Survivor Identificaiton and Retrieval Robot

By

Zhijie Jin

Karun Koppula

Zachary Wasserman

Design Document for ECE 445, Senior Design, Spring 2018

TA: Xinrui Zhu

26 February 2018

Project No. 15

Contents

Introduction

Objective

After the Fukushima Daiichi didaster in after a 2011 tsunami, where robotics proved incapable of responding in disaster scenarios massive amounts of funding has been allocated to solving the problems that prevent robots from operating in unconstrained and unknown environments. Many teams developed different platforms which have since been used in disaster response situations [1] DARPA, specifically, has a Robotics challenge that requires robots to perform tasks that might be required in a disaster scenario that include interacting with objects, maneuvering in uneven and unknown terrain, and even driving a car. Unfortunately progress in the these directions have not made significant enough progress to be able to reliably use in disaster scenarios [2]. Disaster response robots of a different kind are currently being used, however, as documented and run by the Center for Robotic-Assisted Search and Rescue. These robots are for the most part human operated and are limited by human restrictions and the ability to communicate with the operator. Since these robots are not being used for heavy manipulation, their primary task is to identify humans in danger as well as dangerous areas. In disaster scenarios, it is often the case where finding survivors is a time critical operation given their possible medical states. Having an autonomous robotic platform that is onsite and can immediately begin exploration of the environment would increase the overall response time dramatically. It would reduce the time that it takes for first responders to arrive on the scene, set up their robotics systems, and establish safety parameters for operating in the environment. If an autonomous system can do a search and survey and present accurate information to the first responders, they can act immediately and potentially safe lives.

We envision an implementation of a robotic system that uses reinforcement learning to train how best to navigate in a maze environment while identifying the positions of survivors and performing a simple retrieval task. This robot will learn exploration behaviors so that when placed in a new test environment, it can explore the environment without without explicit retraining on the new environment. We hope to explore and overcome the obstacles of implementing a reinforcement algorithm on a real world platform, primarily the amount of noise and stochasticity in the system. We will train an asynchronous n-step, ϵ-greedy Q learning algorithm, with an Intrinsic Curiosity Module [3] to generate reward on a simplified version of the continuous 3-d state space. The Intrinsic Curiosity Module will help the platform to learn basic exploration techniques so that when it is placed in a new test environment, it is not trying to learn the behvaiors from scratch. Our hope is to extend the algorithm in future work to take in the image data as input to decide the action, using advanced machine learning algorithms. We will set artificial constraints on the robot in alignment with the real world maze environment that we set up for testing and training. The maze will be a fully connected space with high contrast intersections with differently textured walls. We will develop a differential drive robot with visual and depth sensors, that is given an occupancy grid of its environment. It will move through the maze using a set of pre-defined action primitives to unknown survivor positions and move a simple objects to a goal position. We would like to explore the extension of arbitrary starting positions if there is time at the end. It will identify and classify these objects using a Single Shot Detection algorithm [4]. Once it has identified a retrieval object, it will switch into a retrieval mode from exploration. Using a simple centering algorithm and a distance sensor, it will move to a position to pick up the object, after which it will return to the starting position, using information learned about the environment stage from the exploration phase. We will set environment rewards for the agent for these retrieval tasks, as well

as for identifying specific high contrast markers placed on the walls of the maze environment to represent areas of interest. Since we are limited in scope for this project, we wanted to develop a proof of concept that can be exteneded in future work to a robotic platform that can operate in real building environments.

Background

In recent years, work has begun on using machine and reinforcement learning techniques to develop robotic systems that can navigate through a simulated environment using visual and distance data, that eliminates the need for classical navigation tasks such as localization and trajectory planning. At the 2017 NIPS conference Pieter Abeel gave a keynote presentation about the use of meta-learning which is an algorithm that can learn a policy for a reinforcement learning task, referencing work done by Mnih [5] This method greatly cuts down on the number of training episodes needed to converge to an optimal behavior for each new environment. It allows the system to generate a general policy for exploration that doesnt overfit to a particular environment. This presentation inspired this project, but is beyond the necessary requirements for this project. With classical reinforcement learning, the training will allow the robot to function very well in a specific environment or on a specific task. Recent research expands this learned exploration behavior to create policies that can learn general exploration behavior in both high dimnesional state spaces and continuous action spaces with reasonably bounded training time. This learning resembles the inherent search behaviors that human beings exhibit without explicit training. Human beings learn this behavior over the course of a lifetime, using cues and methods that are not available to a robotic platform. The ability to replicate this behavior in an autonomous platform will come as a huge development in functionality. Basic AI maze solving algorithms are limited by preset heuristics and unintelligent searches. When the locations of interest are unknown in an environment that is also not entirely known or non-deterministic, the applicability and usefulness of these algorithms are reduced. We are leveraging the Intrinsic Curiosity Module reward function developed by Pathak [3] which learns to avoid maze walls and smoother exploration in a VizDoom environment while searching for objects of interest. Mnih develops a series of asynchronous policy learning algorithms that also learn behvaiors in a maze environment with specific environment rewards. These asynchronous methods provide large reductions in training times in high dimensional spaces. The actor-critic training algorithm is at the leading edge of progress in this direction. They have been used for manipulations tasks [6] as well as navigating 3-D mazes in simulation using only visual and depth measurements [7]. If a robot is trained on a particular building before a disaster, it will be readily equipped for the search on that building, but practically useless in another environment. We will develop a real world platform that can function reasonably in and learn to adapt to new environments using adapted versions of cutting-edge reinforcement learning algorithms.

High-Level Requirements

1. The robotic platform will learn an action policy in a simulated environment that will allow the robot to explore a maze-like environment.

2. When placed into a novel testing environment, the action-policy will retain exploration behaviors learned in the testing environment and reduce environment learning time, from naive exploration learning.

3. The robotic platform will correctly identify the survivor objects in all encounters, pick up and return it to the pre-specified goal position using information learned about the environment.

Design

Physical Design

The physical design of this robotic platform was chosen to minimize the dynamic constraints and associated challenges. Differential drive robots have a relatively simple set of dynamics and are inherently stable with the two unactuated wheels. Because of this, we decided to use a tiered approach to component distribution, which minimizes the physical footprint and allows for a greater safety margin while operating in the maze environment. Within our performance parameters, we do not expect to run into any situations that would make the robot uncontrollable. We have decided to place the actuator and environment sensors pointing forward in the centerline of the robot to align, as close as we can, the coordinate frames of each unit. This reduces the computational complexity, which is important, given the resources necessary for the sensor data processing.

Figure 1 shows a physical design. The motors and wheels will be chosen to suit a range of performance parameters. Because we do not know what the optimal responsiveness of this system in this environment is, we have chosen an arbitrary desired speed and acceleration at the high end of what we estimate to be the operable range. The two unlabelled circles in the Bottom View image are the two unactuated ball bearings that act as rolling supports for the robot. The gripper at the front of the platform will be actuated by two servos, one used to open and close the claw, and the other used to tilt the claw up and down in order to lift the goal object.

Figure 2 shows the tiering aspect of the design. This streamlines the connectivity of the platform by moving the Raspberry Pi closer to the sensors and out of the way of the major power connections between the motors and controller. Not included in the diagrams are possible wiring holes and access points that could be included to clean up the appearance of the platform as well as make wiring easier for the group. We will not permanently mount any of the components into the platform frame, to give us the freedom to interchange componenets.

Block Diagram

Figure 3 gives a layout of the different modules that are necessary for the robotic platform to function properly.

Control Block

The control block includes the Raspberry Pi and all of its included functionalities. This includes overall robot state control, object recognition, and RL exploration algorithms. The Raspberry will completely control the movements and thinking of the robot.

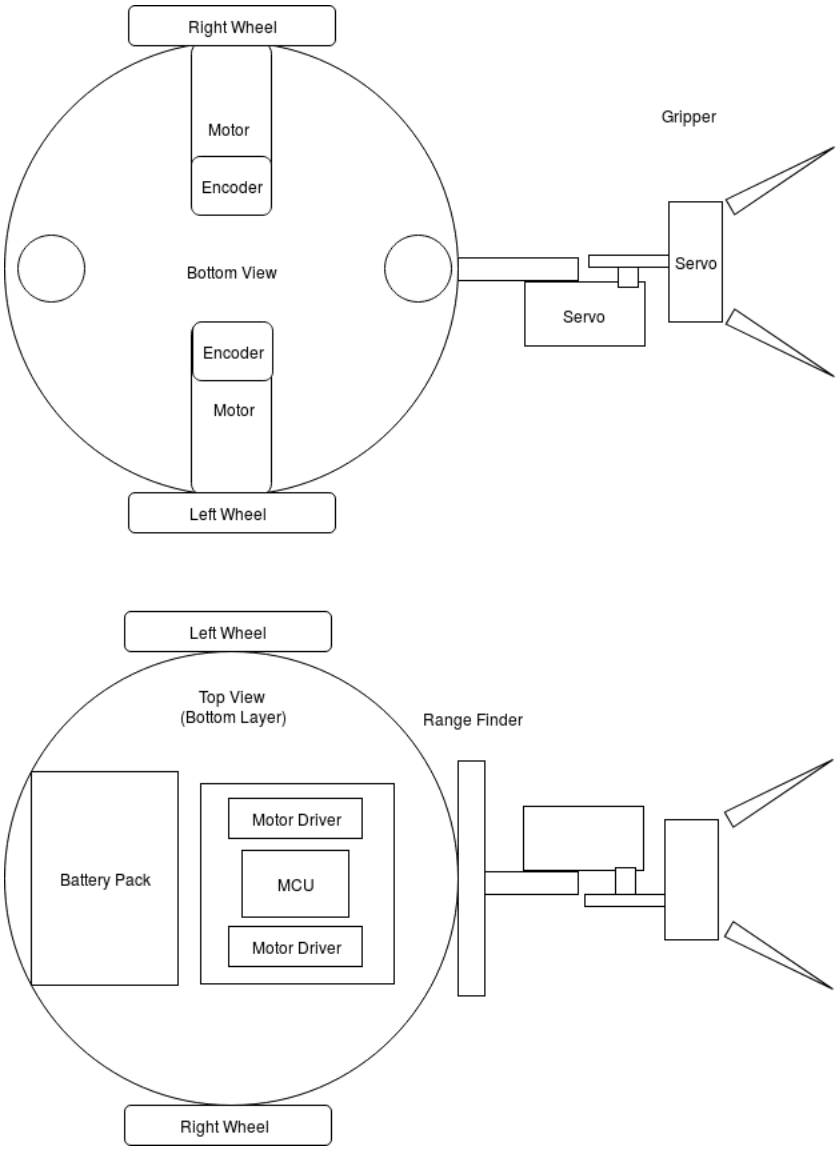

Figure 1: Physical layout of robotic platform.

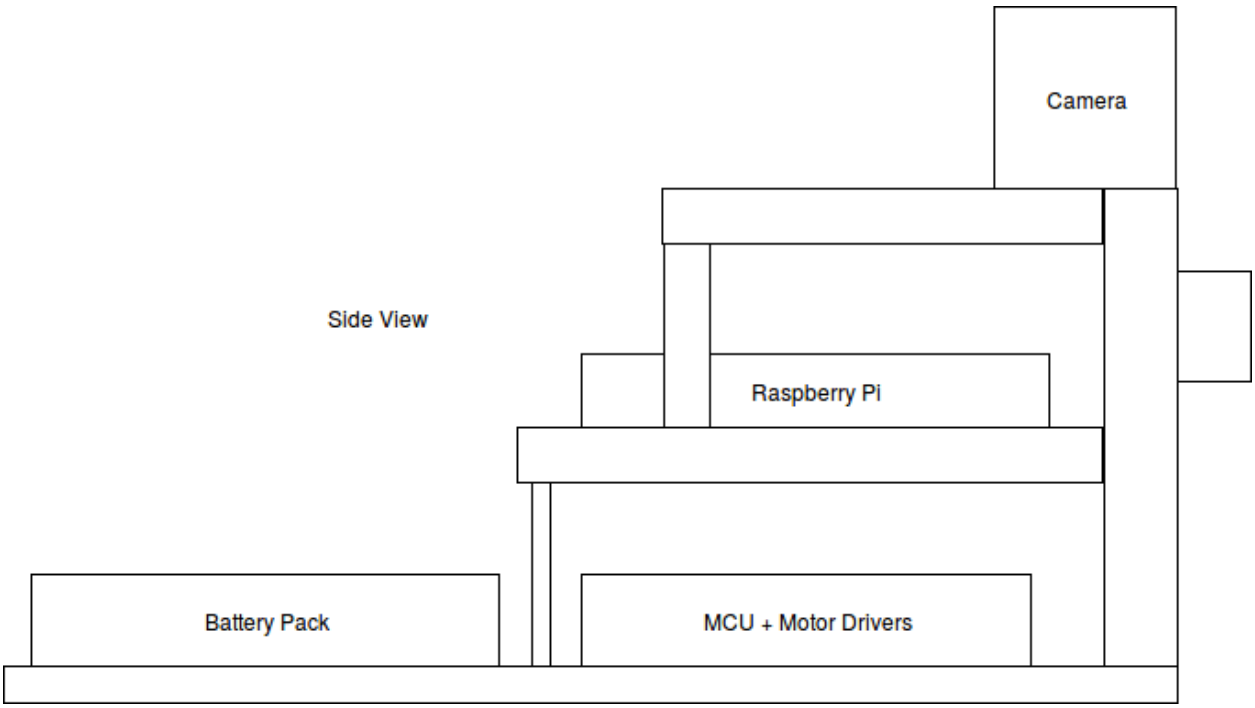

Figure 2: Side view to show tiered layout of components.

Table 1: Raspberry Pi Requirements and Verifications

Requirement	Verification
1. RL Exploration (a) Training time less than 24 hours with appropriately set episode length and environmental complexity (b) 10 Hz action calculation cycle	1. RL Exploration (a) Time training time (b) Use software timer to calculate output speed, or check ROS message publishing rate
2. PID Position Control (a) Accurate robot center of mass position control to within 0.2 cm for each action primitive (b) Rise time of 300 milliseconds, no overshoot, maximum 0.2 cm steady state error	2. PID Position Control (a) Measure physical translation with rule (b) Tune PID gains in simulation (Matlab) then apply to robot
Continued on next page	

Table 1 – continued from previous page

Requirement	Verification
3. Forward Kinematics Model (a) Must predict real-world robot center of mass translation to within 0.5 cm for each action primitive (b) Must predict real-world robot center of mass rotation to within 5 degrees for each action primitive	3. Forward Kinematics Model (a) Measure relative position with physical ruler (b) Measure relative orientation with compass, compare to orientation sensor output
4. Object Detection (a) 95% accurate object classification (b) Object centroid placement within 0.5 cm (c) 10 Hz cycle rate for detection calculation	4. Object Detection (a) Test algorithm on multiple instances of real varied objects and count accuracy, test on test database of images (b) Measure projected centroid position to projected centroid of object using naive image processing techniques (c) Use software timer to calculate output speed, or check ROS message publishing rate
5. State Controller (a) Appropriate control switch when object is to be picked up (b) Appropriate switch to exploitation mode for object return	5. State Controller (a) Check control actions after activation of detection signal (b) Monitor state variable for correct switch

Sensor Block

The sensor block contains all of the sensors that will be used to derive a probabilistic representation of the current state of the robot in the maze. We will be using a camera and rangefinder/depth sensors as the primary information sources. The image data will be used to identify features of the maze that can be correlated to previously knowledge of the environment. It will also be used to identify survivor markings, goal positions, and the object to be manipulated. The depth measurements provide help with localization and help to inform the decisions about driving the robot. The robot will also be equipped with pressure sensors about the body to detect collisions with the maze environment.

Image Sensor

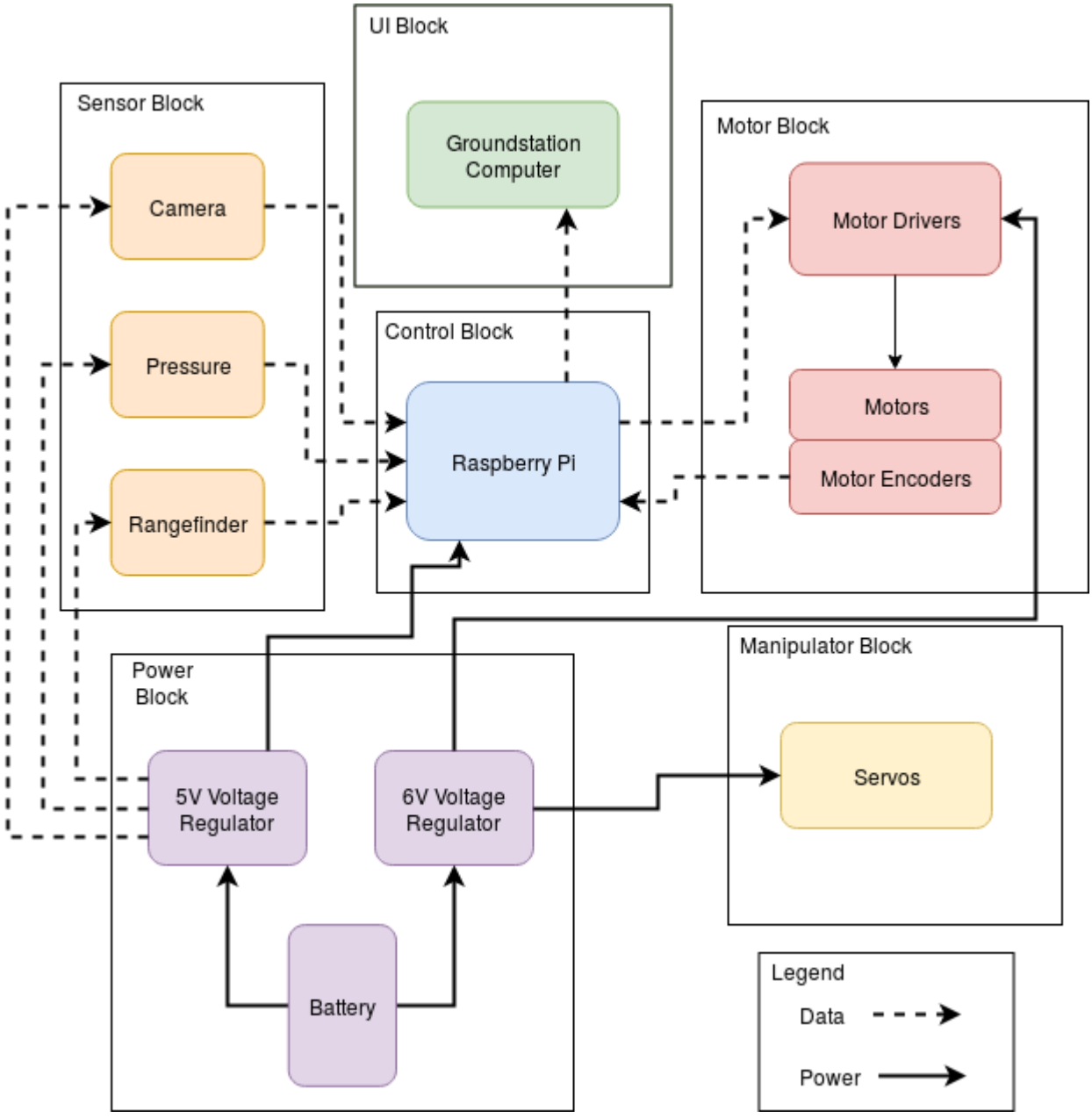

Figure 3: Robot Block Diagram with Legend

Table 2: Image Sensor Requirements and Verifications

Requirement	Verification
1. The camera must be able to communicate with the Raspberry Pi	1. Check for and display camera output using Raspberry Pi
2. Must provide RBG color images at least 10 Hz	2. Time ROS message publishing rate, display image output
3. Field of view greater than 30 degrees	3. Measure with 30 degree angle lines projected from camera frame of reference origin and check output to see if it includes lines

Distance Sensor

Table 3: Distance Sensor Requirements and Verifications

Requirement	Verification
1. The distance sensor must provide distance measurements up to 2m away with 3mm precision	1. Measure with physical ruler, distances to objects
2. The measurements must be sent with at least 10 Hz frequency	2. Check ROS publishing rate for distance measurement

Motor Block

The motor block consists of the motors themselves, along with the encoders and drivers. This is the entirety of the locomotive capabilities of the robot. It will be fed control signal from the control block, but will translate that information into the physical movement necessary to navigate the maze environment. The motor drivers must provide the correct voltage across the motors and protect them from overdrawing current and the encoders must provide the controller with accurate information about the position and speed of the motor.

Motors

Table 4: Motor Requirements and Verifications

Requirement	Verification
1. The motors must be able to drive the robot a $0.3m/s$ at $\pm 10\%$	1. Use linear velocity measurements to check speeds, use time and distance measurements to calculate approximate velocity
2. Must provide enough torque to move robot	2. Apply midrange PWM control signal to motors and check for movement

Encoders

Table 5: Motor Encoder Requirements and Verifications

Requirement	Verification
1. Encoders must provide $0.5° \pm 10\%$	1. Measure angular displacement using a compass for a specific input rotation of $0.5°$

Motor Driver

Table 6: Motor Driver Requirements and Verifications

Requirement	Verification
1. The motors must be able to be driven in both directions at equal speeds	1. Apply backwards control signal and measure wheel speed
2. Current draw must be limited to 2.4 A, stall current	2. Clamp motors and set control signal them while measuring input amperage with a multimeter, check for cutoff

Power Block

The power block consists of the battery and voltage regulators necessary to power all the components onboard the robot. The battery will be a 3S LiPo battery with 2.2Ah capacity.

Battery

Table 7: Battery Requirements and Verifications

Requirement	Verification
1. Must stay above 3V per cell during operation of the robot	1. Measure battery cell voltages constantly during operation using LiPo voltage alarm
2. Must provide power for continuous driving session of at least 15 minutes	2. Time usable battery time during real world maze exploration

Voltage Regulators

Table 8: Voltage Regulator Requirements and Verifications

Requirement	Verification
1. 6V Voltage Regulator (a) Must supply constant 6V ±5% output voltage while the battery discharges (b) Must be able to sustain currents up to 2.4 A	1. 6V Voltage Regulator (a) Measure voltage output of voltage regulator over the course of a fully battery operating discharge (b) Clamp motors and drive until motor driver shutdown, measure amperage with multimeter
2. 5V Voltage Regulator (a) Must supply constant 5V ±5% output voltage while the battery discharges (b) Must be able to sustain currents of 1.2 A	2. 5V Voltage Regulator (a) Measure voltage output of voltage regulator over the course of a fully battery operating discharge (b) Measure output current when all peripherals are tied to Raspberry Pi and running

Manipulator Block

The manipulator block consists of two servos that will be used to grasp and lift the goal object in the maze. Their actuation will be controlled by another block and their power is also provided by another block.

Table 9: Servo Requirements and Verification

Requirement	Verification
1. Grasping Servo (a) Must provide at least $1N \cdot m$ to grasp object to overcome gravitational pull	1. Grasping Servo (a) Grip object above ground level to make sure that it doesn't slip
2. Lifting Servo (a) Must provide at least $1.5N \cdot m$ to lift gripper, grasping servo, and object	2. Lifting Servo (a) Pick up objects with gripper assembly to test ability to lift goal objects

UI Block

In order to visualize the state of the robot and its decision making process, we have decided to send back state information that can be presented on a groundstation computer. This state and decision information can be presented alongside useful visuals such as optimal navigation decisions at each point of the maze for different objectives and overhead views of the maze environment.

Since this block only a groundstation computer, the block requirements are simplified to needing a Wi-Fi enabled computer than can recieve and display information from the Raspberry Pi and activate the robot's maze exploration over SSH protocol.

Software Design

We will use a ROS framework for our robotic platform running on the Respberry Pi. Through this, we can leverage the many libraries that are available for robotic control as well as the various python libraries that can be used. We will use the Tensorflow library to train and test all of our neural networks, for both exploration and object detection purposes. We will use OpenCV for image preprocessing and object detection verificaiton. We will use the OpenAI Gym environment with the Gazebo simulations to train the exploration. Sensor inputs will be sent to the Raspberry Pi. The two seperate algorithms, exploration and object detection, will be running simultaneously using different sets of inputs provided to the Raspberry Pi. We will implement a two-state system that switches into object retrieval mode when it recognizes an object. It will move towards and pick up the object by centering the object centroid in the image and using the distance sensor to approach it. The robot will then use information it has learned about the environment in the exploration phase to return the object to a goal state, by setting the goal state as the destination. After retrieval, it will either return to exploration with a limit on episode length or terminate function.

RL Exploration

We will used a multi-agent asynchronous update n-step Q learning algorithm to train the exploration algorithm on our test environments, it will take the vehicle position, orientation, relevant distance measures, and a subsampled subsection of the input image as the state parameters. We will use the ICM reward function in addition to specific environemental rewards such as penalizing proximity and rewarding finding the preplaced markers and object retrieval. We will train the algorithm using the OpenAI Gym Maze environment

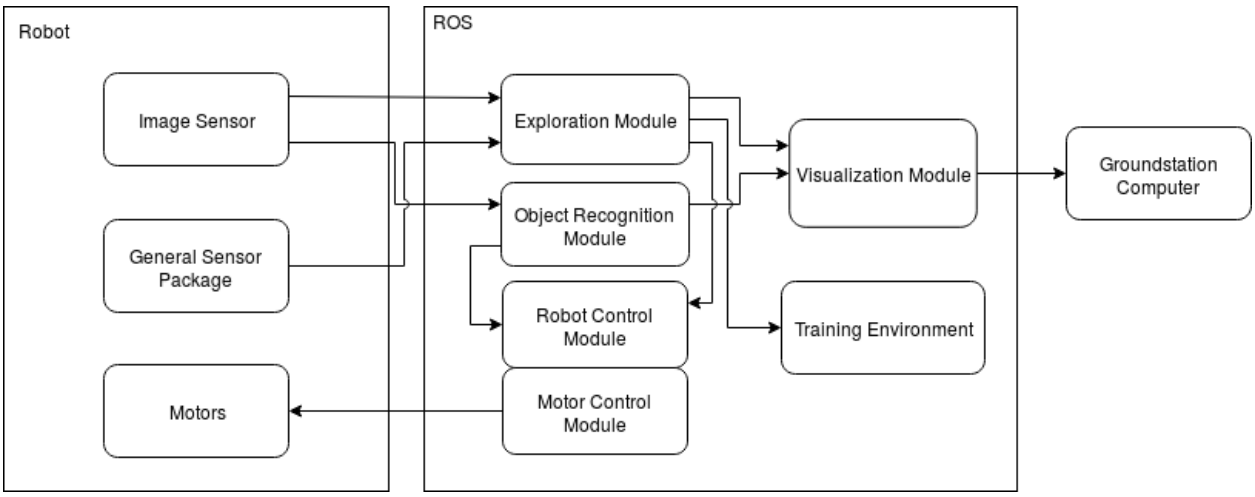

Figure 4: The overall system architecture.

with multiple concurrent agents exploring the state space with asynchronous updates on the model. The model will based on a set of action primitives, do nothing, move forward by an incremental amount, or turn in either direction by an incremental amount. We can use a simple forward kinematics model with each action with verification provided by the control system run on the MCU using the motor encoders. We will not have a global positioning system provided that can account for wheel slippages, but since we are constraining the environment we will assume that this will not occur in a great amount and that the other state values will provide a sufficient check on position error.

The state transition is further specified by Figure 6. We use a forward kinematics model of the robot to predict the next pose of the robot, given the action primitive chosen from the previous state. This model cannot account for possible slippages in the wheels, or environmental disturbances to the robot, because we dont have a valid measurement for global position. If we did, we could use Kalman Filtering to establish a converging estimate of the position and use that in each state. Since we expect the end-goal of this work to search in GPS-denied environments and without motion capture systems, we want to remove the dependence on learned search behaviors from exact position mappings as much as possible. We also want the learned behaviors to be portable to a new maze environment. The ICM research uses only image data in their simulated testing of the algorithm, but we expect that environment noise will necessitate the use of estimated state [3]. Fortunately with appropriate constraints on the environment, we can reduce slippages and disturbances to minimize this effect.

Object Recognition

To recognize the objects of interest, we will preprocess the images provided by the image sensor by truncating the image provided by the RPi camera to the region of interest and downsample (if appropriate) to a smaller size input image for the convolutional neural net. We will use a Single-Shot Detector algorithm to identify and locate the object in the image. Implementability is simplified with examples from the TensorFlow Object Detection zoo. We will need to train our net on images of our object, however, which will require an extensive, labeled data set of images of the object. We will choose the object based on available data sets or the difficulty of creating our own image set.

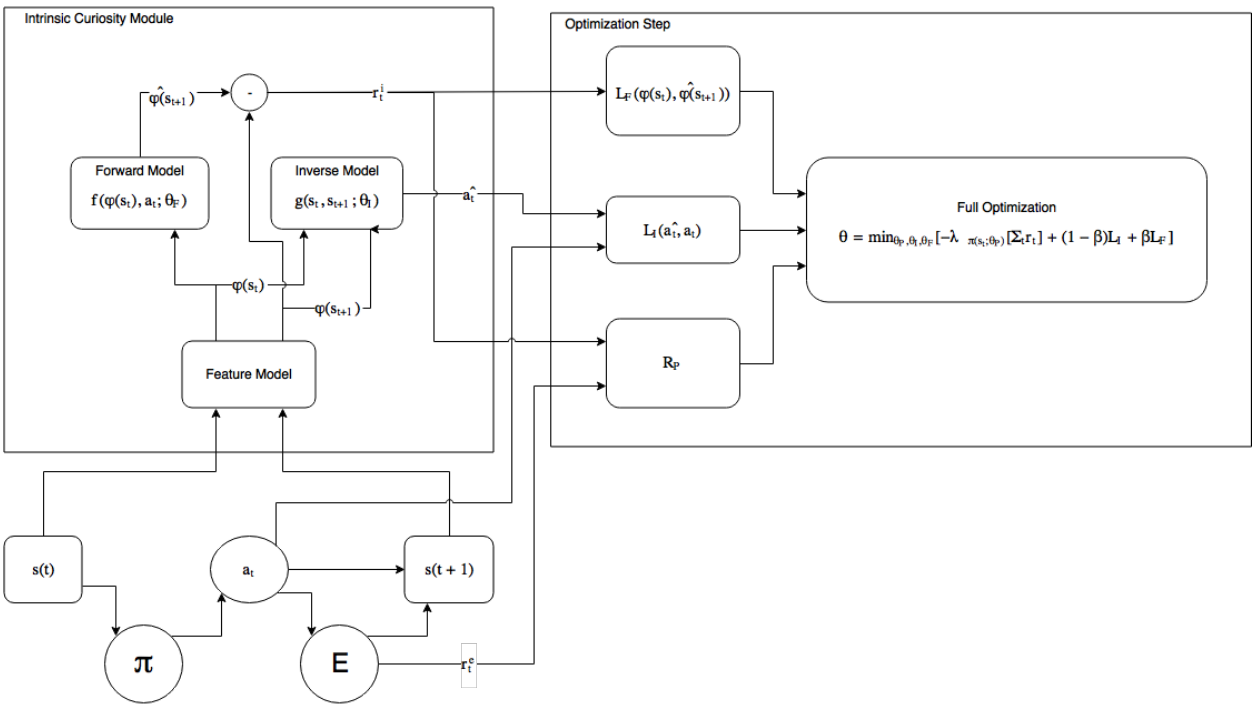

Figure 5: This flowchart shows how the reward is calculated using forward and inverse state transition models.

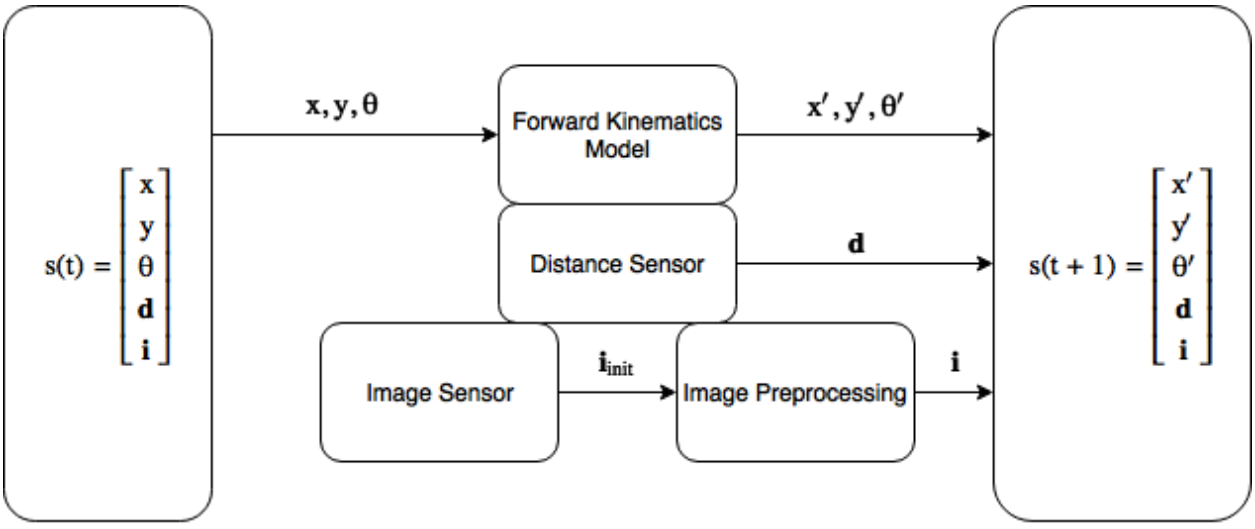

Figure 6: The state transition of the system, given some action primitive a_t

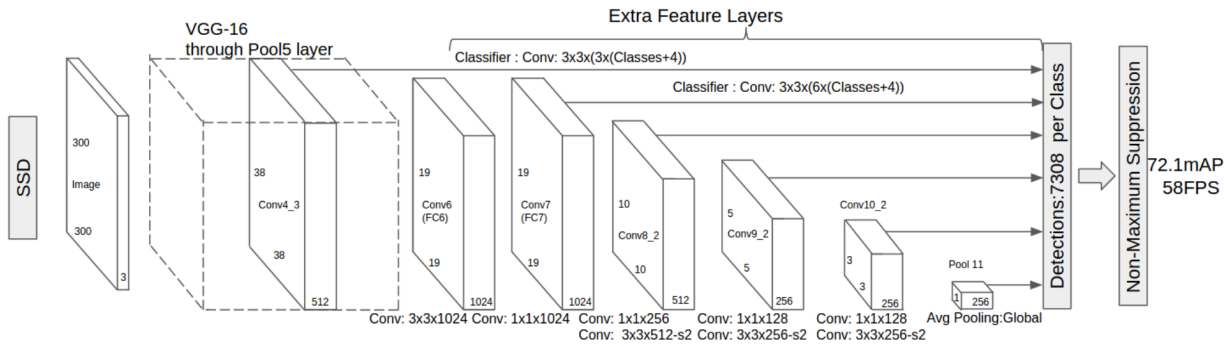

Figure 7: Overview of Single-Shot Detector Algorithm

Environment

Simulated

We will be using the OpenAI gym extension with Gazebo simulation to develop and visualize the maze environment. Unlike the image, our maze environments will not be loops, but will be fully connected with room-like features. We will also increase the wall contrasts and experiment with additional features and texturing on the walls to decrease correlation between states. We can create a simulated robot using the Gazebo software package that includes all of our real world peripherals. Since all of our functionality will be written in ROS using python, it is an ideal simulation and training environment.

Real

For testing and demonstration purposes, we will need to develop a real maze environment that mimics to a high degree the type of maze that we used in simulation. We will be using tape wrapped bricks to create a modular system for making mazes. We will color the faces of the bricks with colored paper to provide visual markings for the robot. The base will be made of plywood to provide a low-slip surface for driving on.

Tolerance Analysis

There exists a large amount of uncertainty in building the robot to navigate through a maze and retrieve objects. There are some negligible errors from resistors, sensors, and pcb board. For example, the range of error for the ultrasonic sensor is approximately around 2mm. If we were to try and find the distance of something about 5 meters away, that would mean that at worst we would have a .04% in our measurement. This is not significant enough to effect the distances at which the robot would observe walls or objects. However, there are also errors that we must to take into consideration when building the robot. Encoder and servo precision might be the main sources of error for out robot. Encoders that come with the motors might have low accuracy. Therefore, in order to minimize the error, encoders first need to be calibrated. After calibration, there errors still occur, we would put markers in the maze to help robot to locate itself. Based on the encoder data and markers in the maze, we can get more accurate position of the robot. Another big error issue may comes from servo precision and camera quality. Errors caused by servo accuracy and fishbowl effect may cause to grapper deviate from desired position. However, we are able to reduce error from servo and camera using some algorithms.

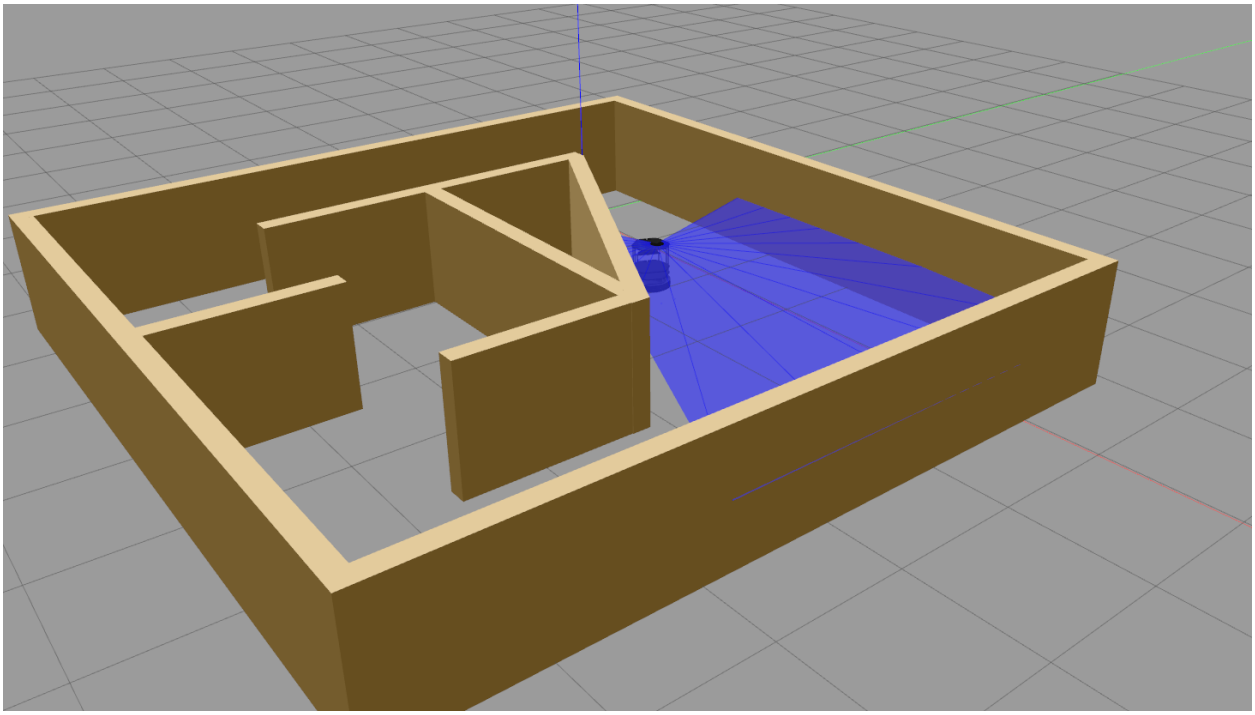

Figure 8: Simulated Maze Environment

Cost and Schedule

Cost

Labor

We calculate that our fixed labor costs over the course of the semester will be

$$3 \cdot 35\frac{\$}{hr} \cdot 12\frac{hr}{wk} \cdot 16wk \cdot 2.5 = \$50,400$$

Parts

Following is a starter table for parts costs. Add cell contents as well as rows and, if necessary, columns. Update the table number according to your sequence. Note that columns 1 and 2 are set up for centered text (words) and columns 3∼5 (numbers) are set up for right-alignment so that decimal points align.

Table 10: Parts Costs

Part	Manufacturer	Retail Cost ($)	Bulk Purchase Cost ($)	Actual Cost ($)
99:1 Metal Gearmotor	Pololu	34.95	31.46	69.9
Universal Mounting Hub	Pololu	6.95	6.12	6.95
11.1V LiPo Battery Pack	Turnigy	10.99	10.99	10.99
HS-422 Servo Motor	RobotShop	11.49	11.03	22.98
Large Robot Gripper	RobotShop	19.50	18.92	19.50
Raspberry Pi	Amazon	42.99	42.99	42.99
Raspberry Pi Camera	Amazon	26.44	26.44	26.44
Ultrasonic Sensor	EMakeFun	9.99	9.99	29.97
Motor Driver (DRV8835)	Texas Instruments	1.80	1.80	1.80
32 GB MicroSD Card	Amazon	13.88	13.88	13.88
Total				245.4

Schedule

Schedule			
Week	Karun Koppula	Zachary Wasserman	Zhijie Jin
2/12	Research project implementation ides	Begin looking into algorithms and simulations. Code partial of simulation	Research voltage regulators
2/19	Create algorithmic workflow for Design Document and write design doc	Prepare report and research OpenAI gym for robot simulation purposes	Research Battery Packs and voltage requirements for parts
2/26	Order Parts, Write n-step Q learning algorithm, and set up ROS framework on RPi	Prepare OpenAI gym simulation environment for robot, look into SDD implementation examples	Design PCB for voltage regulators and motor controllers
3/5	Write ICM and run training set	Write and train SSD algorithm on generic data set provide	Give parts and specifications to Machine Shop
3/12	Create simulated robot and environment, adapt ICM to robot specific state	Develop object data set and train SSD on image set	Develop Forward Kinematics model for action primitives
3/19	Spring Break	Spring Break	Spring Break
3/26	Continue to develop robot environment in simulation	Testing SDD object recognition on RPi using RPi camera	Robot Assembly, funcitonality verification, develop PID control for motors
4/2	Parallelize processing capability on the Pi, train full exploration algorithm in simulation and test performance in real maze, and maze assembly	Develop robot state controller, object pick-up procedure, and maze assembly	Object pick-up procedure and maze assembly
4/9	Test Robot exploration abilities experiment with hyperparameters	Develop and test robot return to goal with picked up object	Test and optimize robot state control
4/16	Experiment with performance, debug	Debug all systems	Debug all systems
4/23	Train robot for demonstration, write final paper	Write final paper	Write final paper
4/30	Finish final paper and demonstration	Finish final paper and demonstration	Finish final paper and demonstration

Ethics and Safety

Ethics

We believe that our project is aligned with the first tenet of the IEEE Code of Ethics, to hold paramount the safety, health, and welfare of the public, [2] because our project is designed to help move robotic understanding of real world systems towards the ability to save lives. We strive to use the understanding of intelligent systems to benefit the public good. This leads to the importance of #5 of the Code, to improve the understanding by individuals and society of the capabilities and societal implications of conventional and emerging technologies, including intelligent systems [2] in that it will be our duty to inform the public about the beneficial uses of the technology that we are working with and how they can be further used to help society. Since the success of the project is directly dependent on the functionality of the reinforcement learning algorithm, it is very important that we accurately report our results, regardless of the outcome. Inconsistent data and unreliable reporting would violate #3 of the Code [2] and would negatively impact the field of robotics research and our character as engineers. In the same vein, it is very important that we give appropriate credit for the previous works that we use and build on to develop our system. It would be unethical to take credit for the work of others in accordance with #7 of the Code [2]. We will be using and learning from many different research sources as well as from our peers and faculty members as we progress through this project and need to accurately present the chain of knowledge and development.

Safety

The major safety consideration of this project resides in the safe operation and storage of the battery. LiPo batteries can be dangerous if used improperly. They do generate heat during high-load discharge, which we must monitor throughout robot operation. We possess an industry approved LiPo charger and balancer for charging and discharging operations. We will, however, still be careful to not work alone while batteries are in operation. Since our group has little experience with building and designing circuits, we will have to be especially careful when designing and testing our custom printed PCB that contains the MCU, motor drivers, and voltage regulators. Voltage regulators can dissapate a lot of heat as well, so we must ensure that appropriate heat dissapation is provided to the circuits and other components. Short circuits, fires, and electricution are all possible safety hazards when working with these materials, so we will take standard lab safety precautions, as well as asking for input from the course staff and other experienced personnel.

References

[1] D. Nosowitz, "Meet japan's earthquake search-and-rescue robots," 2011. [Online]. Available: https://www.popsci.com/technology/article/2011-03/six-robots-could-shape-future-earthquake-search-and-rescue

[2] L. D'Monte, "5 disaster robots that may rescue you from natural disasters," 2015. [Online]. Available: http://www.govtech.com/em/safety/5-Robots-That-May-Rescue-You-From-Natural-Disasters.html

[3] D. Pathak, P. Agrawal, A. A. Efros, and TrevorDarrell, "Curiosity driven exploration by self-supervised prediction," in *International Conference on Machine Learning*, 2017.

[4] J. Xu, "Deep learning for object detection: A comprehensive overview," 2017. [Online]. Available: https://towardsdatascience.com/deep-learning-for-object-detection-a-comprehensive-review-73930816d8d9

[5] V. Mnih, A. P. Badia, M. Mirza, A. Graves, T. P. Lillicrap, T. Harley, D. Silver, and K. Kavukcuoglu, "Asynchronous methods for deep reinforcement learning," *CoRR*, vol. abs/1602.01783, 2016. [Online]. Available: http://arxiv.org/abs/1602.01783

[6] L. Pinto, M. Andrychowicz, P. Welinder, W. Zaremba, and P. Abbeel, "Asymmetric actor critic for image-based robot learning," *CoRR*, vol. abs/1710.06542, 2017. [Online]. Available: http://arxiv.org/abs/1710.06542

[7] P. Mirowski, R. Pascanu, F. Viola, H. Soyer, A. J. Ballard, A. Banino, M. Denil, R. Goroshin, L. Sifre, K. Kavukcuoglu, D. Kumaran, and R. Hadsell, "Learning to navigate in complex environments," *CoRR*, vol. abs/1611.03673, 2016. [Online]. Available: http://arxiv.org/abs/1611.03673