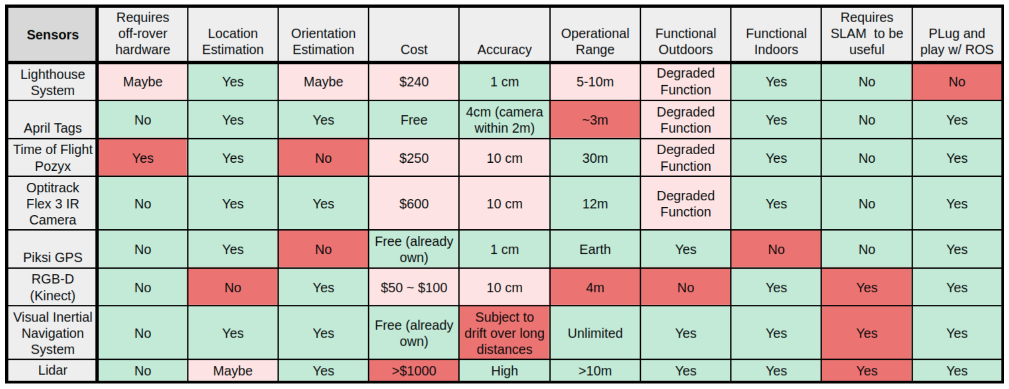
Behavior control system

| | Weights | Custom Scripted behavior | Custom State Machine Framework | ROS SMACH |
|---|---|---|---|---|
| Easy to implement | 3 | 5 | 2 | 5 |
| Promotes behaviour reusability | 5 | 1 | 5 | 5 |
| Extendable to new behaviours | 4 | 3 | 3 | 4 |
| Easy to debug | 2 | 2 | 2 | 3 |
| | **Totals:** | **36** | **47** | **62** |

**Figure 5: Comparison of different strategies to control the rovers.**

As we moved towards full system integration, an overarching controller for the entire towing behavior was required. A system that afforded modular sub-behaviors that could be repeated and rearranged easily was desired. The ROS SMACH (state machine) library served that purpose very well with very little time to learn or implement the system.

### 5.2 Localization System



| **Sensors** | Requires off-rover hardware | Location Estimation | Orientation Estimation | Cost | Accuracy | Operational Range | Functional Outdoors | Functional Indoors | Requires SLAM to be useful | PLug and play w/ ROS |
|---|---|---|---|---|---|---|---|---|---|---|
| Lighthouse System | Maybe | Yes | Maybe | $240 | 1 cm | 5-10m | Degraded Function | Yes | No | No |
| April Tags | No | Yes | Yes | Free | 4cm (camera within 2m) | ~3m | Degraded Function | Yes | No | Yes |
| Time of Flight Pozyx | Yes | Yes | No | $250 | 10 cm | 30m | Degraded Function | Yes | No | Yes |
| Optitrack Flex 3 IR Camera | No | Yes | Yes | $600 | 10 cm | 12m | Degraded Function | Yes | No | Yes |
| Piksi GPS | No | Yes | No | Free (already own) | 1 cm | Earth | Yes | No | No | Yes |
| RGB-D (Kinect) | No | No | Yes | $50 ~ $100 | 10 cm | 4m | No | Yes | Yes | Yes |
| Visual Inertial Navigation System | No | Yes | Yes | Free (already own) | Subject to drift over long distances | Unlimited | Yes | Yes | Yes | Yes |
| Lidar | No | Maybe | Yes | >$1000 | High | >10m | Yes | Yes | Yes | Yes |

**Figure 6: Comparison of several sensors used to localize the rovers.**

Localization Subsystem

| | Weights | Vive Tracker | Visual Odometry | Vicon Mocap |
|---|---|---|---|---|
| Easy to implement | 2 | 2 | 4 | 1 |
| Accuracy | 5 | 4 | 3 | 5 |
| Works outdoors | 4 | 4 | 3 | 3 |
| Cost | 3 | 4 | 4 | 1 |
| Range | 1 | 2 | 2 | 3 |
| | **Totals:** | **54** | **49** | **45** |

**Figure 7: Evaluation of the best sensors from the earlier trade study.**

Our overall relative localization tolerance ended up needing to be about 5 cm. Systems that could provide that accuracy reliably and cheaply in the face of significant physical shocks and rapid dislocations were required. The Vive Tracker system proved a highly reliable and accurate localization scheme with the added benefit of working accurately in 6 DOF.

## 5.3 Path Planning System

Path Planning System

| | Weights | ROS Move Base Default | Custom Planner | TEB Local Planner |
|---|---|---|---|---|
| Accounts for kinodynamic constraints | 5 | 0 | 5 | 4 |
| Easy to implement | 3 | 5 | 1 | 4 |
| Fast planning time | 2 | 4 | 2 | 4 |
| Creates reasonable paths | 4 | 0 | 2 | 4 |
| | **Totals:** | **23** | **40** | **56** |

**Figure 8: Comparison of different options for creating movement plans for the rovers.**

The path planning system provides the necessary navigation capabilities for one rover to perform rescue operations on the entrapped rover. For this purpose, it is imperative to have a path planning system which generates a plan that avoids potential obstacles as well as be compliant with the kinodynamic constraints of the Ackermann-steered rovers. Consequently, higher weightage was given to the factors ”Accounts for kinodynamic constraints” and ”creates reasonable paths”. The ROS Move Base Default planner does not cater for non-holonomic planning scenarios and therefore, was not a viable option for the project. Although a custom planner was specifically developed and tuned according to the rovers kinematic constraints, the resulting trajectories were not optimized locally and were not deemed as reasonable paths to follow. TEB local planner proved to have the best overall performance especially in terms of generating locally optimal and kinematic-ally feasible trajectories while having faster planning times.

## 5.4 Docking Mechanism System

Claw system

| | Weights | Magnet | Claw | Hook |
|---|---|---|---|---|
| Connection strength | 5 | 2 | 5 | 4 |
| Misalignment tolerance | 4 | 1 | 4 | 3 |
| Mechanism durability | 3 | 3 | 4 | 4 |
| Mechanism complexity | 2 | 2 | 4 | 3 |
| | **Totals:** | **27** | **61** | **50** |

**Figure 9: Comparison of different options for forming a physical link between rovers.**

The core of our project was autonomous rescue from high centering conditions, so the reliable performance of the docking mechanism was paramount. Using an actuated claw mechanism provided a good tradeoff between strength, durability, and simplicity in alignment.

## 5.5 Claw Arm System

Claw arm Configuration

| | Weights | Fixed Actuator | 1 DOF Arm | 2 DOF Arm | 3 DOF Arm |
|---|---|---|---|---|---|
| Misalignment Tolerance | 5 | 2 | 3 | 5 | 5 |
| Complexity | 2 | 1 | 4 | 3 | 2 |
| Durability | 3 | 3 | 3 | 2 | 1 |
| | **Totals:** | **21** | **32** | **37** | **32** |

**Figure 10: Comparison of methods for getting the docking mechanism into alignment.**

During initial testing with the claw mechanism it was discovered that the rover planning and drivetrain systems were not physically accurate enough to maneuver the claw into the correct location even in planar scenarios, and a fixed actuator large enough to overcome misalignment in 3D scenarios is cumbersome. Similarly, a 1 DOF arm relied too heavily on accurate control of the rover's relative y-axis location. A 2 DOF arm provides an acceptable amount of tolerance while keeping complexity of implementation and control low.

| | Weights | COTS Pan Tilt Mechanism | Hebi Brackets | Custom Pan Tilt Mechanism |
|---|---|---|---|---|
| Durability | 5 | 2 | 5 | 4 |
| Torque | 1 | 1 | 5 | 4 |
| Cost | 2 | 5 | 1 | 3 |
| Easy to implement | 4 | 3 | 5 | 1 |
| | **Totals:** | **33** | **52** | **34** |

**Figure 11: Comparison of methods for powering a 2 DOF arm.**

Having fixed on a 2 DOF arm, deciding how to actuate it required further study. Most commercial pan-tilt mechanisms have an unacceptably low torque to weight/volume ratio. Hebi motors are compact, strong, and came with an easily worked with ecosystem of parts and software that made interfacing with them easy enough to offset their high cost.

## 6 Cyberphysical Architecture

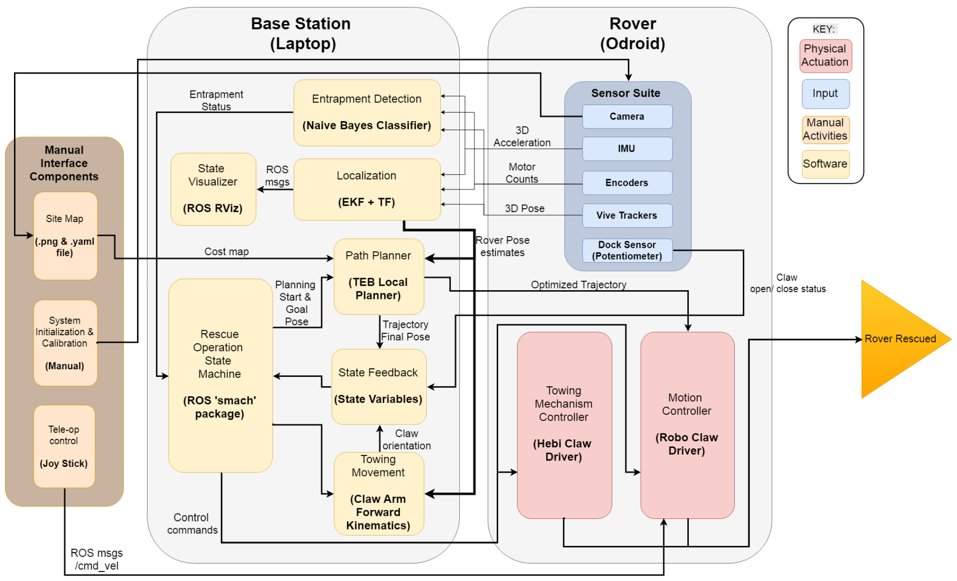

Base Station (Laptop)
Rover (Odroid)
KEY:
Physical Actuation
Input
Manual Activities
Software
Manual Interface Components
Site Map (.png & .yaml file)
System Initialization & Calibration (Manual)
Tele-op control (Joy Stick)
Entrapment Status
Entrapment Detection (Naive Bayes Classifier)
State Visualizer (ROS RViz)
ROS msgs
Localization (EKF + TF)
Cost map
Path Planner (TEB Local Planner)
Planning Start & Goal Pose
Rescue Operation State Machine (ROS 'smach' package)
Trajectory Final Pose
State Feedback (State Variables)
Claw orientation
Towing Movement (Claw Arm Forward Kinematics)
Control commands
ROS msgs /cmd_vel
Sensor Suite
Camera
IMU
Encoders
Vive Trackers
Dock Sensor (Potentiometer)
3D Acceleration
Motor Counts
3D Pose
Rover Pose estimates
Optimized Trajectory
Claw open/ close status
Towing Mechanism Controller (Hebi Claw Driver)
Motion Controller (Robo Claw Driver)
Rover Rescued

**Figure 12: Cyberphysical architecture.**

This section describes the cyberphysical architecture of the system. A graphical representation of the cyberphysical architecture is presented in Figure 12.

At a high level, the system is divided into two parts: mobile rovers which execute motion commands and collect information about their environment, and a stationary base station which communicates with the rovers wirelessly over Wi-Fi and performs data synthesis along with any computationally complex calculations, such as path planning, numerous TF transforms, entrapment detection, and overall behavior control.

Our base station laptop loads our stored site map of the test environment, and is responsible for hosting our ’roscore’ process. The site map is generated off-line using a camera. Initialization and calibration of the system is carried out off-line as well when the system is turned on. In order to manually drive the rovers, joy-stick control interface is developed to tele-op them.

The rovers carry the following sensors: wheel encoders, an IMU, an HTC Vive VR system tracker and a potentiometer. The primary purpose of encoders, IMU and HTC Vive trackers are for localization and entrapment detection and all are run through an EKF. A Bayesian classifier, running on base station, uses rover telemetry to determine if one of the rovers is stuck. The rovers use a double Ackermann drive system for steering, and both the motor attached to the rack-and-pinion and the motor attached to the transmission are controlled by a Roboclaw motor driver, which performs its own PID control on the drive motor velocity. On-board processing is performed by an ODROID XU4 single board computer. The rovers are also equipped with a claw-arm powered by HEBI motors that serve the purpose of towing the other rover. The control of the claw arm is achieved with the HEBI motor drivers.

Using the loaded map and rover pose estimates, TEB local planner generates a feasible path through the environment. Towing behaviors are controlled from the base station through a state machine which is implemented using the ROS SMACH package. However, the claw will close automatically if it detects something has entered it's grabbing envelope to avoid potential latency issues. The base station also performs the calculations to generate the control angles for the claw arm. The planned trajectory and control angles for the claw arm are sent over to the rover via Wi-Fi, where an on-board computer uses it as well as its own telemetry data to perform a rudimentary closed loop positioning control.

## 7 System Overview

The symbiotic rover system, as shown below in figure 14, consists of rover hardware assembly, sensor suite, actuator suite, state estimation subsystem, entrapment detection subsystem, docking and towing subsystem, path planning and simulation subsystem, and motion control subsystem. In addition, rovers are also equipped with communication modules, power modules and necessary computing resources.

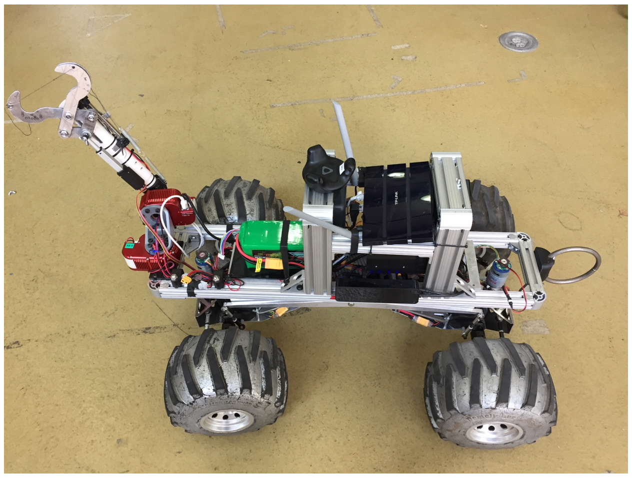


**Figure 13: Side view of the rescuing AutoKrawler.**

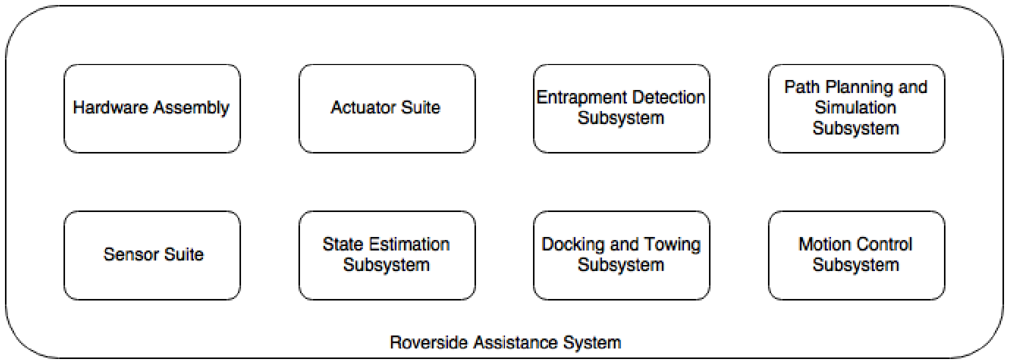

Hardware Assembly
Actuator Suite
Entrapment Detection Subsystem
Path Planning and Simulation Subsystem
Sensor Suite
State Estimation Subsystem
Docking and Towing Subsystem
Motion Control Subsystem
Roverside Assistance System

**Figure 14: Current system and important subsystems.**

## 7.1 Subsystem Design, Modeling and Analysis

### 7.1.1 Rover Hardware Assembly

The hardware assembly of the rover consist of the following major sub-systems:

**Mobile Base.** The chassis of the rover is built base on Ackermann steering [1] with differential drive. The mobile base as shown in figure 15 is capable of traversing across rough terrain and also navigating along steep inclinations up to 45° while affording a safe tilt angle of approximately 60°.

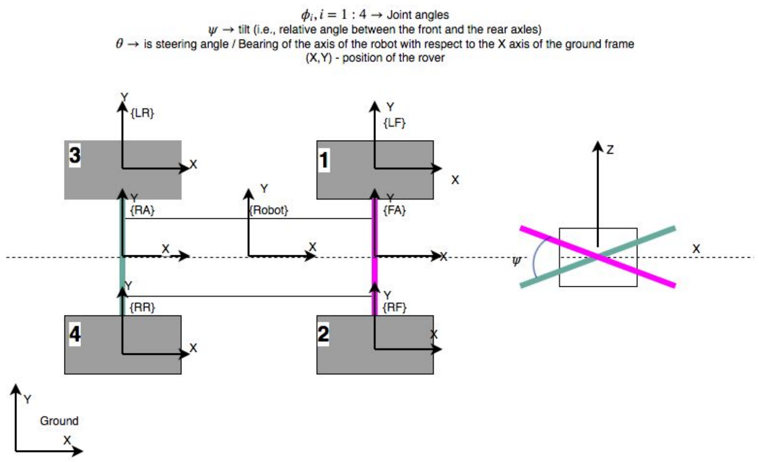

$\phi_i, i = 1:4 \rightarrow$ Joint angles
$\psi \rightarrow$ tilt (i.e., relative angle between the front and the rear axles)
$\theta \rightarrow$ is steering angle / Bearing of the axis of the robot with respect to the X axis of the ground frame
(X,Y) - position of the rover
{LR}
{LF}
{RA}
{Robot}
{FA}
{RR}
{RF}
1
2
3
4
X
Y
Z
$\psi$
Ground

**Figure 15: Representation of the mobile base.**

**Rover Body.** A roller cage atop the mobile base, houses the sensors, actuators and docking ports on-board each robot. In addition to structural stability, the roller cage design also protects the sensors from any physical damage in case of a roll-over when operating in the rocky field. As seen in figure 13, the roll cage houses a manipulators, battery pack, electronics, sensors and a circular hook which acts as a docking port on each robot.

**Claw Manipulator.** During a rescue mission, the rescue robot follows/plans an approach path in a region pre-explored by the system of rovers. In this consideration, there are very limited non-prehensile activities, for e.g. push, pull etc., that the mobile robot can afford to perform and more generally, any such activity in a disaster zone involves the potential risk of compromising the rescue rover as well. But the task of dislodging a stuck/ entrapped rover involves non-prehensile motion. Thus one of the two rovers, designated as the rescue rover, is equipped with a 2-DOF (pitch and yaw) manipulator so as to increase the volume of the grasp-cone which increases the likelihood of a successful docking attempt and decreases the necessity to continuously re-plan the path of the mobile base in order to ensure docking.

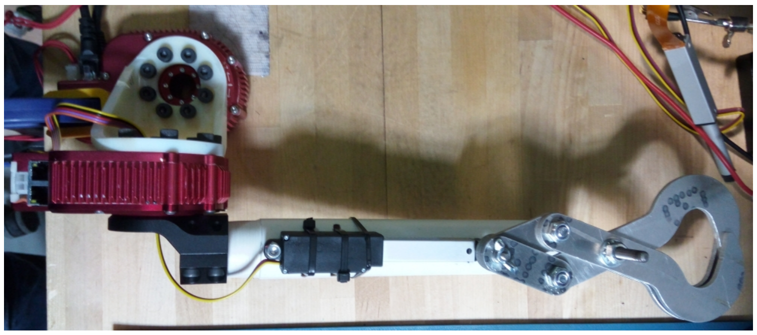


**Figure 16: Standalone assembly of the claw manipulator.**

### 7.1.2 Sensor Suite

The sensor suite of an individual rover consists of an inertial measurement unit (IMU), a set of wheel encoders, a set of HTC Vive tracking system (consisting of a HTC Vive tracker and a HTC Vive base station) [2], wheel encoders, and a tactile force sensor for docking.

The inertial measurement unit senses the ego-pose information, including rover acceleration, angular velocity and global orientation with respect to the Earth. The wheel encoders measure the rotation speed of the wheels, which can be further processed and filtered into the wheel odometry. The HTC Vive tracking system, as shown in figure 17 gives high-accuracy relative pose estimation between a pair of rovers down to 2.0 mm. The filtered odometry incorporate the perceived information from the encoders, the IMU and the HTC Vive tracking system using an extended Kalman filter, providing flexible localization and pose estimation for the rover system. The docking tactile force sensor is actually a linear potentiometer attached to a springloaded tripwire strung between the claw tips, which detects the status of docking.

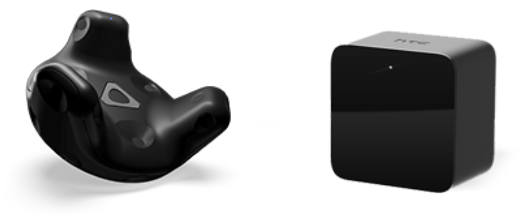


**Figure 17: An image of an HTC Vive tracker (Left) and an HTC base station (Right).**

The incorporated HTC Vive tracking system consists of at least one tracker and a base station (also known as a lighthouse, or a beacon). A base station sweeps a planar laser horizontally and a planar laser vertically across the scene in a synchronous manner. Multiple photo-diodes embedded in a HTC Vive tracker detect the laser sweeps and calculate the time of flight for those sweeps to reach them so as to solve for the relative pose of themselves with respect to the base station. With the known constellation of the photo-diodes, the poses of the photo-diodes can be jointly solved so as to provide information about the position and orientation of the tracker.

### 7.1.3 State Estimation Subsystem

The current state estimation subsystem uses an extended Kalman filter (EKF) to merge different sensor measurements from the wheel encoders (wheel odometry), the IMU and the HTC Vive tracking system. The output of the state estimation subsystem is a filtered odometry that provides optimal estimation of a rover's current pose.

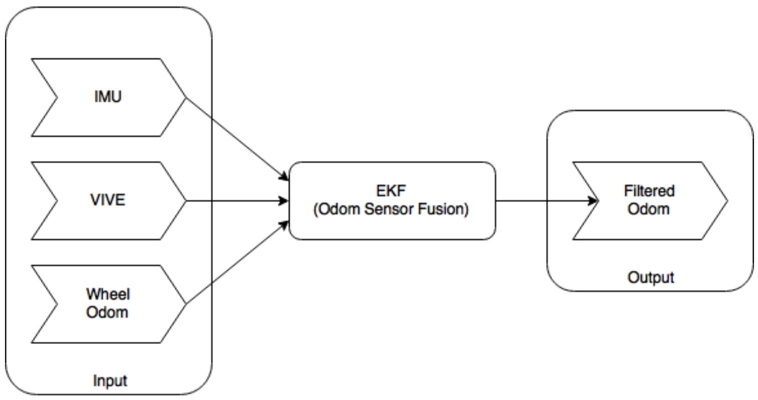


**Figure 18: Current EKF input and output.**

The IMU directly measures the acceleration, the orientation and the angular velocity of the rover in 3-space, which are filtered into the EKF loop in the prediction stage. The wheel odometry is derived from the raw measurement from the wheel encoders, where the current rover's planar velocity, position and orientation are given. And the HTC Vive tracking system can provide high-accuracy measurement of the rover's position, orientation, linear velocity and angular velocity with respect to the HTC Vive base station (within certain range of operation). The wheel odometry and the HTC Vive odometry are jointly filtered into the update stage of the EKF loop.

An advantage of using the HTC Vive tracking system is the measurement accuracy it can reach, which is within about 2.0 millimeters in terms of position and 1.0 degree in terms of orientation. The drawback is that it relies on the existence of a HTC Vive base station, meaning that a HTC Vive tracker must be operated within a 5-meter envelope of the HTC Vive base station to function properly. The HTC Vive tracker will stop publishing measurement information if the HTC Vive base station is too far away, in which case the EKF module will then rely only on the IMU and the wheel odometry. An improvement in the future work will be incorporating measurement from GPS and/or a visual odometer [3, 4] into the sensor suite, so that the rovers can operating beyond the HTC Vive base station operational range.

### 7.1.4 Entrapment Detection Subsystem

The entrapment detection subsystem [5] takes in multiple sensor readings and provides real-time entrapment status estimation along with a likelihood indicating the confidence of the estimation. An entrapment of a rover is formally defined by the divergence (disagreement) of the assumed rover velocity derived from the wheel encoders, and the ground-truth rover velocity which can be approximated by the measured rover velocity given by sensor readings from other than wheel encoders:

$$\|\dot{\mathbf{x}}_g\| < 0 + \epsilon_0 \; and \; \left\| \frac{\partial \mathbf{FK}(\mathbf{q})}{\partial \mathbf{q}} \dot{\mathbf{q}} - \dot{\mathbf{x}}_g \right\| > \epsilon_{ag}$$

Where $\dot{\mathbf{x}}_g$ is the ground-truth rover velocity, $\mathbf{q}$ is the rover locomotion actuator joint position, $\epsilon_0$ and $\epsilon_{ag}$ are maximum tolerable errors. The first inequality indicates the rover does not move under a tolerable error, and the second inequality indicates the rover velocity estimated from the locomotion actuator odometry diverges from the ground-truth rover velocity beyond a tolerable error. Intuitively, a rover is considered entrapped if its true velocity is nearly zero and the assumed velocity disagrees with its true velocity.

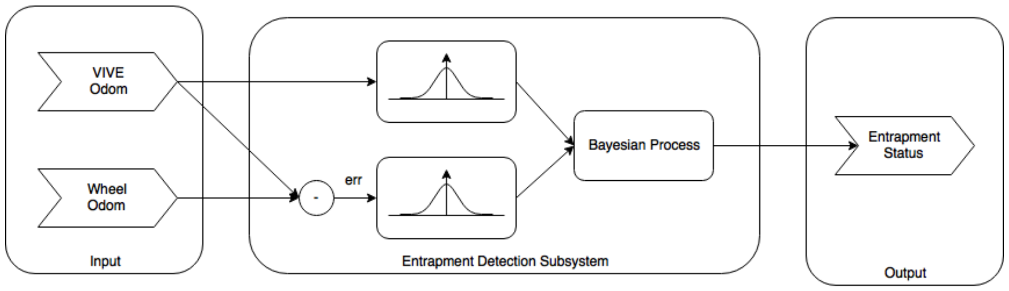


**Figure 19: Entrapment detection subsystem.**

Practically, the rover velocity estimated from the wheel odometry is used as the assumed rover velocity, and that from the HTC Vive odometry, which can be replaced with GPS or visual odometry [3, 4], is used to approximate the ground-truth rover velocity. The approximated velocity divergence criteria is redefined as a weighted error

$$Q = \sqrt{\begin{bmatrix} e_v & e_\omega \end{bmatrix} R \begin{bmatrix} e_v \\ e_\omega \end{bmatrix}}$$

where $e_v = \|\mathbf{v}_a - \mathbf{v}_m\|$ is the linear velocity error between the assumed linear velocity and the measured linear velocity, $e_\omega = \|\omega_a - \omega_m\|$ is the angular velocity error between assumed and measured angular velocities, and $R$ is the weight matrix.

Prior knowledge of the variance of static rover velocity measurement and acceptable velocity divergence are assumed to conform to a Gaussian distribution and the parameters of the Gaussian models are derived by data collected from experiments. The rover statics probability and velocity agreement probability are then treated as inputs to a Bayesian process to give the likelihood of rover entrapment.

$$\mathbf{Pr}(D|Q) = \frac{\mathbf{Pr}(Q|D)\mathbf{Pr}(D)}{\sum_d \mathbf{Pr}(Q|D=d)\mathbf{Pr}(D=d)}$$

where $D \in \{\textit{diverged}, \textit{consistent}\}$ is the rover velocity divergence status random variable, and $Q \in \mathbb{R}_{\geq 0}$ is the weighted quadratic divergence between the assumed rover velocity and the measured rover velocity. The prior probability $\mathbf{Pr}(D)$ is from initialization or the posterior probability $\mathbf{Pr}(D|Q)$ from the previous estimation.

The movement status of the rover is estimated by the norm of the measured rover velocity. The probability whether the rover is moving or not

$$\mathbf{Pr}(M|\|\mathbf{v}_m\|) = \frac{\mathbf{Pr}(\|\mathbf{v}_m\||M)\mathbf{Pr}(M)}{\sum_m \mathbf{Pr}(\|\mathbf{v}_m\||M=m)\mathbf{Pr}(M=m)}$$

where $M \in \{\textit{moving}, \textit{stopped}\}$ is the rover movement status random variable, and $\|\mathbf{v}_m\| \in \mathbb{R}_{\geq 0}$ is the measured rover velocity norm random variable. The prior probability $\mathbf{Pr}(M)$ is from initialization or the posterior probability $\mathbf{Pr}(M|\|\mathbf{v}_m\|)$ from the previous estimation.

After the posterior probabilities are estimated, the probability that the rover is entrapped can be estimated using the conditional independence property of Naive Bayes classifiers.

$$\begin{aligned}
&\mathbf{Pr}(S = \textit{entrapped}|Q, \|\mathbf{v}_m\|) \\
=&\mathbf{Pr}(D = \textit{diverged}, M = \textit{stopped}|Q, \|\mathbf{v}_m\|) \\
=&\mathbf{Pr}(D = \textit{diverged}|Q)\mathbf{Pr}(M = \textit{stopped}|\|\mathbf{v}_m\|)
\end{aligned}$$

where $S \in \{\textit{entrapped}, \textit{slipping}, \textit{moving}, \textit{stopped}\}$ is the rover status random variable, and an entrapment is assumed if the joint condition is met that the rover velocity is diverged and the measured rover velocity suggests that the rover is stopped. A bonus of adopting this approach is that, instead of giving only the entrapment status estimation, it provides a complete rover status estimation along with probabilities or confidence levels assigned to each status.

### 7.1.5 Map Generation System

To detect environmental obstacles, a point cloud of the test environment was constructed using the Intel Realsense D435 depth camera. The depthmap information from the Realsense was combined with the highly accurate odometry information form a Vive tracker and fed into the RTAB-Map[6] package to yield significantly more accurate pointclouds than Realsense is capable of generating by itself. From the point cloud surface normals are estimated and used to detect high slope features in the terrain. These features are then labeled as obstacles In the map and avoided by the path planner, as depicted in 20.

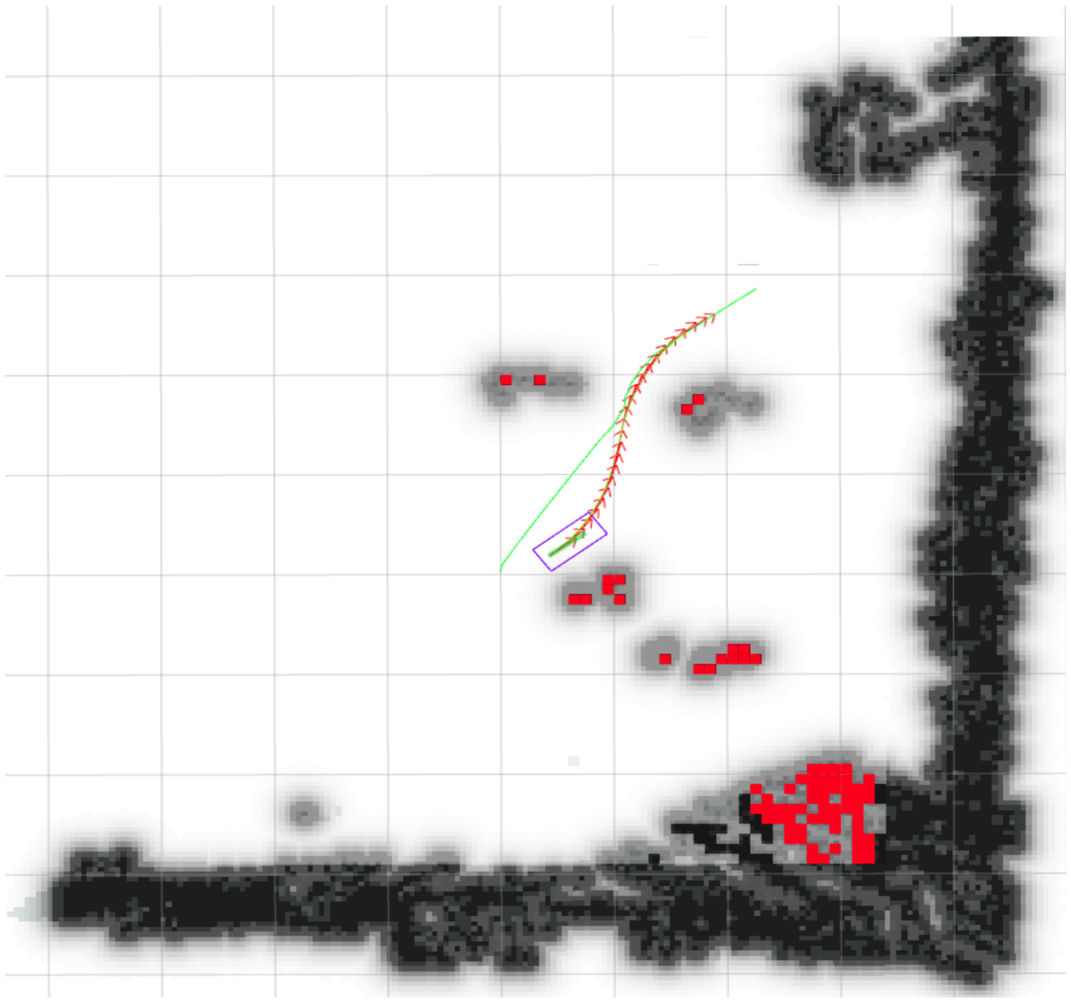


**Figure 20: Depiction of the costmap generated for SVE, with rover model planning through it**

### 7.1.6 Path Planning Subsystem

The approach angle and the planned path for a mobile rover are reliant on the known motion capabilities of the rover. Henceforth, these motion capabilities would be referred to as primitives or specifically *motion primitives*. The motion primitives are determined based on the design and calibration of the fabricated and assembled rover. Experiments were conducted Experiments were conducted, as shown in figure 27, to determine the first cut-estimate of the minimum-turning radius of the 4x4 Ackermann driven AutoKrawler rovers. The details of the experiment is presented in section 7.2.3.

The path planning subsystem handles the functionality of generating feasible motion plans for a rover to a desired goal position and orientation while avoiding obstacles and respecting the kinematic constraints of the rover. In the event of entrapment of one rover (stuck rover), the other rover (rescue rover) plans a feasible to the stuck rover in order to dock and tow it.

Towing another rover necessitates planning/re-planning a path to a desired position and orientation while respecting kinodynamic constraints of the rover. Time-Elastic-Band (TEB) planner generates locally optimal trajectories online while avoiding obstacles, minimizing execution time and being compliant with desired velocities and accelerations. The implementation of TEB planner is provided as a plug-in to the ROS navigation stack in a ROS package "teb_local_planner". Figure 21 shows an instance of TEB local planner planning a path (shown in green line) along with an generating an optimized trajectory dynamically while taking into the various constraints of the system (shown in red). A dynamic obstacle (shown as red square) is also shown in figure 21.

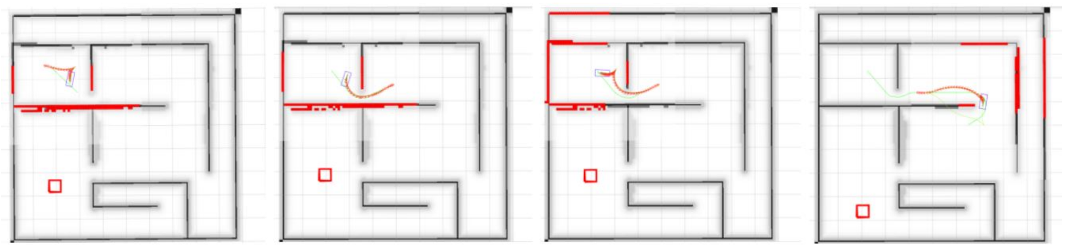


**Figure 21: Simulation of TEB local planner along with visualization of dynamic obstacle.**

When a rover is entrapped, the pose (position and orientation) of the stuck rover is registered as the goal state for path planning. TEB local planner uses the current state of the rescue rover as the start state for path planning and carries out dynamic path planning and re-planning as well as optimizing the trajectories generated to navigate the rescue rover. Numerous parameters were tuned and tested in order to achieve optimal performance of the planner for this project. Parameters that were relevant to successful functioning of the project are as follows: (1) `max_vel_x`: Maximum translational velocity of the robot in meters/sec; (2) `min_turning_radius`: Maximum translational velocity of the robot in meters/sec; (3) `xy_goal_tolerance`: Allowed final euclidean distance to the goal position in meters; (4) `free_goal_vel`: Remove the goal velocity constraint such that the robot can arrive at the goal with maximum speed; (5) `min_obstacle_dist`: Minimum desired separation from obstacles in meters; (6) `inflation_dist`: Buffer zone around obstacles with non-zero penalty costs; (7) `weight_viapoint`: Optimization weight for minimizing the distance to via-points.

A brief overview of the effect of the parameters on the planning behavior along with their tuning considerations is provided in the table below.

| Parameter | Effect/Tuning Considerations |
|---|---|
| `max_vel_x` | This parameter was lowered in order to allow more time for the HEBI Claw arm to constantly orient itself towards the ring in a timely fashion as the rover is navigating towards to the stuck rover. |
| `min_turning_radius` | Although the minimum turning radius of the rover theoretically was about $0.4$ m, it was observed through testing that keeping it at a larger value proved to be more effective. This is attributed to the hardware limitations and inconsistencies involved in a real world. It was also observed that increasing this parameter resulted in paths that were more conducive with the rovers to follow. |
| `xy_goal_tolerance` | This parameter has a direct relation with the reachability space of the claw arm and therefore the reachability space of the claw arm was taken into account in determining the value of this parameter. |
| `free_goal_vel` | This was set to `false`, i.e. not allowing the robot to reach the goal pose with max velocity. This is to prevent overshooting when reaching the desired goal pose. |
| `min_obstacle_dist` | The stuck rover is represented as an obstacle so that the rescue rover does not create infeasible paths through the stuck rover. If this parameter is set low, the rescue rover may generate plans that involve collision with the other rover. If this is set too high, the rescue rover will fail to plan a path to the desired pose which ensures successful docking. |
| `inflation_dist` | Increasing this causes the planner to generate plans that tends to circumvents obstacles from a greater distance. This parameter was tuned in combination with the `min_obstacle_dist` to achieve the desired behavior of the rovers. |
| `weight_viapoint` | This parameter specified how much importance should be given in following the intermittent way points. Making this too low caused the rover to go off track from the planned path. Making this too high caused the rover to be too tightly constrained in hitting every way point which added to the jerky motions of the rover at various occasions. |

The commands from the planner are published and subscribed by the motion controller which controls the operation of drive motors. We employ off-the shelf motor control drivers, `Roboclaw`, in order to modulate and regulate the supply currents to each individual actuator. Figure **??** represents the navigation pipeline on-board each of the rovers.

### 7.1.7 Power Distribution

To protect the electrical components and batteries several safety mechanisms were added. First, rover batteries plug directly into a 60A circuit breaker. Overall current draw from the system should never exceed approximately 55A, so 60A was deemed a reasonable choice. The circuit breaker mainly provides a convenient kill switch, and prevents the possibility of a battery overcharge due to a mistakenly connected battery in our parallel harness, which allows hot swapping smaller batteries during operation of the rovers. This practice reduced downtime early in the year, but in practice by the end of the year we had found the 16Ah 24V batteries purchased to powers the Hebis were sufficient to operate a rover for many hours at a time. The Hebis are directly powered by the 24V battery, but the rest of the rover electronics are powered through a 40A 24V to 12v converter. The current then flows through a 30A fuse, and then through a low voltage cutoff device which disconnects the rovers if voltage drops below 11.7V. This precludes damage to the batteries due to overdischarge. Finally power is then distributed to all 12 volt components, and a UBEC converter provides 5 volts to our computer and powered USB hub.

### 7.1.8 Docking and Towing Mechanism

The rover docking adopts a reactive strategy, where the claw manipulator on a rover always points its end effector (the claw) towards the ring of its companion rover, so that when the rescue rover approaches the entrapped rover, the docking mechanisms shall dock together automatically. After a successful docking, both rovers will move towards the same direction to get rid of the entrapment.

**Claw.** The claw consists of a linear actuator connected to a modified 4 bar linkage, both fixed on a lightweight 0.3 m long PVC pipe. The claw closes when the actuator contracts. The actuator has an internal potentiometer feedback that allows close control of the position of the claw. To aid in detection of the tow ring affixed to the other rover, a tripwire was run from the tips of the claw to a spring loaded linear potentiometer. Whenever an object depresses the tripwire, the linear potentiometer moves and software knows to close the claw. To reduce noise from the potentiometers and guard against momentary shocks due to the movement of the rovers, a rolling average over the previous 0.5 seconds is used to determine the current status of the claw position and tripwire.

The rolling average data also allows for a form of stall protection when closing the claw. By taking the standard deviation of the values used for the rolling average, a rough velocity of the claw actuator is determined. When the claw is closing and the standard deviation drops below a threshold, the claw control code assumes the claw is closed on something. This protects the motor from stalling out and allows the claw to grasp objects too large to fit inside its enclosed radius.

**2-DOF Claw Manipulator.** The pipe the claw is connected to is attached to a 2-DOF motor assembly constructed out of stock HEBI actuators and mount kits. HEBI motors come with a T-slot profile compliant mount plates and purchasable right angle brackets, allowing us to easily construct a simple robot arm. The HEBI actuators bear the forces associated with towing. We assumed that rated torque gave a roughly acceptable measure of how much force could be applied to the motor without damage, and selected models with 20 Nm stall torque.

**Autonomous Tracking Algorithm for Claw Manipulator.** The objective that the claw manipulator always points its end effector towards the ring of its companion rover can be formally formulated as an minimizing the cross-product of the claw manipulator end-effector vector and the ring vector, where the claw manipulator end-effector vector $\mathbf{v}_e$ is vector from the last joint of the claw manipulator to the end-effector, the ring vector is the vector from the last joint of the claw manipulator to the ring of the companion rover, and the zero point of the cross product indicates these two vectors become parallel. In order to remove the ambiguity that the two parallel vectors can be either point towards the same direction or the exact opposite directions, the sign of the dot-product of the two vectors can be taken advantage of, where a positive dot-product indicates the two vectors point within the same semi-sphere, while a negative one indicates they form an obtuse angle. The objective function can be formulated as

$$L = \frac{1}{2}\delta(-\mathbf{v}_e^{\mathsf{T}}\mathbf{v}_r)\|\mathbf{v}_e \times \mathbf{v}_r\|^2$$

where

$$\delta(u) = \begin{cases} 1 \text{ if } u \geq 0 \\ -1 \text{ if } u < 0 \end{cases} \quad \text{and} \quad \begin{cases} \mathbf{v}_e = \mathbf{x}_e - \mathbf{x}_1 \\ \mathbf{v}_r = \mathbf{x}_r - \mathbf{x}_1 \end{cases}$$

with $x_e$ being the coordinate of the end effector, $x_1$ the coordinate of the joint next to the end effector, and $x_r$ the target coordinate to point towards. The objective of optimization is to find a set of joint configurations that minimizes the objective function.

$$\mathbf{q}^* = \arg\min_{\mathbf{q}} [L(\mathbf{q})] = \arg\min_{\mathbf{q}} \left[\frac{1}{2}\delta(-\mathbf{v}_e^{\mathsf{T}}\mathbf{v}_r)\|\mathbf{v}_e \times \mathbf{v}_r\|^2\right]$$

The negative derivative $-\frac{\partial L}{\partial \mathbf{q}} = -\frac{\partial \mathbf{u}}{\partial \mathbf{q}}\frac{\partial L}{\partial \mathbf{u}}$ of the loss function with respect to the joint configuration **q** is the direction to update the joint configuration so that the claw manipulator end-effector vector points closer to the direction of the ring vector.

### 7.1.9 State Machine

The ROS SMACH library was used to control the high level towing sequence. The state variables from the rovers, Vive trackers, move_base planning, and entrapment detection are all synthesized and used to control the current actions of the robots. The final state machine is depicted in figure 22. Ultimately, a small number of states with a large quantity of internal logic were used. This allowed the same state to carry out variations of the same complicated behavior. In general, failures of the rescuing rover to dock with the stuck rover were handled by changing the goal behavior of the ”Drive” state, while failures of the rovers to free the stuck rover were handled by simply return.

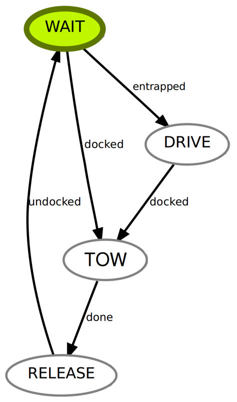


**Figure 22: The state machine controlling towing behavior.**

### 7.1.10 Networking Architecture

The HEBI actuators are controlled over Ethernet and required DHCP server, but the on-board computer (Odroid) only has one network card. So it is impossible for the Odroid to become both the client of the master router and a master router with the HEBI actuators as its clients. So we upgraded the networking architecture so that each rover are equipped with a on-board router that serves as a relay to connect to the master router of the entire network, as shown below in figure 23.

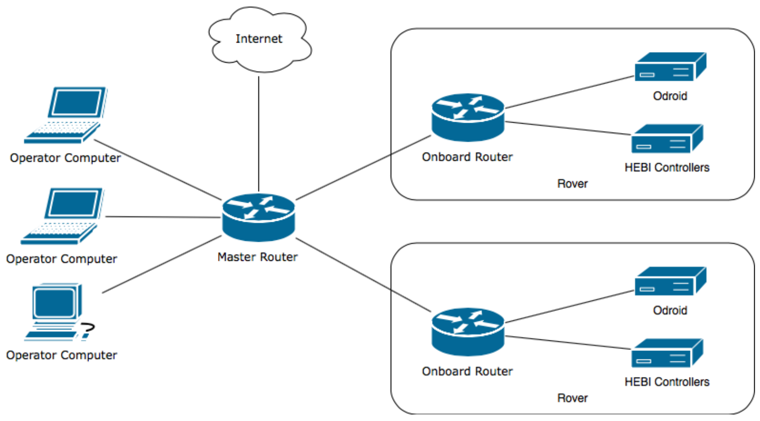


**Figure 23: The networking architecture of the system.**

An extra advantage of the above architecture is that we could actually get rid of the Wi-Fi dongles installed on the rovers in the previous networking architecture. These Wi-Fi dongles were not powerful enough for longer distance signal transmission and are easily interfered by environmental radio noise.

## 7.2 Subsystem Evaluation

### 7.2.1 Localization Performance Evaluation

The rover localization involves the HTC Vive Tracking system and the wheel odometers, which are filtered into an extended Kalman filter (EKF). The performance of a rover localization is evaluated by the overlay experiment as shown in figure 24 and a rover alignment experiment as shown in figure 25.

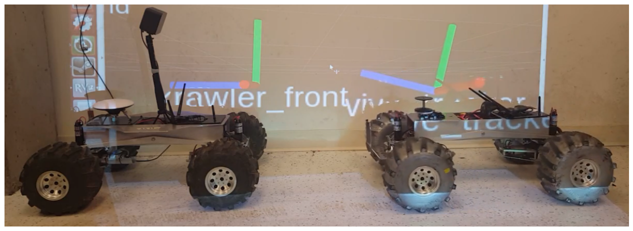


**Figure 24: Overlay of the estimated rover locations and the actual rover locations as one rover (right) approached another (left).**

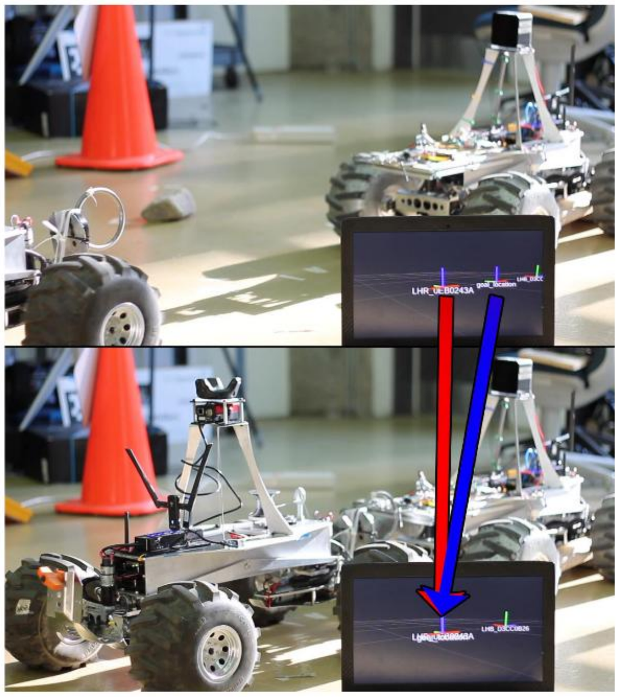


**Figure 25: Rover alignment and docking experiment.**

From the above rover localization performance experiments, it was observed that the odometry information from rover localization system, which can be used to estimate the pose of one robot with respect to another, reached an accuracy of $< 1$ cm. This experiment concludes the performance of the sensor suite and the state estimation subsystems jointly.

### 7.2.2 Entrapment Detection Evaluation

The requirement for the entrapment detection subsystem is that it shall detect an entrapment of a rover within $60$ sec from the time the entrapment actually happens. $6$ independent tests were conducted to evaluation the performance of the entrapment detection subsystem. In the experiment, the weight matrix for the entrapment detector is defined as

$$R = \begin{bmatrix} 0.95 & 0 \\ 0 & 0.05 \end{bmatrix}$$

The results of the tests are presented in figure 26.

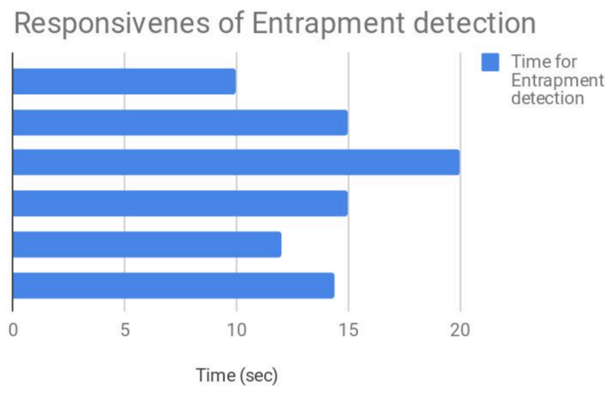


**Figure 26: Results of entrapment detection evaluation experiments.**

From the results of the entrapment detection subsystem performance evaluation experiments, it was observed that in all the trials the entrapment detection subsystem was able to detection entrapment in $< 60$ sec, yielding an average detection time of $14.33 \pm 3.09$ sec.

### 7.2.3 Minimum Turning Radius Test

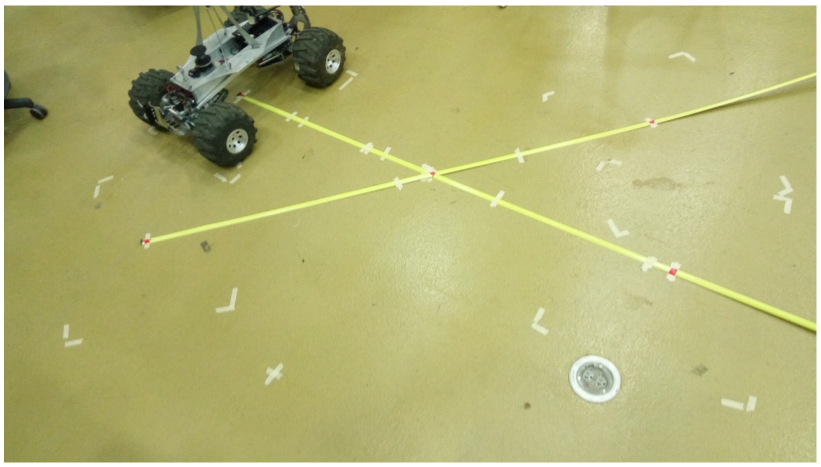


**Figure 27: Test setup for estimating the minimum turning radius.**

Experiments were conducted, as shown in figure 27, to determine the first cut-estimate of the minimum-turning radius of a 4x4 Ackermann driven AutoKrawler rover.

The observation suggested that the diameter of the circle $93 \pm 5$ cm corroborates with our initial estimate $(\approx 1)$ m based on the visual inspection of instantaneous center of rotation $I_C$ of the rover when fully steered. The results of the experiment are shown in the following table 1.

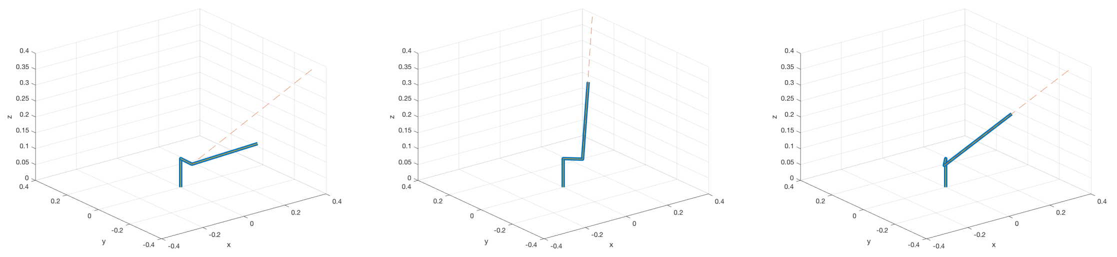
**Table 1: Results of the minimum turning radius tests.**

| Experiment# | Manual error estimate (inches) | Measurement (inches) |
|---|---|---|
| 1 | $\pm 2$ | 74 |
| 2 | $\pm 2$ | 77 |

The minimum turning radius of an AutoKrawler rover is $37 \pm 2$ in.

### 7.2.4 Autonomous Tracking Algorithm Evaluation

The autonomous tracking algorithm for the claw manipulator is evaluated using simulation experiments in *MatLab*. Two experiments were conducted to evaluate the performance of the autonomous tracking algorithm, as shown below in figure 28.



**(a)** $\mathbf{q_0} = \begin{bmatrix}0 & 0\end{bmatrix}^\mathsf{T}$ **(b)** $\mathbf{q_1} = \begin{bmatrix}0.8385 & 0.5724\end{bmatrix}^\mathsf{T}$ **(c)** $\mathbf{q_2} = \begin{bmatrix}-0.7323 & 0.5724\end{bmatrix}^\mathsf{T}$

**Figure 28: Simulation results for the claw manipulator autonomous tracking algorithm evaluation.**

Both experiments started with a joint configuration of $\mathbf{q_0} = \begin{bmatrix}0 & 0\end{bmatrix}^\mathsf{T}$ as shown in figure 28a. In the first test, the target (ring) was located at $\begin{bmatrix}1.0 & 1.0 & 1.0\end{bmatrix}^\mathsf{T}$, and after 100 optimization iterations the end-effector points towards the direction resembles that of the ring vector with the joint configuration converged at $\mathbf{q_1} = \begin{bmatrix}0.8385 & 0.5724\end{bmatrix}^\mathsf{T}$, as shown in figure 28b. Similarly, in the second test, the target (ring) was located at $\begin{bmatrix}1.0 & -1.0 & 1.0\end{bmatrix}^\mathsf{T}$, and after 100 optimization iterations the end-effector points towards the direction resembles that of the ring vector with the joint configuration converged at $\mathbf{q_2} = \begin{bmatrix}-0.7323 & 0.5724\end{bmatrix}^\mathsf{T}$, as shown in figure 28c.

## 7.3 SVE Performance Evaluation

**Objective** Demonstrate the autonomous rescue of a rover in an outdoor area with with numerous obstacles.

### Equipment

- Two Autokrawler platforms, with one equipped with a claw manipulator
- External HTC VIVE base station and tripod
- Two PS4 Joysticks for controlling the AutoKrawlers
- Operator laptops
- Prearranged and mapped obstacles and environment

### Procedure

1. Initialize both rovers at known locations in the test environment.
2. Tele-operate a rover over a high centering obstacle at least 2m away from the rescue rover.
3. High centered rover autonomously signals for assistance.
4. Rescue rover drives to and aligns with the high centered rover, while avoiding any potentially obstructing obstacles along the way.
5. Claw manipulator constantly orients towards the docking ring of the companion rover.
6. Claw triggers on the docking ring and rovers dock.
7. Rovers coordinate driving actions, freeing the entrapped (high centered) rover.
8. Rovers undock.

### Validation Criteria

1. The rescue sequence succeeds in 60% of trials.
2. The rescue rover does not get entrapped during the rescue procedure.
3. Automatic entrapment detection occurs within 60 seconds of entrapment.
4. Rescuing rover avoids obstacles greater than .3m in height.

Table 2: **Spring Validation Experiment (SVE) Results**

| Validation Criteria | SVE Results | SVE Encore Results |
|---|---|---|
| The rescue sequence succeeds in 60% of trials. | Rovers successfully freed entrapped rover in 60% (3 of 5) of initial trials. Success rate dropped to 50% (4 of 8) in later trials with more difficult scenarios. | Rovers successfully freed the entrapped rover in 90% of trials (9 of 10). The single failure occurred due to a miscalibrated velocity setting in the first trial. |
| The rescue rover does not get entrapped during the rescue procedure. | The rescue rover was never entrapped during rescue operations. | The rescue rover was never entrapped during rescue operations. |
| Automatic entrapment detection occurs within 60 seconds of entrapment. | Automatic entrapment detection was demonstrated 3 times, and detected entrapment within 15 seconds each time. | Automatic entrapment detection failed in 3 out of 3 attempts. |
| Rescuing rover avoids obstacles greater than .3m in height. | Rescuing rover was aware of the obstacle created by the other rover and avoided it. Full obstacle avoidance was not demonstrated. | Rescuing rover was aware of all obstacles in the test area, averaging .2m in height. Full obstacle avoidance was not demonstrated. |

## 7.4 Strong and Weak Points

### 7.4.1 Strengths

The main strengths of our project are listed as follows:

**Vive Trackers** The localization of the rovers in a given environment was very accurate within 6m by 6m space. The robustness and accuracy of localization module is attributed to the HTC Vive Tracker system. Within the region of laser sweeps of the HTC Vive base station, Vive trackers were able to give provide accurate 3D pose estimates of themselves.

**State Machine** Through exhaustive testing by carrying out multiple tests and catering for corner cases by adding redundant checks to the state transitions, the state machine turned out to be very stable and resulted in appropriate behavior consistently. The system was subjected to stress testing such as intentionally making the rescue rover miss the target through various means. In all these scenarios, the state machine ensured the rover behaved in the corrective manner by determining the failure of rescue operation and re-attempting the rescue operation autonomously without any manual intervention.

**Upgraded Steering and Drive Motors** Prior to changing the steering and drive motors to a higher torque motors, the rovers significantly lacked the ability to effectively navigate through a given trajectory. Upgrading the steering and drive motors boosted up the performance to the extent that even when single drive motors were operational, the rovers were still capable of navigating around the environment accurately.

### 7.4.2 Weaknesses

Amidst numerous strengths of the system, there were few weaknesses that had cost us time delays and additional effort from time to time. These weakness are described as follows:

**Linear Actuators** The linear actuator controlling the claw has no internal limit switches or current limits; meaning that there is nothing to stop its internal motor from stalling when the limits of its range are reached. Developing software controls that accounted for all the edge cases of limiting the actuator while moving it took a lot of time and many, many linear actuators were stalled out during development. In addition, the frictional and spring load on the actuator was much higher than anticipated and even the stronger actuators we started ordering were barely up to the task. Even when not stalling the motors, some of the gearboxes in the actuators broke.

**Wiring** The power distribution setup in the rovers had very little electrical protection built into its PCB, and the addition of multiple safety mechanisms and new electrical components was largely accomplished by adding more wires between things. This resulted in an extremely chaotic internal wiring space full of many loose and hard to follow connections. During testing in rough environments, multiple connections on the rover came free and seriously delayed testing.

The rovers were also highly sensitive to the order of connection of the Roboclaw motor drivers over USB; a mistake in the order just before SVE resulted in the destruction of one of the steering pinion gears.

**KillerKrawler Mechanical Design** The Autokrawlers are heavily modified KillerKrawlers, RC cars made by RC4WD. Most of the chassis of the original RC has been removed, but the axles are largely unmodified. The particular design of the KillerKrawler made replacing the steering motors (a frequent occurrence due to wiring and mechanical failures) exceptionally difficult and time intensive. In addition, the portion of the axles that hold the wheels on the rovers place a significant amount of stress on the threaded section of a bolt, creating high stress concentrations and making it very easy for the axles to break. We had four wheels fall off in the last week of the project.

# 8 Project Management

## 8.1 Schedule

Over the period of two semesters, the scope of project was improvised in order to specifically handle the case of dislodging an entrapped rover using a secondary rescue rover. While the resultant symbiotic roving system employed a claw and a hook mechanism, the team worked on several possible solutions, for example magnetic coupling, winching system etc., for this major technology development activity. The major activities undertaken and the progress at the end of the each of these work periods are summarized in the work breakdown structure. Figures 29 and 31 illustrate the project progress at the end of the fall and the spring semesters. In order to clarify the scope of development for each sub-system, work packages were used to track the progress of the project. Major tasks that defined each of the sub-system are captured in figure 33. The overall progress through the fall and spring semesters are captured in figure 30 and 32 respectively. As seen in figure 32 the steady progress over the last few months involved extensive refinement of the necessary autonomous behaviors that were a result of several sub-systems working in unison.

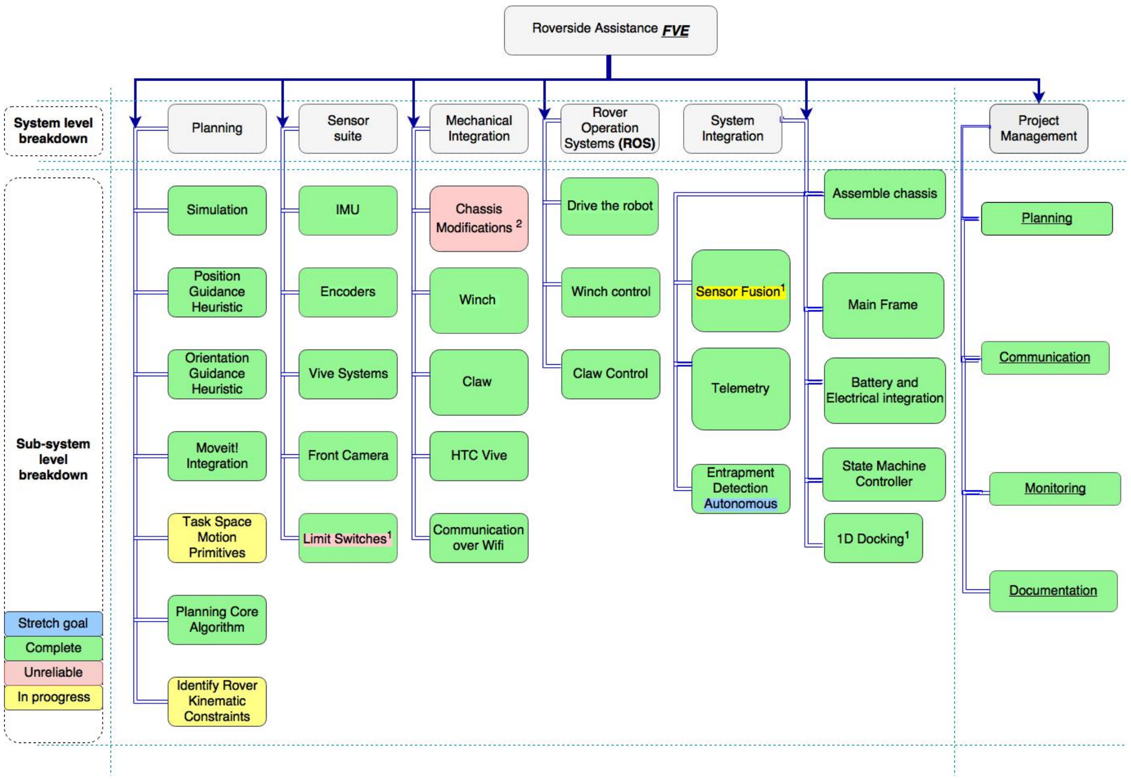
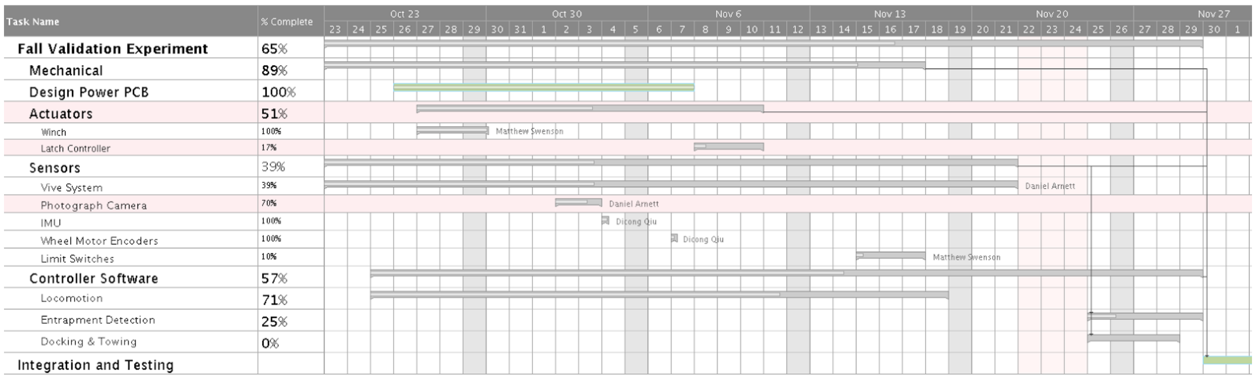


Figure 29: Work Breakdown structure at the end of Fall Validation period.



| Task Name | % Complete |
|---|---|
| Fall Validation Experiment | 65% |
| Mechanical | 89% |
| Design Power PCB | 100% |
| Actuators | 51% |
| Winch | 100% |
| Latch Controller | 17% |
| Sensors | 39% |
| Vive System | 39% |
| Photograph Camera | 70% |
| IMU | 100% |
| Wheel Motor Encoders | 100% |
| Limit Switches | 10% |
| Controller Software | 57% |
| Locomotion | 71% |
| Entrapment Detection | 25% |
| Docking & Towing | 0% |
| Integration and Testing | |

**Figure 30: Fall semester progress.**

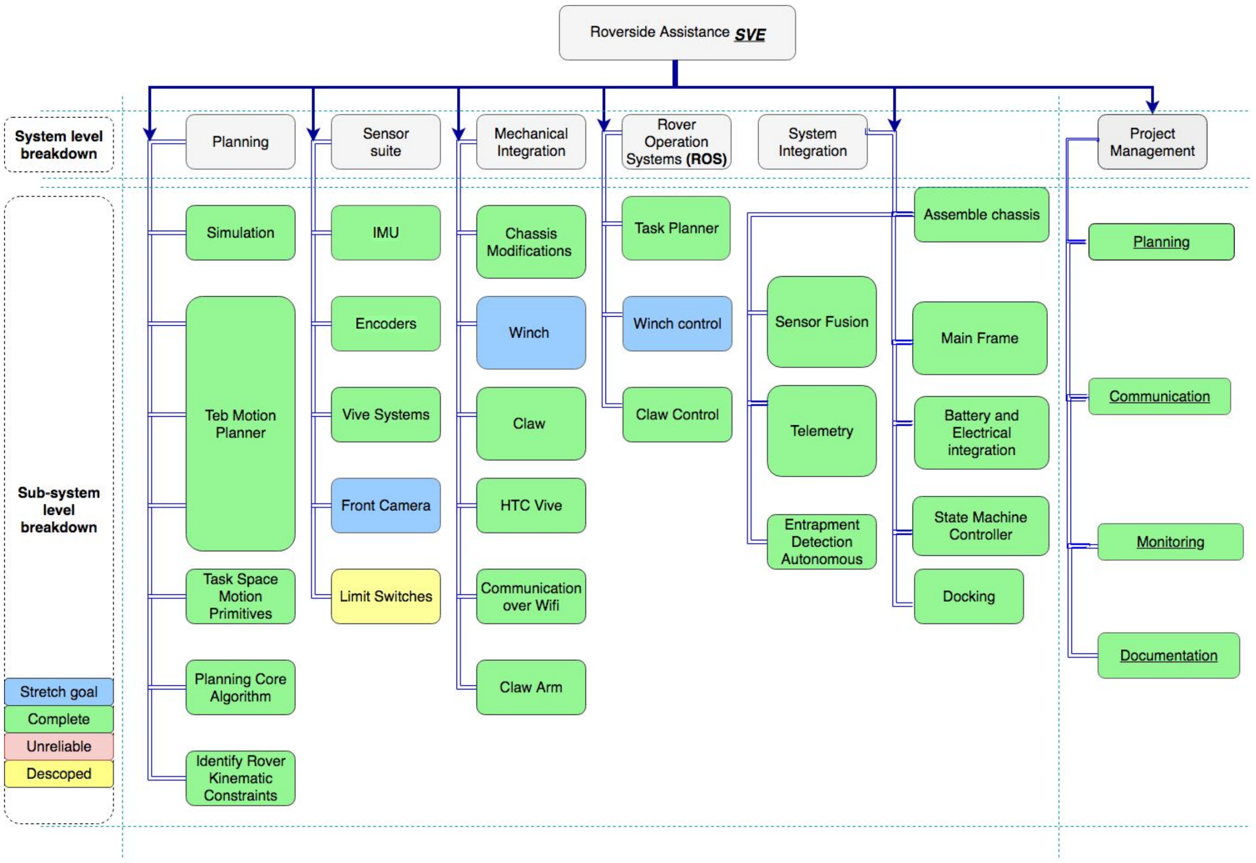


**Figure 31: Work Breakdown structure at the end of final Spring Validation period.**

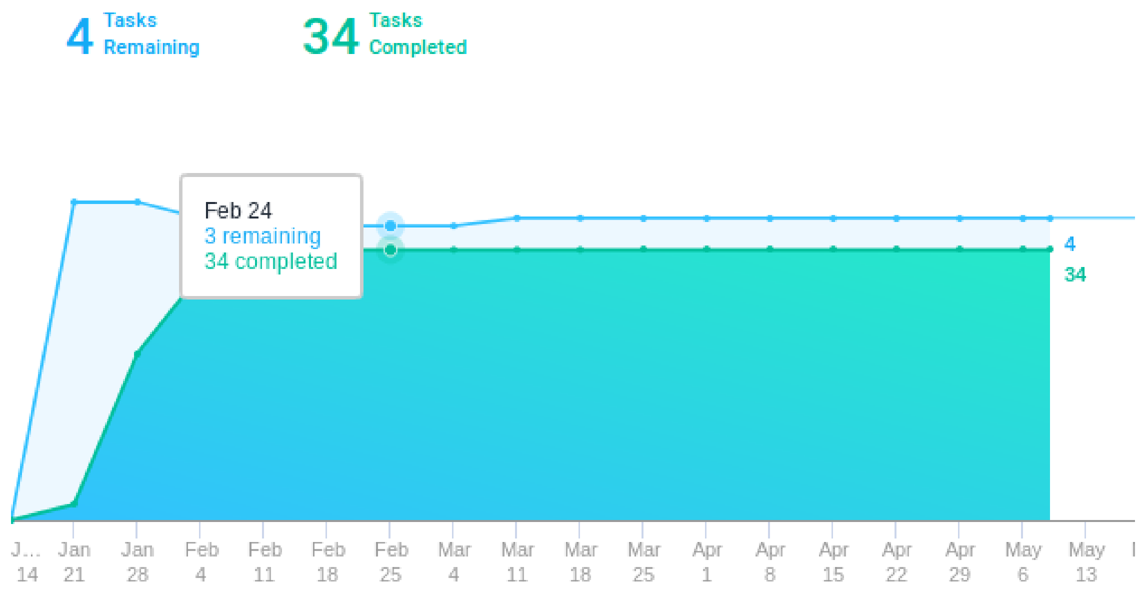
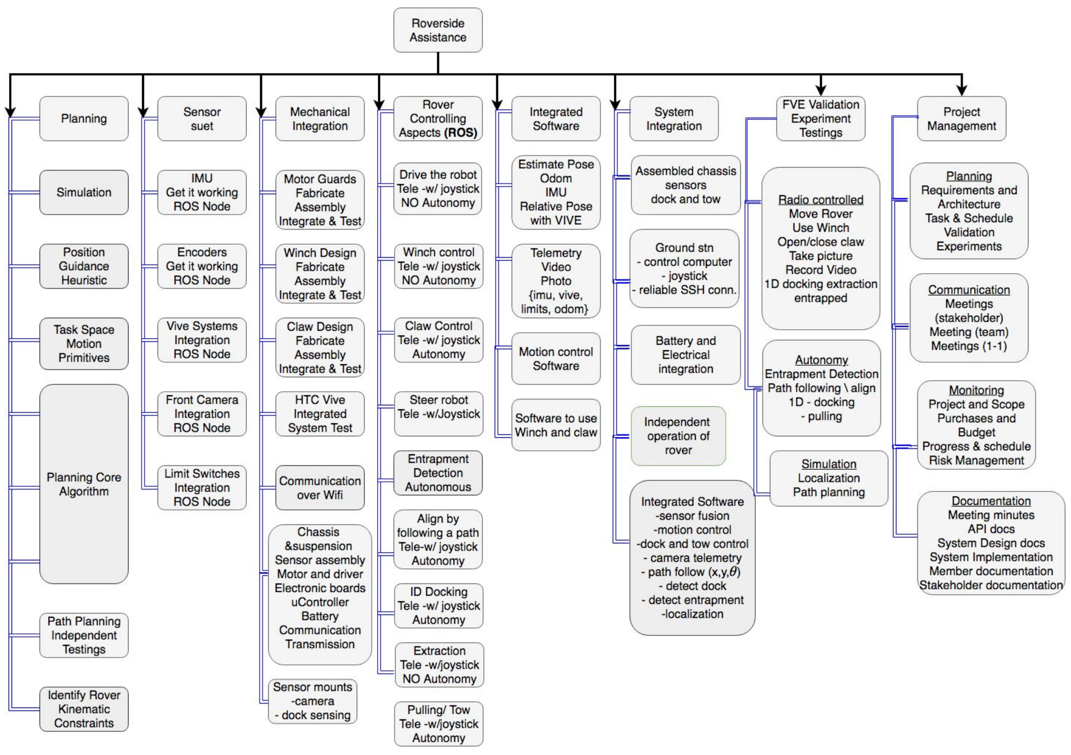


**Figure 32: Overall project progress during the Spring semester.**



Roverside Assistance
Planning
Simulation
Position Guidance Heuristic
Task Space Motion Primitives
Planning Core Algorithm
Path Planning Independent Testings
Identify Rover Kinematic Constraints
Sensor suet
IMU Get it working ROS Node
Encoders Get it working ROS Node
Vive Systems Integration ROS Node
Front Camera Integration ROS Node
Limit Switches Integration ROS Node
Mechanical Integration
Motor Guards Fabricate Assembly Integrate & Test
Winch Design Fabricate Assembly Integrate & Test
Claw Design Fabricate Assembly Integrate & Test
HTC Vive Integrated System Test
Communication over Wifi
Chassis &suspension Sensor assembly Motor and driver Electronic boards uController Battery Communication Transmission
Sensor mounts -camera - dock sensing
Rover Controlling Aspects (ROS)
Drive the robot Tele -w/ joystick NO Autonomy
Winch control Tele -w/ joystick NO Autonomy
Claw Control Tele -w/ joystick Autonomy
Steer robot Tele -w/Joystick
Entrapment Detection Autonomous
Align by following a path Tele-w/ joystick Autonomy
ID Docking Tele -w/ joystick Autonomy
Extraction Tele -w/joystick NO Autonomy
Pulling/ Tow Tele -w/joystick Autonomy
Integrated Software
Estimate Pose Odom IMU Relative Pose with VIVE
Telemetry Video Photo {imu, vive, limits, odom}
Motion control Software
Software to use Winch and claw
System Integration
Assembled chassis sensors dock and tow
Ground stn - control computer - joystick - reliable SSH conn.
Battery and Electrical integration
Independent operation of rover
Integrated Software -sensor fusion -motion control -dock and tow control - camera telemetry - path follow (x,y,θ) - detect dock - detect entrapment -localization
FVE Validation Experiment Testings
Radio controlled Move Rover Use Winch Open/close claw Take picture Record Video 1D docking extraction entrapped
Autonomy Entrapment Detection Path following \ align 1D - docking - pulling
Simulation Localization Path planning
Project Management
Planning Requirements and Architecture Task & Schedule Validation Experiments
Communication Meetings (stakeholder) Meeting (team) Meetings (1-1)
Monitoring Project and Scope Purchases and Budget Progress & schedule Risk Management
Documentation Meeting minutes API docs System Design docs System Implementation Member documentation Stakeholder documentation

**Figure 33: Major work packages.**

## 8.2 Budget

This project was supported by an NASA Research Initiative entitled *Symbiotic Exploration*. The total expenditure was about $11,440$ USD. In addition to the hardware and electronics for the robots, the project incurred substantial costs for spares and replacements due to the extreme testing that was necessary to validate the operation of the system in progressively rough terrain and also to support the unconventional hardware design for docking. Miscellaneous costs included necessary tools and electronics for developing the robots and also a one-time investment for components required for setting up the ground station and necessary debugging tools. Finally, in order to accommodate necessary safety measures and precaution when operating with multiple fire-hazard lithium-ion batteries, battery safe boxes for charging and circuit breakers were part of the project budget.

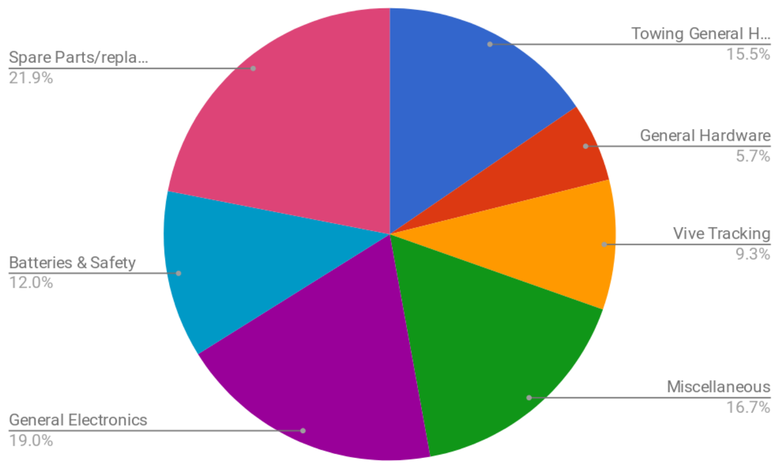


**Figure 34: Distribution of expenditure.**

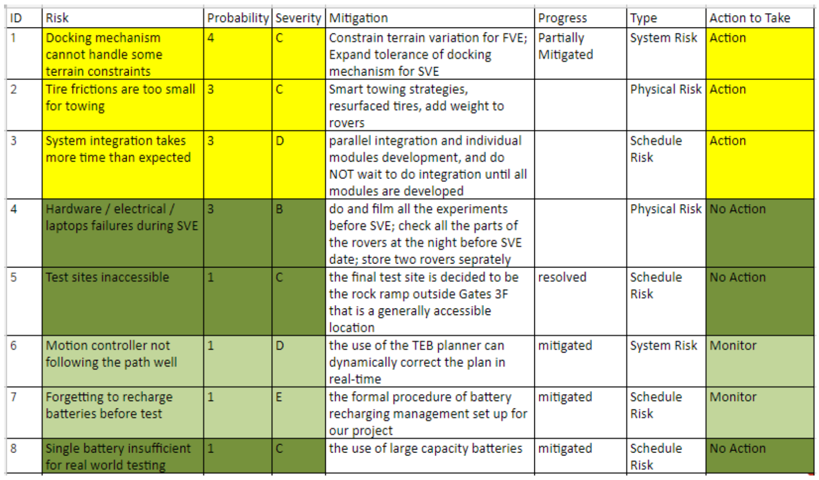
## 8.3 Risk Management

For the risk management, some risks have been mitigated or resolved, such as:

- Test sites inaccessible, which is resolved because the final test site is decided to be the rock ramp outside Gates 3F that is a generally accessible location;
- Motion controller not following the path well, which has been mitigated due to the use of the TEB planner that could dynamically correct the plan in real-time;
- Forgetting to recharge batteries before test, which has been mitigated since the formal procedure of battery recharging management set up for our project;
- Single battery insufficient for real world testing, which has been mitigated due to the use of large capacity batteries.

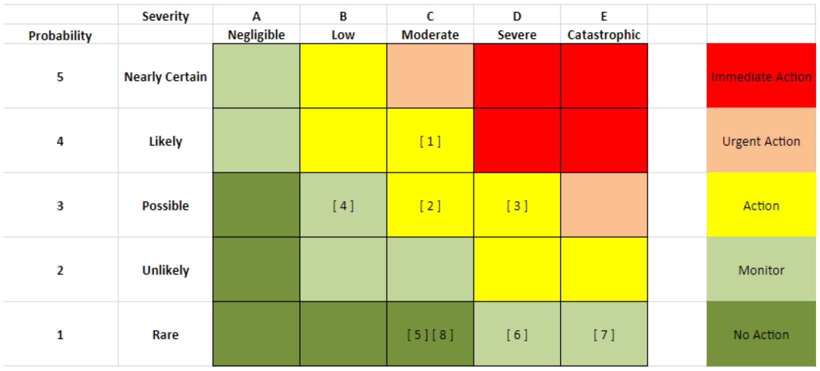
A summary of all risks is shown in figure 35, along with the figure 36 showing the risk matrix for them with the likelihood and consequence matrix for each risk.

For the unresolved risks, docking mechanism not able to handle some terrain constraints, tire frictions being too small for towing and system integration taking more time than expected are the three most outstanding ones. Hardware, electrical and/or laptops failures during SVE is the least severe risk.



| ID | Risk | Probability | Severity | Mitigation | Progress | Type | Action to Take |
|---|---|---|---|---|---|---|---|
| 1 | Docking mechanism cannot handle some terrain constraints | 4 | C | Constrain terrain variation for FVE; Expand tolerance of docking mechanism for SVE | Partially Mitigated | System Risk | Action |
| 2 | Tire frictions are too small for towing | 3 | C | Smart towing strategies, resurfaced tires, add weight to rovers | | Physical Risk | Action |
| 3 | System integration takes more time than expected | 3 | D | parallel integration and individual modules development, and do NOT wait to do integration until all modules are developed | | Schedule Risk | Action |
| 4 | Hardware / electrical / laptops failures during SVE | 3 | B | do and film all the experiments before SVE; check all the parts of the rovers at the night before SVE date; store two rovers seprately | | Physical Risk | No Action |
| 5 | Test sites inaccessible | 1 | C | the final test site is decided to be the rock ramp outside Gates 3F that is a generally accessible location | resolved | Schedule Risk | No Action |
| 6 | Motion controller not following the path well | 1 | D | the use of the TEB planner can dynamically correct the plan in real-time | mitigated | System Risk | Monitor |
| 7 | Forgetting to recharge batteries before test | 1 | E | the formal procedure of battery recharging management set up for our project | mitigated | Schedule Risk | Monitor |
| 8 | Single battery insufficient for real world testing | 1 | C | the use of large capacity batteries | mitigated | Schedule Risk | No Action |

**Figure 35: List of all risks.**



| | Severity | A | B | C | D | E | |
|---|---|---|---|---|---|---|---|
| Probability | | Negligible | Low | Moderate | Severe | Catastrophic | |
| 5 | Nearly Certain | | | | | | Immediate Action |
| 4 | Likely | | | [ 1 ] | | | Urgent Action |
| 3 | Possible | | [ 4 ] | [ 2 ] | [ 3 ] | | Action |
| 2 | Unlikely | | | | | | Monitor |
| 1 | Rare | | | [ 5 ] [ 8 ] | [ 6 ] | [ 7 ] | No Action |

**Figure 36: The risk matrix for the risks.**

### 8.3.1 Successful Risk Identifications

one of the remaining risk that has been identified was the hardware, electrical and laptops failures during SVE, which has actually happened during both SVE and the SVE encore. During the SVE and SVE encore, a hardware failure was that wheels fell off from a rover due to broken axles. Additionally, a linear actuator failed to function properly before the SVE encore, which severely obstructed the progress of the rehearsal. However, the risk was not marked as either a high likelihood risk or associated with high severity.

### 8.3.2 Fail Risk Identifications

Although the hardware failure risk has been identified, but it has been marked as a low likelihood risk associated with low severity, which did not turn out to be the situation. The hardware failure did happened and severely affected the demonstration of the SVE and obstructed the rehearsal of the SVE encore. Moreover, due to the hardware failure, the replacement of the hardware parts actually affected the performance of the entrapment detector, whose hyperparameters heavily depends on the hardware configurations. So during the SVE and the SVE encore, the entrapment detector did not perform in its best tunned status, which was a risk that has not been properly identified.

In addition, the risk of faulty wiring has not been properly identified, but actually happened, which leads to unexpected stopping of the motors. A pinion was also destroyed since a Roboclaw cable was plugged in backwards. And this issue has added to the delay of the project.

Another management risk that was ignored was the unreliable Internet/Git access from the AutoKrawlers. It has been a long existing issue, but the way we went around it was to trying copy and paste everytime any file was updated. As the software system became more complicated, appropriate version control was necessary. But the unreliable Internet access was a great obstruction that wasted a lot of our time.

# 9 Conclusions

Planetary rovers are capable of extricating one another when one is entrapped. When docked planetary rovers demonstrate increased traction and terrain traversability over when they are separated. Spinning, planar lasers provide accurate, high-speed relative localization measurements useful for docking. By adding the capacity for docking and towing, planetary rovers are capable of exploring steeper, more challenging terrain.

## 9.1 Lessons Learned

The most valuable resource for this project was time, so most of the lessons learned in this project relate to time management. The biggest time sink in this project was due to poorly installed electromechanical components. Motors repeatedly burned out, nuts and bolts repeatedly fell off. Much of this time would have been saved by specifying more reliable, higher torque motors and using locktight on all nuts and screws. The second time-saving strategy we developed was to thoroughly research every feature we needed in search for subsytems with this feature. ROS for example has software systems for many common robotics problems such as path planning and obstacle detection. When we leveraged those subsystems the project advanced quickly and when we didn't leverage those subsystems this project frequently fell behind schedule. Finally the most important lesson learned was to properly train all team members best safety practices and enforce them frequently. Risks such as battery safety had to be declared and known by all team members to preclude costly and time consuming accidents.

## 9.2 Future Work

Future work should incorporate reasoning about the modality of entrapment. The mode of entrapment impacts the likelihood of extrication, the risk to the recovery vehicle, and the means of extrication most likely to succeed. 3D modeling could be used for detection of obstacles and hazards as well as modeling of the entrapped rover to measure entrapment features like rocks on which it might be high centered, or sand it may have sunk into. 3D modeling could also be used to detect hazards such as high slopes or obstacles such as rocks or craters. Thermal imaging could be used for slip prediction [7]. All of these techniques have potential to add reliability and safeguarding to the rover extrication process, and could be used to better understand entrapment.

# References

[1] Wm C Mitchell, Allan Staniforth, and Ian Scott. Analysis of ackermann steering geometry. Technical report, SAE Technical Paper, 2006.

[2] Diederick C Niehorster, Li Li, and Markus Lappe. The accuracy and precision of position and orientation tracking in the htc vive virtual reality system for scientific research. *i-Perception*, 8(3):2041669517708205, 2017.

[3] Peter Corke, Dennis Strelow, and Sanjiv Singh. Omnidirectional visual odometry for a planetary rover. In *Intelligent Robots and Systems, 2004.(IROS 2004). Proceedings. 2004 IEEE/RSJ International Conference on*, volume 4, pages 4007–4012. IEEE, 2004.

[4] Sudin Dinesh, K Koteswara Rao, Manju Unnikrishnan, V Brinda, VR Lalithambika, and MV Dhekane. Improvements in visual odometry algorithm for planetary exploration rovers. In *Emerging Trends in Communication, Control, Signal Processing & Computing Applications (C2SPCA), 2013 International Conference on*, pages 1–6. IEEE, 2013.

[5] Dicong Qiu. Naive bayes entrapment detection for planetary rovers. *arXiv preprint arXiv:1801.10571*, 2018.

[6] introLab. RTAB-Map real time appearance based mapping.

[7] Christopher Cunningham. Improving prediction of traversability for planetary rovers using thermal imaging. 2017.