Carnegie Mellon University

Robotics Capstone Project

Requirement Specifications and Analysis

Friction Force Explorers:

Don Zheng

Neil Jassal

Yichu Jin

Rachel Holladay

supervised by

Dr. David Wettergreen

Version 2.2

December 14, 2016

Contents

List of Figures

1 Executive Summary

This report provides a requirements specification for our robotics capstone project. Through this document we aim to sufficiently specify the requirements for our project.

1.1 Project Overview

The goal of this project is to build a multi-agent system that collaboratively and efficiently recreates inputted images at variable scale. This system can be used for either reproducing works of art on a larger scale for aesthetic purposes or for marking elements of infrastructure. By using a team of robots that work together, as opposed to a single robotic system, we hope to gain greater efficiency as well as explore various coordination schemes.

1.2 Document Outline

Following this executive summary, we begin by outline the purpose of our project, including its goals (Sec. 2.1) and motivation (Sec. 2.3). We detail both our intended scope in this project (Sec. 2.4) and assumptions we make about our environment and operation (Sec. 2.5).

We then identify scenarios where our system could be deployed (Sec. 3). From these scenarios we can describe the various use cases of our system (Sec. 4).

This various scenarios informed the requirements of our system(Sec. 5). We begin by identifying our functional requirements (Sec. 5.1), which define the functionality of our system. Next we detail our non-functional requirements (Sec. 5.2), which provide us with testable performance benchmarks.

2 Project Description

We plan to build a multi-agent autonomous robotic drawing system. Our goals, motivation, definitions, scope and assumptions are provided below.

2.1 Product Goal

The goal of our project is to use robotics to bring people's ideas on paper into a reality. We aim to build a system that takes an image as input and reproduces that image on some surfaces, such as ground, poster, or pavement. This direct interaction with the physical world drives much of our design, as well as the need to use a dispensable writing tool, such as chalk or marker.

By using multiple robots, we hope to efficiently decompose a possibly large task into smaller, independent pieces which could be completed in parallel. In doing so we can also explore various coordination and scheduling strategies that naturally arise in a multi-robot scenario.

2.2 Definitions

System. The system includes all robots, user interface and any accessories, such as localization markers.
Robot. A single robot agent within the system.

2.3 Motivation

Traditionally robots have been pitched as excelling in the three D's: dull, dirty and dangerous. Our project primarily focuses on automating a dull task. The United States has nearly 47,000 miles of interstate highway and more than 5,000 airports with paved runways [1]. Each of these pieces of infrastructure in delineated and marked with drawn lines, which, when worn, must be repainted. These tasks are time consuming, expensive and, in many occasions, dirty, for example painting bicycle lanes would cost on average 133,170 dollars per mile [2].

By enabling robotic automation in painting these lines, we can improve the quality of our infrastructure while saving time and money in the long term. We can expand past transportation infrastructure to sporting arenas. Sports such as baseball, football or American football have fields with markings that must be regularly maintained to insure fair game play.

While this robot can serve a very functional purpose, it is not without its playful side. The robot can potentially be used as a children's toy for bringing the imagination of drawings to life.

While a fully operational system would ideally be able to cover all of the above motivational examples, we do not expect our system to. We discuss the expected scope of our project in Sec. 2.4.

2.4 Product Scope

While our system has a great deal of potential, we want to ensure that we can reasonably achieve testable goals. Therefore, in this section we will discuss robotic function and scenarios which are out of scope of this project.

We describe as system has being a multi-agent system. Based on our task, the number of robots in the system could scale up infinitely. Due to time, cost and planning constraints, we begin with a multi-agent system of two homogenous robots.

In our motivation and goal, we mention large scale applications such as airport runways and stadiums. However, for ease of testing we intend to primarily target this early version to smaller scale projects, such as drawing a design on a large, indoor poster.

Inherently, there is little that prevents our proposed project from scaling in this dimension aside from increasing the durability of our system. This level of durability, to functional well outdoors, is considering out of scope at this point.

In describing the functionality of our robot, we have purposefully not specified the exact type of writing implement. There are many possible tools including by not limited to chalk, marker, liquid chalk, etc. While we hope to explore several of these options, we do not anticipate being able to explore all options equally.

Delving further into writing implements there are also many smaller, interesting sub-points that we do not believe we will be able to build, such as drawing with multiple colors simultaneously or accounting for a large variety of sized writing tools.

2.5 Assumptions

In designing and parameterizing the needs of our system we will make the following assumptions about or environment and operation:

A1: We assume that our drawing surface is free of any obstacles.

A2: We assume that our system functions indoors, thus removing the need to handle various weather conditions.

A3: We assume that the robots are working on flat, homogenous surface. This disqualifies uneven or muddy ground, which is considered out of scope (Sec. 2.4).

A4: We assume that the writing implement being used by each robot in the system can be loaded into the robot manually by a human. Thus we do not expect our robots to auto-load writing tools.

A5: We assume that the writing implement can by used by making contact between the tip of the implement and ground, such as a pencil. This assumption removes using writing tools like spray paint.

3 Scenarios

The robotic drawing system has many use cases, three of which are detailed below. These scenarios then informed the use cases, described in Sec. 4. These are the scenarios that motivate this project and are the scenarios that a fully developed multi-agent autonomous robotic drawing system is expected to achieve. However, for the scope of this class, the focus on the chalk drawing scenario, as detailed in Sec. 3.1, in an indoor environment.

3.1 Chalk Drawing

Chalk drawings are often used around campuses and different communities for aesthetic purpose, information sharing, and events announcements, as seen in Fig.1. However, these drawings are often limited by size and complexity of the drawing. Therefore, the robotic system will be designed to draw large-scale items on blacktop or asphalt surfaces with chalks.

Since chalk will be the main painting tool in this scenario and these robots may work outdoors, we need to design the system to prepare for situations like drawing on relative wet surfaces and protecting chalk from rain or excessive humidity. We also need to design the system to accommodate the fact that chalk becomes shorter as it is used.

Figure 1: Various Examples of outdoor chalk drawing for pure fun or more serious announcements.

Figure 2: Airports rely on gridlines (left), which are currently painted and repainted by human operators (right).

3.2 Infrastructure Lane Drawing

Drawing infrastructure lanes, such as those for parking lots, highways, streets, and airports, can often be dull and expensive [3, 4]. For example, in Fig.2, the massive amount of routing lines that airport runway systems rely on. However, these lines are usually painted via a human operator in a repetitive and time-consuming manner. This leaves an opportunity to improve this process through a robotic system.

Since the drawn lanes must be smooth and consistent, motor control becomes critical in this scenario. Completing this scenario requires working outdoors. Therefore, the robotic system needs to be more robust both in hardware and in software. The mechanical parts need to be weather resistant to a certain extent and the programming scripts need to account for unpredictable situations when working outdoor.

3.3 Sport Lines

Drawing lanes for sports fields, including soccer fields, football fields, basketball court, etc, could be another scenario for this robotic system. Similar to the infrastructure example, these lines are quite repetitive and must be redrawn often to ensure a quality field. Sports lines are especially motivating as labor can constitute around 95% of their maintenance costs [5] [6], which a robotic drawing system could help reduce.

When painting certain sports fields, like soccer fields or football fields, these robots need to travel

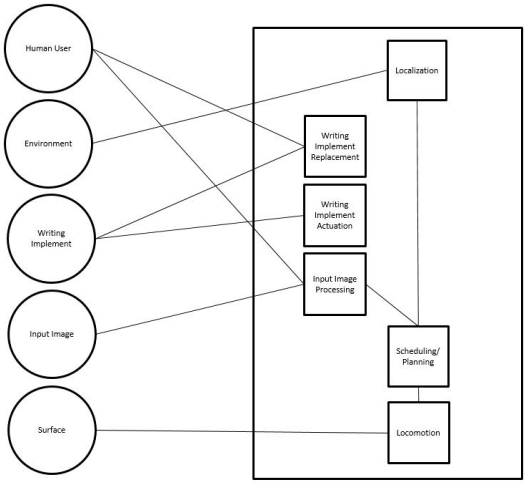

Figure 3: Diagram representing the relationship between use cases and actors in the system

on uneven surfaces, such as grass. Therefore, better suspension and drive system will be designed to complete these tasks. Similar to the infrastructure lane drawing scenario (Sec. 3.2), the robotic system needs to guarantee the quality of drawing lanes and being weather resistant.

4 Use Cases

In detailing the various scenarios that our system might encounter, described in Sec. 3, we developed a series of common uses cases that are necessary to the robot's functionality. These use cases are intended to describe the key functions of our robot and are further used to inform the requirements listed in Sec. 5.

4.1 Reload Writing Implement

Summary: A robot's main consumable, the writing implement, must be reloaded or replaced when empty or when a different implement (with different color or stroke) is desired.
Actors: Writing implement, human
Precondition: No writing tool in the system or the one that is currently loaded is no longer needed for the task.
Post-condition: System has desired writing tool loaded.
Alternative: If the robot is unable to load the implement then the system should report improper loading results to the user. Additionally, other robot agents should be told that the robot to be reloaded is broken, and therefore cannot draw. Hence, in order to complete the picture the other robots must re-plan accordingly.
Description: When the robot is working on a large or intricate drawing that requires a large amount of consumable writing material, the robot may not be able to carry enough material to complete its allocation of the drawing. Additionally, some drawings may involve a variety of colors, strokes, or other properties that necessitate the use of different writing materials. Thus, the robot must be able to replace its writing implement with human assistance. The robot must be able to

recognize reloading failures and alert the human operator of their occurrences. If the failure is not corrected, The robot must communicate its inability to perform to other robots in the system so they can re-plan drawing paths accordingly.

4.2 Process Input Image

Summary: The system must take in a human-provided image and interpret what the robot system is required to draw.
Actors: Image, Human
Precondition: No existing image being actively drawn by the robot system.
Post-condition: The image is processed and ready for work distribution between robots.
Alternative: If this step fails the robotic system should report an image processing failure.
Description: The system's task will be input using a human-produced image following specific guidelines. From the image, the system can determine where to place markings in the real world, and how the work should be distributed amongst the robot workers.

4.3 Localization

Summary: The robot must be able to determine where it is in the world.
Actors: Environment
Precondition: If the robot needs to localize, then there must be a large amount of uncertainty about current location and orientation of the robot.
Post-condition: Following localization we hope to have minimal uncertainty about the current location and orientation of the robot.
Alternative: If the robot cannot localize then this should be reported to human operator and the other robots should be alerted of the robot's inability to localize and orient itself. Additionally, any robot that cannot localize can potentially draw incorrectly or collide with another component of the system. Thus any robot that fails to localize should halt all movement.
Description: In order to create an accurate reproduction of the input image, the robot know how its location maps to a location on the input image. If it is unable to do so, it cannot continue drawing and must alert other robots to the fact so that they can re-plan and reschedule the workload.

4.4 Scheduling and Robot Planning/Coordination

Summary: The robot workers must determine an efficient allocation of the work.
Actors: Robots, Input image
Precondition: No plan or schedule currently exists.
Post-condition: Each robot has an allocation of work and a planned path.
Alternative: If we fail the plan, the human user is alerted and the operation is aborted.
Description: The work required must be determined from the input image, and analyzed to determine an efficient allocation of work, as well as a path schedule for each of the robots. Additionally, robots must communicate work completed and failure states to each other in case re-planning is

required.

4.5 Move Robot

Summary: The robot moves across flat terrain.
Actors: Ground
Precondition: Robot is stationary.
Post-condition: Robot is moving as commanded.
Alternative: If the robot is unable to move, the the human user is alerted and work is redistributed between remaining robots.
Summary: The robot moves across the writing surface, using its writing implement when required.

4.6 Use Writing Implement

Summary: The robot creates a mark on the writing surface.
Actors: Writing surface, Writing implement
Precondition: The robot has a plan to draw an mark, but that mark has not currently be drawn yet.
Post-condition: The robot's planned mark is drawn on the writing surface.
Alternative: If the robot cannot write, the robot alerts user that the writing implement must be replaced (or inserted if one does not exist).
Summary: The robot creates markings on the surface with the given writing implement, and ensures that its movements do not disrupt drawing accuracy.

5 Requirements

We outline our system's requirements, both functional and nonfunctional. A priority number is provided for each requirement. The system requirements are prioritized on a Likert scale from 1 to 7, detailed below. To describe the scale, a 1 is considered a desirable but unnecessary requirement, a 4 is considered a necessary requirement but open to significant changes, and a 7 is considered an uncompromisable requirement for a minimum viable product.

	Lowest (1)	Low	Medium-Low	Neutral (4)	Medium-High	High	Highest (7)
Priority	☐	☐	☐	☐	☐	☐	☐

5.1 Functional Requirements

FR1: Omnidirectional Movement Priority 7

· Robots must be able to move instantaneously in any commanded direction on a 2D plane represented by the floor of motion. Omnidirectional movement is necessary to ensure agents can adequately and efficiently cover the drawing workspace, while simultaneously being able to draw lines of varying

curvature.

FR2: Autonomous Priority 6

· Given an input image to draw, robot agents must be able to autonomously complete the drawing. This includes the following steps: processing the input, planning and commanding individual agents, and having robots move and draw without external input. We allow for human intervention only in the case of system failure.

FR3: Robots Localize Globally and Locally Priority 7

· All drawing robots can determine their locations and orientations on a global and local scale. Global scale is relative to the localization markers and the specified drawing surface. Local scale is relative to other robot agents. This requirement is necessary so the robot system can coordinate planning together to avoid collisions or an inefficient spread of work. Robot localization should be accurate to within one inch - that is, each robot agent should know its position to within 1 inch of its true location and within 10 degrees of its orientation.

FR4: Within Bounds Priority 3

· While in operation, all robot agents must stay within bounds of the workspace. This is important to minimize external collisions, including possibly unsafe interactions with the world, as well as to ensure localization maintains accuracy while drawing occurs.

FR5: Insert Writing Tools Priority 2

· A writing implement must be able to be inserted by a user into one robot under 1 minute. New writing implements will need to be inserted whenever existing ones become unusable, or upon first-time setup.

FR6: Remove Writing Tools Priority 2

· .A writing implement must be able to be removed from a single robot by a user under 1 minute. Writing tools will need to be removed when the existing implement becomes unusable.

FR7: Replace Writing Tools Priority 2

· A user must be able to remove and then insert a writing implement. Following in accordance with Functional Requirements FR5 and FR6, this process must be done in under 2 minutes. Tool replacement is necessary whenever an existing tool becomes unusable for system operation.

FR8: Reliable Communication Priority 7

· Robot agents and any system controllers must have the consistent ability to communicate between each other. This is necessary to ensure accurate and timely planning, and status updates between agents. Connection strength must be such that send information to each other within 1 second.

FR9: Drive Control System Priority 5

· Robot agents must have a controller to ensure motions made are accurate. Accurate motions is paramount to ensure an accurate drawing. The drive control system must be in compliance with motion accuracy parameters, as specified in functional requirement, FR**??**.

FR10: Turn on or off writing tool Priority 1

· All robot agents need to be able to enable or disable use of the writing implement. The robot must also have the ability to move regardless of the state of the writing implement. Not all drawings are contiguous lines, and as such the robots must be able to disable the writing tool to move to a new drawing location.

FR11: Input Drawing Plan Priority 6

· The main system controller must be able to receive an input that allows it to command the robot agents to draw an appropriate image. The system must be able to parse the input into a state usable for robot planning and control in order to draw the input.

FR12: Robots Know Progress Priority 2

· All robot agents are required to understand how much of the drawing each one has completed, and which sections are left to be drawn. This is necessary to ensure an equal spread of workload across all robot agents. The ability to communicate progress can be guaranteed by consistent inter-robot communication, as defined by FR8.

FR13: Kill Switch Priority 3

· Human bystanders must be able to end all system operation instantaneously with a kill switch or power button. This is necessary to ensure that the system can be shut down in case of an unsafe error or problem.

FR14: User Interface to robot Priority 1

· An intuitive and useful user experience is necessary for efficient usage of the system. Having a simple to operate system also reduces the likelihood of user error during the input or operation stage. For industrial or commercial applications, accessibility becomes important as well. A quality user interface can be tested by running a user study or survey with potential users.

FR15: Battery Power Priority 4

· Individual robot agents must be able to run continuously for a minimum of 30 minutes before needing battery charging or replacement. This will provide enough time to complete a minimum of a single drawing.

5.2 Non-Functional Requirements

NFR1: Documentation Priority 3

· Documentation of the design process, software, and hardware implementation is important for debugging, recreation, and general understanding of this project.

NFR2: Error Handling Priority 6

· The system must be able to handle errors appropriately. This includes problems that arise from localization, locomotion, planning, or using the writing tool. The system must be able to determine, based on conditions of the failure, whether to halt all operation or to continue without using broken subsystems. Errors can occur inside of any of the use cases listed in Sec. 4.

NFR3: Weight Restriction Priority 4

· Individual robot agent weight must be under 50 pounds. This is one step in ensuring portability of the individual agents.

NFR4: Size Restriction Priority 5

· Individual robot agent size must be able to fit within a standard doorway. This is defined as 80 in. x 36 in. [7].

NFR5: Efficiency Priority 2

· Robot system must be efficient and complete drawing tasks quickly. Timeliness can be measured by the amount of time taken to complete drawings. Robot planning must also exist in such a manner that evenly splits the workload among all robot agents working in the system. Therefore, a speedup of a maximum of 2x the rate is expected when using two robots as opposed to one. The system should perform as fast or faster than a human attempting the same drawing task.

NFR6: Quality Priority 4

· The drawing that results from the robotic system must closely match the input image. Quality can be qualitatively measured by visual comparison, or using software via a number of image difference metrics.

NFR7: Mobile App Priority 1

· User interface and experience will be done using a mobile application. This application will allow users to remotely input images for drawing, enable, disable, and pause the robot system's operation. The app will also be able to track and display current progress.

NFR8: Reliability Priority 6

· Robot system must be robust, and be resilient to breaking down or failing to operate properly. Reliability can be measured by percent uptime relative to total time spent in use.

NFR9: Coordination Priority 4

· The key to a multi-robot agent system is to reduce the individual workload, meaning having coordinated and efficient work is vital. Robots must be able to work together to minimize overlap in the drawings, and to avoid duplicating work. A two-robot system that coordinates must perform faster than a single robot performing the same task.

NFR10: Budget Priority 7

· Design and implementation of this robotic system is limited by budget, which must be strictly adhered to. The budget is limited $2500.

NFR11: Safe Priority 4

· Robots must maintain safety with respect to each other and the external world at all times. This requires that all robot agents avoid collisions. Robot agents must also have an enforced maximum speed limit to avoid damage to themselves or human bystanders in case of collision.

NFR12: Positional Accuracy Priority 7

· Robots must be able to move with a positional accuracy of a 1 inch radius for every 3 feet of motion. Tight positional accuracy is vital to ensuring the robots can accurately complete the drawing.

NFR13: Rotational Accuracy Priority 7

- Robots must be able to turn with a rotational accuracy of within 10 degrees per 90 degrees turned. Similar to Sec. 13, rotational accuracy is necessary for the robots to accurately complete the drawing.

NFR14: Tool Switching Duration Priority 2

- Individual robots must be able to enable or disable the writing tool within 10 seconds.

References

[1] "Building america's future.." [Online; accessed 6 October 2016].

[2] M. A. Bushell, B. W. Poole, C. V. Zegeer, and D. A. Rodriguez, "Costs for pedestrian and bicyclist infrastructure improvements: A resource for researchers, engineers, planners, and the general public," 2013.

[3] D. Speidel *et al.*, "Airfield marking handbook," 2008.

[4] "Salaryexpert: Line painting machine operator highways salary in united states." [Online; accessed 16 October 2016].

[5] S. Stewart and A. V. Dore, "Sports field line marking alternatives," 2008.

[6] FieldTurf, "10 year cost analysis - fieldturf vs. natural grass," 2008.

[7] "Home depot: How to choose exterior doors." [Online; accessed 16 October 2016].

Carnegie Mellon University

Robotics Capstone Project

Concept Design

Friction Force Explorers:

Don Zheng
Neil Jassal
Yichu Jin
Rachel Holladay

supervised by
Dr. David Wettergreen

Version 2.0
December 14, 2016

Contents

List of Figures

1 System Description

The goal of our project is to develop a multi-agent robotic drawing system. Using a team of homogenous robots, we can efficiently parallelize the dull task of drawing. Our system is motivated by the miles of paint needed to maintain airport runways and sports arenas, as well as the widespread use of sidewalk chalk. Below, in Fig.1, we model our overall system graphically, with a more detailed description following. We then enumerate each subsystem in Sec. 3.

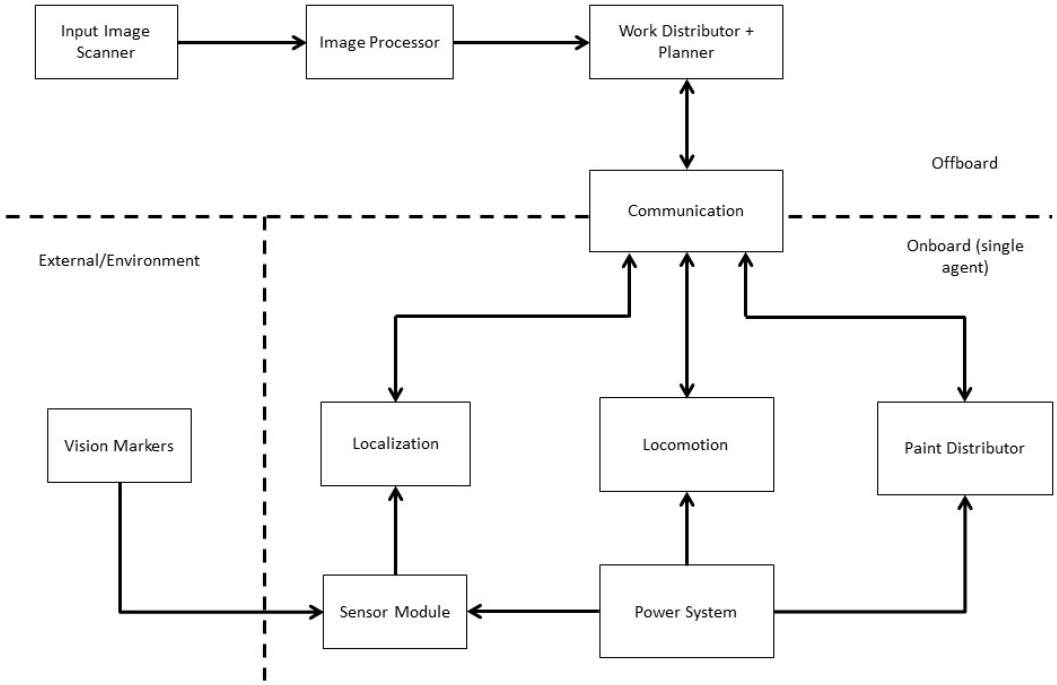

Figure 1: High Level Architecture Diagram

Our system encapsulates three sections, shown in Fig.1 through dotted lines. First, there is a single offboard unit that handles the coordination, planning and user interface. The initial drawing process begins in the offboard unit, where users will generate an input image . Image processing will then occur, where the input is processed into a format compatible by the planner and work distribution subsystem Sec. 8. Moving forward, the planner module will take in updates to current robot progress and use it to update commands it sends to robot agents.

The second section are the robot agents, marked in Fig.1 as "onboard". While the diagram shows one onboard agent, this schema is identical for each agent. Commands are sent to these agents. Communication protocols are able to send information both to and from the offboard work planner to robot agents (Sec. 9). From here, onboard systems are separated into three subsystems: Localization (Sec. 6), Locomotion (Sec. 5), and Paint Distribution (Sec. 4). The paint distributor is involved with all commands regarding engaging and disengaging the writing implement to output paint onto the writing surface. Locomotion is the drive system, which engages wheels and motors used for movement. This subsystem also includes any motion-related safety mechanisms, such as quickly stopping. The Localization subsystem works to determine the position and orientation of an individual robot agent.

Finally our third component comes from environmental setup, which is mainly the situational landmark tools need to localize the robot.

2 Concept Operation

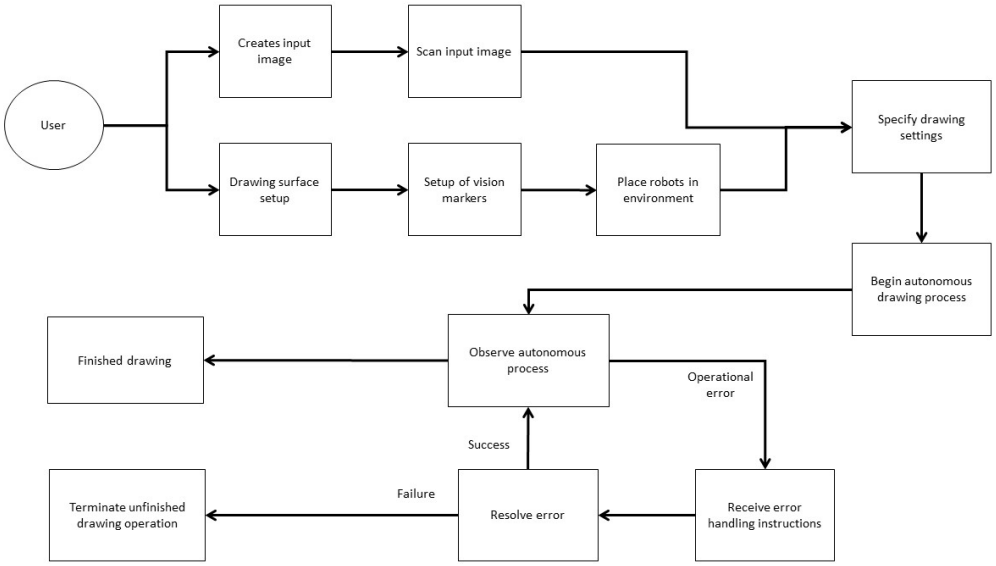

Figure 2: State Machine of Concept of Operations

Above, in Fig.2 we outline the user experience with the system. This is detailed below and our user interface is fully explained in Sec. 10.

The initial setup contains two operations that can be performed simultaneously: adding the image to be drawn to the system, and setup of the drawing surface. Adding the input image involves generating the image, and then scanning it into the system. This satisfies requirements for input the drawing plan (Requirements Specification Sec. 5.1, FR 11), and a user interface (Requirements Specification Sec. 5.1, FR 14). Setup of the system involves placing the drawing surface, placing and calibrating vision markers, and finally placing the robot agents within the bounds of the drawing surface. Once both steps are done, the user can enter any required settings for their use, and begin operation. Processing of the input image is done automatically by the system and is invisible to the user (Sec. 7).

Once the autonomous drawing process begins, the user simply observes the robots until they complete the task. Errors will be reported to the user, who then has the option of fixing the issue to continue operation, or terminating the drawing process. The autonomous process satisfies FR 2.

3 Subsystem Listing

The remainder of this document is split based on the various subsystems. We briefly describe each subsystem here. The remainder of this document is split based on the various subsystems. We briefly describe each subsystem here.

Writing Implement (Sec. 4): The mechanism that uses the writing tool to deposit material onto the writing surface.
Locomotion (Sec. 5): This system involves robot motion across the writing surface.
Localization (Sec. 6): All pieces of the robot system involved with determining position and orientation of the robot agents within the drawing space.
Image Processing (Sec. 7): Takes a user-created image and processes into a format usable by the work scheduler (Sec. 8).
Work Scheduling, Distribution, and Planning (Sec. 8): Determines trajectories for individual robots to complete the drawing, based on the user's input image.
Communication (Sec. 9): This subsystem involves all communication between robot agents and the offboard system.
User Interface (Sec. 10): A unified system for user input and interaction into the system.
Power System (Sec. 11): Supplies power to all necessary parts of the robot system.

4 Writing Implement

The writing implement mechanism deposits writing material onto the working surfaces. It manages reloading of the writing implement and ensures that the implement is properly secured. This subsystem can also draw discontinuous strokes and draw in various widths. In the event of writing material depletion, its sensors will detect the occurrence and alert the communication module (Sec. 9).

The writing implement system is modular, so that various writing tools (such as chalk, markers, pens, etc.) can be easily inserted by users. This involves a fixed motor mechanism on the robot, with mounts for each tool that can be locked into place on the robot. The mounts can optionally connect to two motors: One for linear motion to raise and lower the writing implement, and another to rotate the writing material as described above.

Critical Components: Writing implement housing, actuation mechanism, writing material levels sensor, and reloading mechanism.

4.1 Use Cases

4.1.1 Load/Switch Writing Implement

Description: The robot has the ability to allow quick swapping of writing implements. This may be necessary when the current implement has been depleted, or a new implement with different writing properties is desired. The writing implement actuator itself is modular and detachable, so different kinds of writing implements (markers, chalk, etc.) can be used in the same robot.

4.1.2 Actuate Writing Implement

Description: The robot can use its writing implement to make markings on the working surface. It can vary drawing properties such as line continuity and stroke thickness.

4.2 Trade Study

To ensure the robot is suitable for a wide range of drawing applications, we consider three different drawing materials: chalk, markers, and paint. For chalk drawings, the robot can either utilize a traditional chalk or a liquid chalk marker. Markers are usually applied through markers, for example Sharpie Permanent Markers. Typical ways to apply paint is either using a roller or a brush or using spray paint can. The mentioned drawing materials will be evaluated against system requirements in the context below.

As non-functional requirements NFR 6 and NFR 8 indicate, the robotic system needs to reliably paint drawings that closely match the input image. Compared to chalks and markers, methods for applying paint is often more complex, which makes it difficult to ensure stroke quality. Also, drawing with paint often involve dripping and uneven coatings. Such drawbacks will further defeat our requirement on drawing quality (NFR 6). Therefore, we plan to pursue chalk and markers for drawing materials.

There are two ways of applying chalk drawings: using traditional chalk and using a liquid chalk marker. These two methods are evaluated against criteria including mechanism complexity and drawing quality. Since traditional chalk shortens and flattens while drawing, the designed mechanism needs to account for this potential change in length and provides an extra rotational degree of freedom to unify line width. On the other hand, the mechanism for liquid chalk markers does not need to account for the marker length and requires only one translational degree of freedom. Hence, in terms of mechanism complexity, a liquid chalk marker is better than traditional chalk. However, in terms of drawing quality, both methods are uncertain. Therefore, a prototype will be used to reveal the difference in drawing quality between the two methods and to help us determine which one to use.

4.3 Artistic Sketch

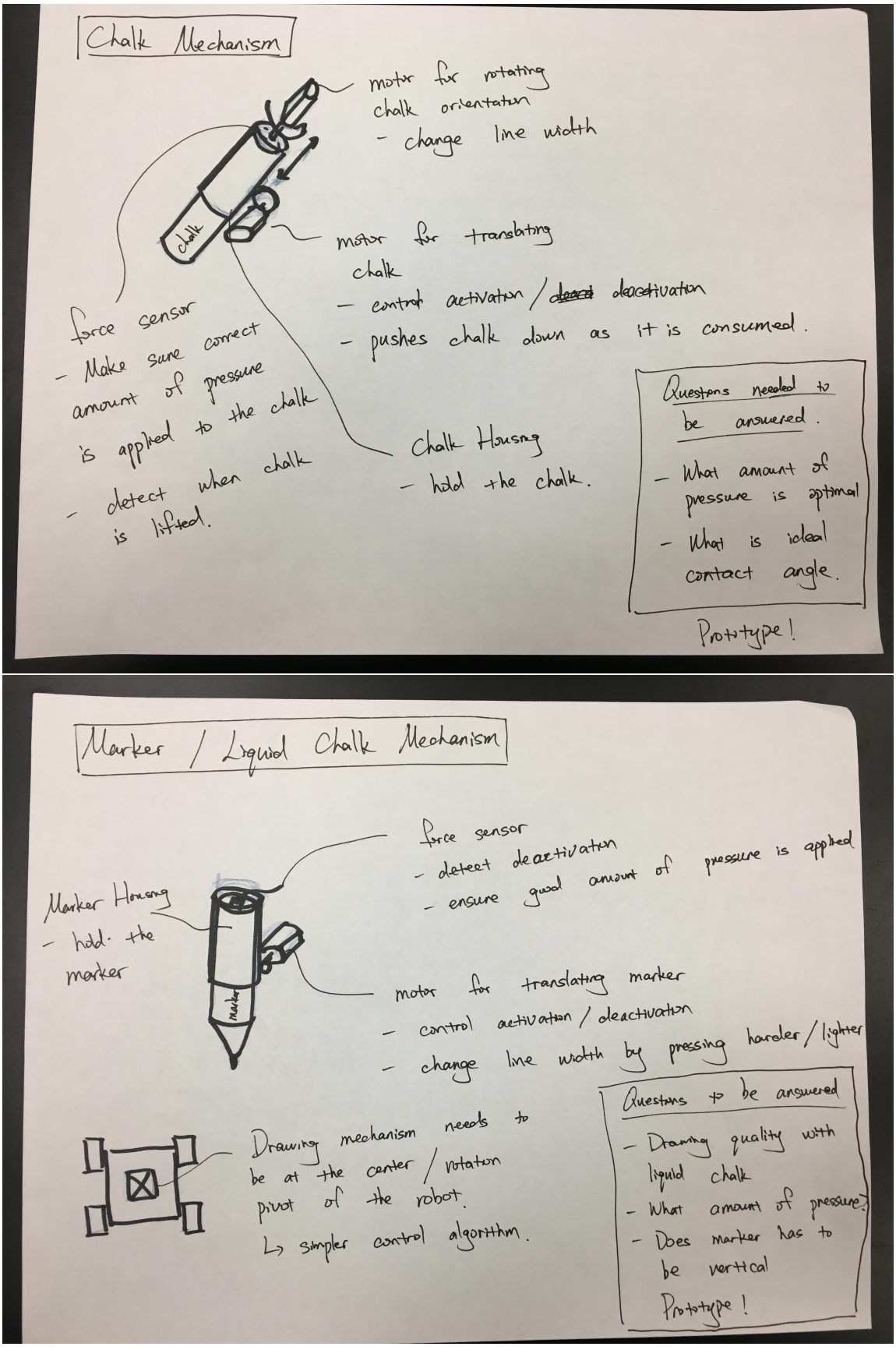

Figure 3: Writing Implement Sketch and Design Justifications

4.4 Requirements Fulfilled

Our modular implement system allows us to easily insert (FR 5), remove (FR 6) and replace (FR 7) tools quickly (NFR 14). The motorized control allows the robot to turn the writing tool on and off (FR 10). The size of the system is designed to be within weight (NFR 3), size (NFR 4) and budget (NFR 10) restrictions. Through continued prototyping process, we will consider reliability (NFR 8) and quality (NFR 6) and how we can build in error handling (NFR 2).

5 Locomotion

Description: The robot's locomotion system propels each individual robot across the working surface. The robot uses a holonomic locomotion system, it can move in any direction, which allows the robots to execute intricate designs. Mechanum wheels will allow the robots that flexibility, while at the same time preserving stability and ease of control. They also provide enough clearance for the writing implement to work effectively. A driving system that allows for multidirectional movement satisfies motion-related requirements (Requirements Specification, Sec. 5.1, FR 9). Additionally, the locomotion system contains encoders whose data is used by the localization module (Sec. 6).

Critical Components: Wheels or treads, motors and encoders.

5.1 Use Cases

5.1.1 Locomote

Description: The robot can use its array of motors and wheels to propel itself across the flat working surface. It can make arbitrarily sharp turns and acute curves in order to allow input drawings that incorporate such components.

5.2 Trade Study

One of our functional requirements with the highest priority (FR 1) is to be able to move in specified directions with a high degree of accuracy. Additionally, a medium high priority functional requirement (FR 9) is to have a drive control system that enables this movement. Therefore, it is clear from our requirements that locomotion is critical to our robot's performance. When considering locomotion options there are three large categories: aerial, legged and wheeled.

Our drawing occurs on the 2D plane. Therefore, any flight based system would have to constrain one translational degree of freedom and two rotational degrees of freedom. Doing so would require precise control algorithm and large power input. Practically speaking, this overcomplicates the problem for little benefit, and as such rules out airborne locomotion systems.

Therefore, locomotion options are limited to a legged or wheeled system. While a legged system would have the ability to traverse uneven terrain, such capability is not necessary for this project due to the assumption (A3) that the drawing surface is flat and homogenous. Compared to wheeled systems, legged systems usually involve more complicated control algorithms and electrical and mechanical components. In addition, legged systems often fail to ensure stability in the robot body, which for this system holds the writing implement. Considering system complexity and locomotion stability, it can be concluded a wheeled system is best.

In continuing to prioritize functional requirement (FR 1), we investigate other possible wheeled drive systems. Four different drive systems are evaluated: differential drive, Ackerman steering, four-wheel steering, and holonomic drive. Each of the four drive systems are evaluated based on mobility, motion accuracy and amount of damage done to the drawing surface.

Differential drive systems typically consist of two independently driven wheels and a non-actuated wheel. Such a drive system has good mobility due to its minimum number of driving wheels. However, the non-actuated wheel often creates unwanted resistance while turning, which both compromises the drawing accuracy and damages the drawing surface. Occasionally, the non-driven wheel can fall into a singularity and ruin drawing quality.

Next is Ackerman steering, where the back pair of wheels is fixed in orientation but the front pair can pivot. This drive system does not suffer from singularities and does not damage drawing surfaces. However, turning with ackerman steering often requires a large turning radius dependant on the wheel

base, which makes sharp corners difficult to draw. Despite a high accuracy of motion and minimal damage to the drawing surface, Ackerman steering has a low mobility and is not feasible.

Another option is four-wheel steering, which is similar to Ackerman steering but allows the back wheels to pivot as well. Even though such drive system can achieve in-place turn, doing so often requires the wheels to pivot in place, which can potentially damage the drawing surface. Hence, four-wheel steering ranks high in mobility and motion accuracy, but low in damage to drawing surfaces.

Holonomic drive involves independent control of four omnidirectional wheels. Such drive system can easily turn in place and does not cause any potential damage to drawing surfaces. The drive system's motion accuracy depends on wheel choice. Holonomic drive is often achieved with either omniwheels or mecanum wheels. Omniwheel does not resist any force in its lateral direction, resulting in low motion accuracy due to slippage. Mecanum wheels, on the other hand, resist sideways disturbances and ensure straight motion due to the wheel's obliquely oriented rollers.

In conclusion, the system's locomotion system will be a wheeled holonomic drive system with mecanum wheels, since such combination best satisfies system requirements and performs best in criteria concerning mobility, motion accuracy, and amount of damage to drawing surfaces.

5.3 Artistic Sketch

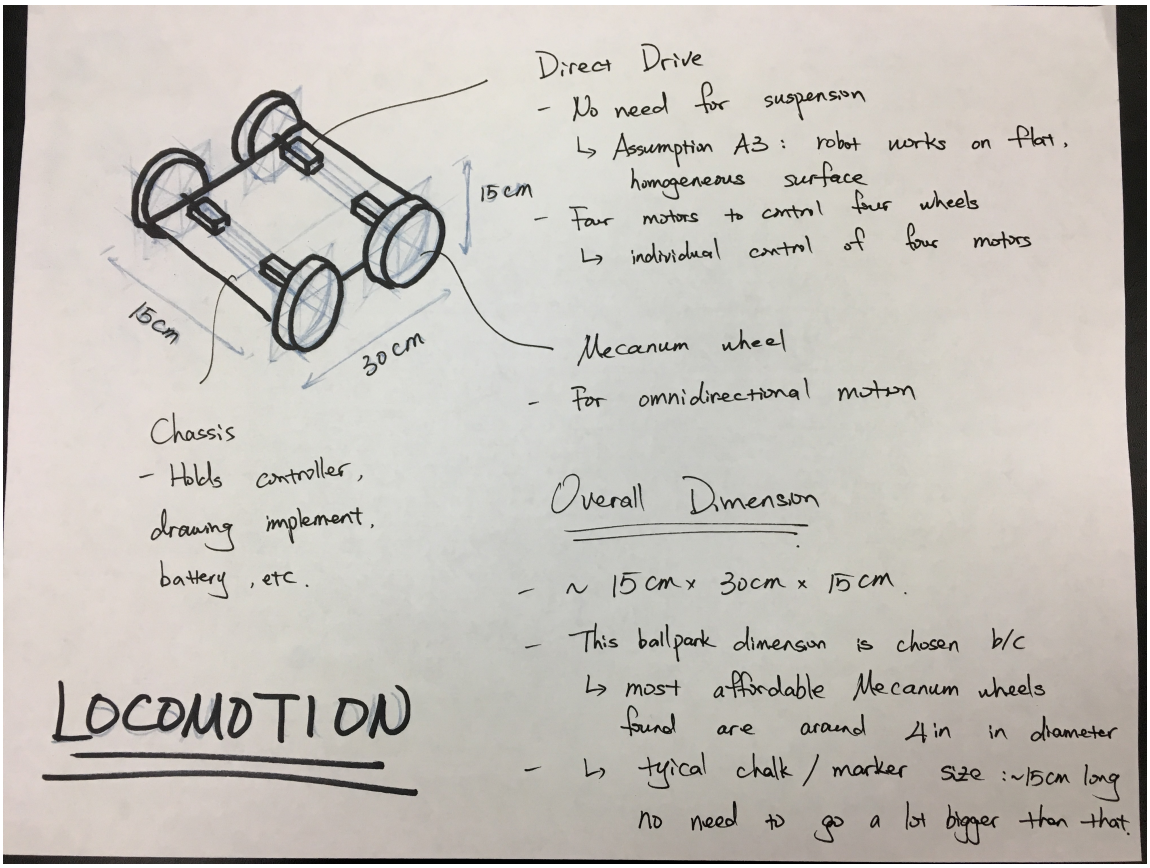

Figure 4: Locomotion Sketch

5.4 Requirements Fulfilled

By using mecanum wheels, we can achieve omnidirectional movement (FR 1) and reliable control (FR 9). Our emergency stop (FR 13) and error handling (NFR 2) will be integrated into the drive system through the communication network Sec. 9. Our mecanum wheels have been specified to stay within our weight (NFR 3), size (NFR 4) and budget (NFR 10) limits. The control of the mecanums will allow us to be efficient (NFR 5), due to their omnidirectional motion and speed, while maintaining quality (NFR 6), reliability (NFR 8) and safety (NFR 11). Additionally the omnidirectional control wil allow us positional (NFR 12) and rotational (NFR 13) accuracy.

6 Localization

The localization system uses an overhead camera and marker (Sec. 6.2) to determine a robot's position and orientation. It uses an overhead-mounted camera with a full, unobstructed view of the workspace and the robot agents. Static markers are mounted in the corners of the workspace, as well as atop the robot workers themselves. The corner markers determine the boundaries of the workspace, to which the input image can be scaled. The system use the robot-mounted markers to determine the position and orientation of each robot in relation to the workspace. It then communicates the positions and orientations of the robots to the scheduling module (Sec. 8). This module directly satisfies local and global localization requirements, as well as indirectly allows for safe and bounded motion from the robots (Requirements Specification, 5.1, FR 3, FR 4).

Critical Components: Localization algorithm, localization markers.

6.1 Use Cases

6.1.1 Localize

Description: The robot uses a combination of robot-mounted and static environmental markers, as well as a camera mounted above the working surface to determine the locations of each of the robots. This information is used to determine the motion plan of the robot.

6.2 Trade Study

In multi-agent planning, it is important to accurately localize robots' positions and orientations. Keeping in mind limitations in price and ease of use, we come up with two major methods for localization: vision based and marker based. They are described below.

Vision-based localization involves using cameras or other visual sensors to directly obtain information of the environment and localize the robots based on found landmarks in that environment. One example of this is SLAM (Simultaneous Localization and Mapping), often used by autonomous vehicles to simultaneously build maps and localize [1]. With this approach, robots could build small maps of their surroundings and match their locations to features they find in the environment. Benefits of this method include being location agnostic and requiring no additional parts or external setup. Pure vision systems are difficult to calibrate and localization accuracy can depend heavily on static surroundings, but this is relatively easy to guarantee given the requirements of the system (Requirements Specification, 2.3, 2.4 A1).

The other choice of methodology is marker based. Using markers placed around the drawing surface, robot agents can quickly locate these markers and their positions relative to each marker, and consequently triangulate their positions and orientations. While requiring additional setup and more parts to calibrate than vision based localization, existing technology makes it convenient and cheap to get marker based localization working. One example of a marker-based localization system is AprilTags [2], which can be described as 2D barcodes placed in a scene. Marker-based localization can be further classified into two subcategories: passive and active. Passive markers do not output any information and exist for the robot agents to observe and triangulate accordingly. AprilTags is an example of the passive marker system. On the other hand, active markers will "look at" robot agents to determine where the agents are, rather than the robots searching for markers. While less common, active systems behave well in conditions when the markers may not always be easily visible to robot agents [3].

Given the convenience and ease of use of marker-based localization, it is clearly the better choice. Additionally, since the workspace is limited to an indoor, flat, and homogenous area in our requirements

specification (Requirements Specification, 2.3, 2.4 A1) we can reliably count on overhead line-of-sight as opposed to having the robots observe or be observed by the passive or active markers. An indoor space also allows for easy mounting of an overhead camera. Consequently, we have decided to pursue a passive marker system observed from above by a single camera.

6.3 Artistic Sketch

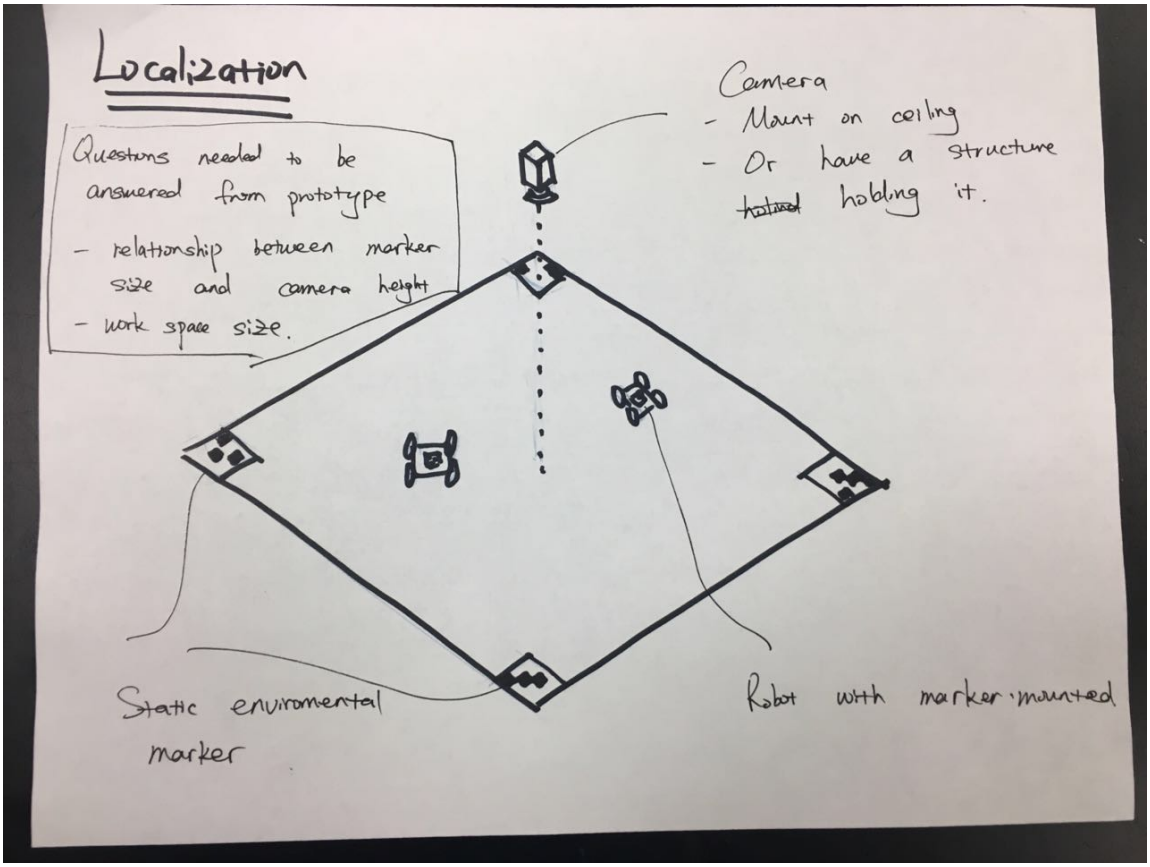

Figure 5: Localization Sketch

6.4 Requirements Fulfilled

Our localization system allows the robot to be autonomous (FR 2) and localize (FR 3), which gives them the information to stay within workspace boundaries (FR 4). Our localization will be refined to account for error handling (NFR 2). Precision in localization will enable our robot to produce a quality image (NFR 6) with positional (NFR 12) and rotational accuracy (NFR 13) that is coordinated with the rest of the system (NFR 9). As we continue with our overhead camera system we will develop reliability (NFR 8) and safety (NFR 11) into the system. We believe that our simple overhead camera will be within budget (NFR 10).

7 Image Processing

This subsystem takes a user-provided image as input, and processes in such a way that it can be read in and used by the work scheduling subsystem (Sec. 8). The input from the user incorporates a front-end user interface, satisfying requirements related to user interaction, which are defined below in Section Sec. 7.3.

The image processor takes input from the image scanner (Sec. 7) and produces information that is recognizable by the planning module (Sec. 8).

Critical Components: User interface, image sensor, Image processing algorithm.

7.1 Use Cases

7.1.1 Input User Image

Description: The system can take in an image provided by the user, and from it determine what the workspace should look like upon completion of the task. The user also inputs parameters such as scale, resolution, and possibly color.

7.2 Software Architecture

This software system was designed based around taking user input in the form of an image, and processing it into a format best suited for use by the work scheduling subsystem (Sec. 8).

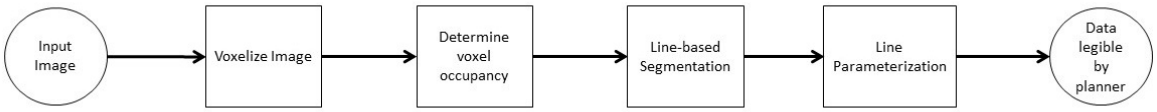

Figure 6: Image Processing Software Subsystem

The image processing pipeline (Fig.6) takes the image scanned by the user, and parses it into a form readable by the work planner. The goal of this software system is to create a series of lines that describes the image. To do this, the system first creates an occupancy grid of the input via voxelization. For a black and white image, the occupancy grid determines black or white. In the case of color, voxels are assigned color based on the image values inside of the voxel.

Once voxelization is complete, lines are formed from the voxels through a nearest-neighbor search. These lines are simply a series of voxel squares that pass through the image. In order to preserve curvature of lines from the input, the grid-delineated lines are reparamterized into paths using splines. These paths are then sent to the work planning module.

7.3 Requirements Fulfilled

Our input system, as seen in Fig.6 allows a user to input an image (FR 11). Each component is designed with error handling (NFR 2) and reliability (NFR 8).

8 Work Scheduling, Distribution, and Planning

Given the output from the image processor (Sec. 7), and parameters such as the number of robots and the size of the workspace, this module determines the work that will be distributed to each robot. The software architecture (Sec. 8.2) section explains the operational flow of the system..

Critical Components: Scheduling and distribution algorithm.
Further Research: Scheduling multiple robots in a loosely coordinated joint task presents an interesting research question that we are continuing to look into.

8.1 Use Cases

8.1.1 Decompose Plan

Description: The system generates an efficient distribution of work to each of the individual robots. The work distribution takes into account the amount of writing material needed, thereby limiting the need for reloading. It is also based on distance that the robots must cover and the speed at which they must travel to ensure that robot agents idle for as little time as possible.

8.1.2 Schedule Lines

Description: The system can divide each robot's specific tasks into subtasks, and determines the order in which they will be completed. This order ensures that the robot spends little time traversing unnecessary distance and completes its tasks in a short amount of time.

8.1.3 Plan Movement

Description: Each robot is capable of determining how to actuate its motors to get from its current location to a desired location. By using its motion model based on the properties of the robot's locomotion system, the system can queue motor values and adjust for noise and error along the way.

8.1.4 Detect Collisions

Description: The system can use a combination of localization sensors and motor encoders to determine if any robot has encountered an obstacle.

8.2 Software Architecture

A key aspect of our software system is distributing and planning the work to the robot agents, modeled in Fig.7. Two of our nonfunctional requirements are to be efficient and have the robots coordinate with each other (NFR 5, NFR 9). These two objectives inform the planning pipeline. We split the planning and coordination into two separate problems, allowing us to frame coordination as a scheduling problem [4]. Hence our work distribution and planning can be handled offline while our coordinate occurs online.

From our image processing Sec. 7.2, we recieve processed image data. Using this data, we compute the length of every individual line to be drawn. Guided by the assumption that line length correlates to amount of work done and the time to perform that work, we then use those lengths to load balance when assigning which lines should be drawn by each robot. Once each line has been assigned to each robot, each robot has a complete picture of their work and can be assigned an ordering to their lines. To limit wasteful movement, the robots order lines greedily: having completed one line, they pick their next lines (from their assigned set) based on which line has the closet endpoint to the line they have just finished. Having ordered lines, we now can determine each robot's set of paths.

Next, we briefly describe a sketch of our coordination mechanism. Each robot has a queue of paths, based on the set ordering, to execute. Given that a robot has not finished all of it's assigned paths, it removes a path from the queue. The path is then uniformly timed and converted into a trajectory. Given timing information, the system can check for path collision between any trajectories currently being executed. If the trajectory is not at risk of collision, then it can be executed - and is sent to the robot via the protocol mentioned in Sec. 9.2.

If the trajectory is in collision with a currently executing trajectory, then the trajectory being processed defers to the one being executed. The timing of the processing trajectory is adjusted by adding a pause to avoid collision. Given this modified timing we can then execute it.

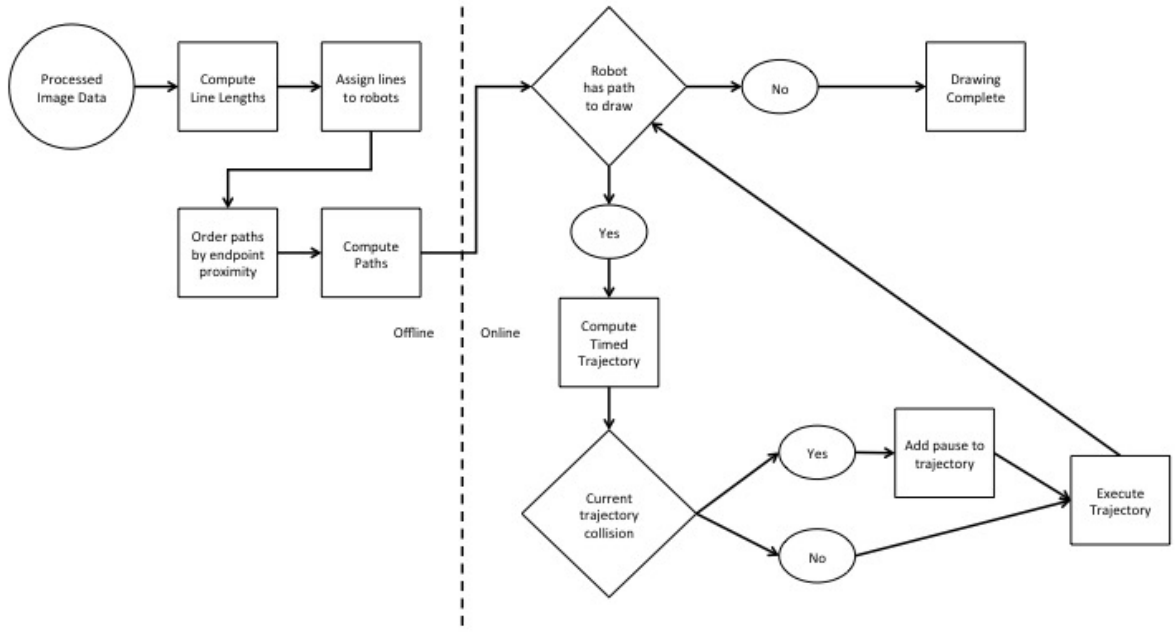

Figure 7: Planning and Scheduling Software Model

Once all paths have been drawn, the drawing is completed.

8.3 Requirements Fulfilled

Our planning framework allows the system to be autonomous (FR 2) and will enforce that the robots stay within bounds (FR 4). The planning framework will also control the writing implement on/off switch (FR 13) and keep track of total system progress (FR 12). Our planner is designed to partition work evenly for efficiency (NFR 5) and carefully plan for quality (NFR 6) and reliability (NFR 8). The online component of Fig.7 coordinates the robots to avoid collision (NFR 9). Additionally as we flesh out the rest of the system we will build error handling into the software (NFR 2).

9 Communication

The communication module is the link between the offboard system and the individual robots. To facilitate real-time changes in the working schedule, communication will be speedy and reliable. Potential communication protocols include WiFi and Bluetooth. Communication between the offboard system will allow individual robots to know their progress relative to the entire drawing (Requirements Specification, 5.1, FR 12).

Critical Components: Antennae, wireless communication protocol.

9.1 Use Cases

9.1.1 Communicate and Parse Data

Description: Individual robots report sensor readings to the offboard central computer, and the central computer sends updated localization information and schedules back to the robots.

9.2 Software Architecture

The communication diagram (Fig.8) describes how an individual robot communicates with the offboard path planning system. The system can be described by communication in four categories: motion commands (green), writing tool commands (blue), error handling (red), and localization (yellow).

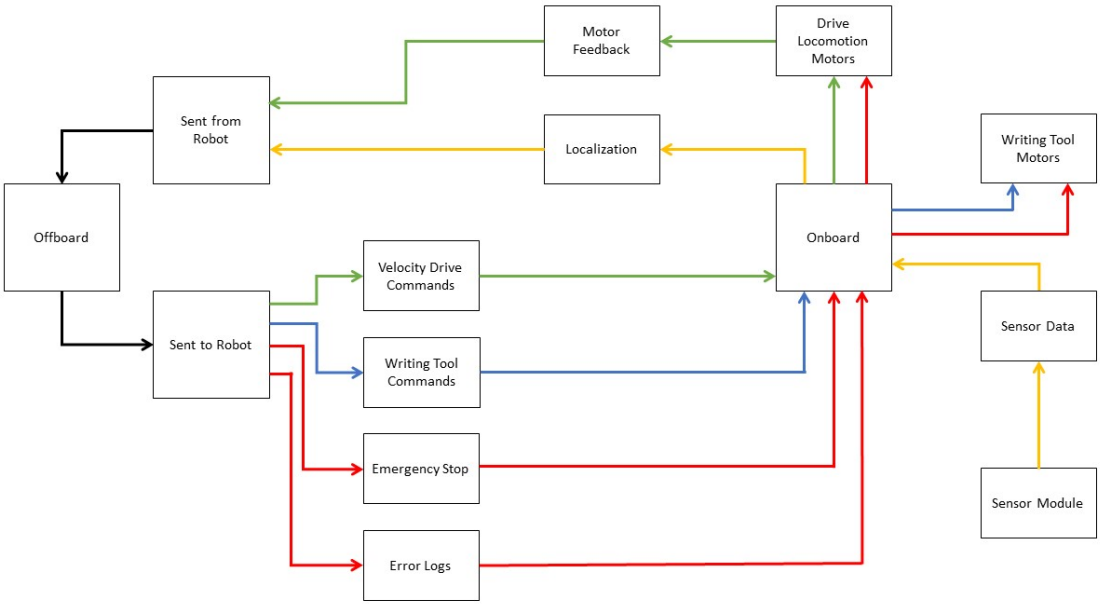

Figure 8: Communication Software Diagram. Green are the motion commands, Blue are the writing tool commands, Red is the error handling system, and Yellow is the localization system

The offboard system will use the planning algorithm to determine motion plans for the robot systems. It determines and then sends velocity commands to the onboard system. The onboard system passes these commands to the drive motors for motion.

To command the writing tool, the offboard system also decides how to move the writing implement based on localization of the robot systems. Similar to velocity commands, the onboard system receives commands for the writing tool, and commands the system accordingly.

Localization receives input from the sensor module, and computes position and orientation. This data is then sent back to the central offboard module to update planning and scheduling.

Error logs are generated by the offboard system based on localization and planning, and sent to each robot's onboard system for appropriate reaction. This includes the emergency stop, which will immediately shut down writing tool and drive motors.

9.3 Requirements Fulfilled

We believe that by partitioning communication as seen in Fig.8, we will have reliable communication (FR 8) and by using an offboard system for communication, each robot will know the progress of the drawing (FR 12). Also included in software architecture is the provision for an emergency stop (FR 13) and error handling (NFR 2). As we iterate on this design we will continue to consider reliability (NFR 8), budget (NFR 10) and safety (NFR 11).

10 User Interface

The user interface provides a unified system for user input. This is the system with which the user specifies the input image and monitors the robots' progress. The system will also display any error messages or anomalies detected, and how the user should address them. There will also be a system-wide kill switch in case of emergencies.

Critical Components: Screen, input device.
Planned Prototype: We plan to sketch our a prototype of our user interface.

10.1 Use Cases

10.1.1 Display Information

Description: The system is able to show a user information pertinent to system operation, including but not limited to the location of the robots, their battery level, amount of the task completed, estimated time of completion, and any existing obstructions or anomalies.

10.2 Requirements Fulfilled

Our prototype will be designed to allow the user to input a drawing plan (FR 11), cancel progress in case of emergency (FR 13) and provide an intutive experience (FR 14). Additionally this user interface must come complete with documentation (NFR 1) and error handling (NFR 2) and be reliable (NFR 8) and within budget (NFR 10). A low priority requirement is also to develop a user interface through a mobile app (NFR 7).

11 Power System

The power system supplies power to the rest of the system. Each robot has an onboard battery that is small and light enough to satisfy the size and weight requirements (NFR 3, NFR 4), but also provides enough uptime to last a entire drawing session. The power system will satisfy battery life and contribute to portability requirements.

Critical Components: Battery, voltage regulator modules.
Planned Trade Study: We are continuing to investigate the power system and will have an evaluation of our research soon.

11.1 Requirements Fulfilled

In prototyping our power system, we will have to satisfy the kill switch requirement (FR 13) while also using a battery that will allow each robot to operate for at least half an hour (FR 15). We will also balance having a power system that allow for smooth error handling (NFR 2), is within our weight and size restrictions (NFR 3, NFR 4), budget (NFR 10), and is reliable (NFR 8) and safe (NFR 11).

12 Installation

In addition to the robot system itself, the system installation will require:

- Adequately sized indoor drawing surface
- Localization markers
- Overhead camera mount
- Writing implement

We will be using an indoor obstacle-free surface in accordance with our requirements specification (Requirements Specification, 2.3, A1) which placed outdoor, non-homogenous drawing surfaces and obstacles out of scope.

We will set up the localization markers on the corners of the working surface, as well as atop the robots themselves. The system can then identify and calibrate to the localization markers to track the working surface and the robots. We will also adjust the camera mount so that it allows the system's camera a full, unobstructed view of the working surface. Finally, we will place the robots at set locations on the working surface and generate the input image.

13 Requirements Table

Requirement	Section
FR 1	Sec. 5
FR 2	Sec. 6, Sec. 8
FR 3	Sec. 6
FR 4	Sec. 6, Sec. 8
FR 5	Sec. 4
FR 6	Sec. 4
FR 7	Sec. 4
FR 8	Sec. 9
FR 9	Sec. 5
FR 10	Sec. 4, Sec. 8
FR 11	Sec. 7, Sec. 10
FR 12	Sec. 8, Sec. 9
FR 13	Sec. 5, Sec. 9, Sec. 10, Sec. 11
FR 14	Sec. 10
FR 15	Sec. 11
NFR 1	Sec. 10
NFR 2	Sec. 4, Sec. 5, Sec. 6, Sec. 7, Sec. 8, Sec. 9, Sec. 10, Sec. 11
NFR 3	Sec. 4, Sec. 5, Sec. 11
NFR 4	Sec. 4, Sec. 5, Sec. 11
NFR 5	Sec. 5, Sec. 8
NFR 6	Sec. 4, Sec. 5, Sec. 6, Sec. 8
NFR 7	Sec. 10
NFR 8	Sec. 4, Sec. 5, Sec. 6, Sec. 7, Sec. 8, Sec. 9, Sec. 10, Sec. 11
NFR 9	Sec. 6, Sec. 8
NFR 10	Sec. 4, Sec. 5, Sec. 6, Sec. 9, Sec. 10, Sec. 11
NFR 11	Sec. 5, Sec. 6, Sec. 9, Sec. 11
NFR 12	Sec. 5, Sec. 6
NFR 13	Sec. 5, Sec. 6
NFR 14	Sec. 4

References

[1] M. G. Dissanayake, P. Newman, S. Clark, H. F. Durrant-Whyte, and M. Csorba, "A solution to the simultaneous localization and map building (slam) problem," *IEEE Transactions on robotics and automation*, vol. 17, no. 3, pp. 229–241, 2001.

[2] E. Olson, "Apriltag: A robust and flexible visual fiducial system," in *ICRA*, pp. 3400–3407, IEEE, 2011.

[3] R. Cassinis, F. Tampalini, and R. Fedrigotti, "Active markers for outdoor and indoor robot localization," *Proceedings of TAROS*, pp. 27–34, 2005.

[4] P. A. O'Donnell and T. Lozano-Pérez, "Deadlock-free and collision-free coordination of two robot manipulators," in *ICRA*, IEEE, 1989.

Carnegie Mellon University

Robotics Capstone Project

Test Plan

Friction Force Explorers:

Don Zheng
Neil Jassal
Yichu Jin
Rachel Holladay

supervised by
Dr. David Wettergreen

Version 2.0
December 14, 2016

Contents

List of Figures

1 Introduction

This document provides a test plan for our robotics capstone project. The purpose is to provide a series of tests to verify and validate system design and implementation. Tests involve one or more subsystems, and can either validate design decisions or verify successful system operation. The test plan also documents cases in which the parts of system may fail, as well as describing the risk involved and methods to avoid these incidents. The tests described are designed to fully test requirements of this system.

2 Design Verification

2.1 Writing Implement

2.1.1 Performance Test: Loading

Test Question: Is a human operator able to reload the writing tool within the required time limit of 10s, and by how much?
Operational Procedure: With a writing tool already installed in the writing assembly, a human test subject will perform 3 reload tests which are separately timed.
Metric: Duration of the shortest reload time.
Acceptance Criteria: The shortest reload time is under 10s.
Requirement(s) Verified: NFR 14

2.1.2 Performance Test: Writing Quality

Test Question: Is the drawing produced by the robot of acceptable quality?
Operational Procedure: Using a fully loaded writing tool, the robot attempts to draw along a route with at least 4 ft. of travel distance and 3 turns exceeding 50 degrees. Verify that the resulting drawing is of acceptable quality.
Metric: Percent thickness of the route at its narrowest point compared to the maximum thickness of a line created with the writing tool. Boolean on whether or not there are complete breaks in the line.
Acceptance Criteria: The percent thickness is at least 70%, and there are no complete breaks in the line.
Requirement(s) Verified: NFR 6

2.1.3 Functional Test: Replace Writing Tool

Test Question: Is a human user able to replace the writing tool?
Operational Procedure: A human test subject attempts to replace a writing tool already inside the robot. **Metric:** Whether or not the human succeeds in the task before giving up.
Acceptance Criteria: The human must successfully replace the writing tool without giving up within time limits specified by requirements (2 minutes).
Requirement(s) Verified: FR 7 FR 5 FR 6

2.1.4 Functional Test: Writing Pressure Control

Test Question: Can the writing tool be actuated to move up an down?
Operational Procedure: With a writing tool in the writing assembly, the motors for moving up and down attempt to move throughout their range.
Metric: Whether or not the writing tool moves. This test measures the writing tool's ability to precisely control the writing implement within its vertical range of motion.
Acceptance Criteria: The writing tool must move through the writing assembly's full vertical range.
Requirement(s) Verified: FR 10

2.1.5 Functional Test: Simultaneous Driving and Writing

Test Question: Can the writing tool make a mark while driving?
Operational Procedure: With a fully loaded writing implement, the robot will mark a 1 ft. line on the writing surface.
Metric: Whether or not any discernible mark is made and the full distance is covered.

Acceptance Criteria: A discernible mark must be made and the full distance must be travelled without the robot becoming stuck or breaking.
Requirement(s) Verified: NFR 6

2.1.6 Functional Test: Marking

Test Question: Does the writing tool make a mark when pushed down?
Operational Procedure: The robot will press down on a writing surface with a fully loaded writing implement. Robot must mark with pressure necessary to mark the surface without damaging the surface or writing tool.
Metric: Whether or not a any discerible mark is made.
Acceptance Criteria: A discernible mark must be made.
Requirement(s) Verified: NFR 6

2.1.7 Functional Test: Force Sensor

Test Question: Can the force sensor measure applied force?
Operational Procedure: The robot will press a writing implement down onto a writing surface with three different pressure settings: optimal, underactuated, and overactuated. Writing pressure settings will determine how strong the mark on the surface is. This allows the robot to maintain a consistently strong mark on the writing surface throughout operation.
Metric: Whether or not the sensor can distinguish between the three different pressure settings.
Acceptance Criteria: The sensor must be able to distinguish between all three settings.
Requirement(s) Verified: NFR 6

2.2 Locomotion

2.2.1 Performance Test: Accuracy

Test Question: Is the robot able to drive with positional and rotational accuracy?
Operational Procedure: The robot drives along a predetermined testing route consisting of at least 3 feet of linear distance and 90 degrees of turn.
Metric: The difference of the robot's final position and orientation from the intended position and orientation.
Acceptance Criteria: The robot's position must be less than 1 inch away from the intended position and its orientation must be within 10 degrees of the intended orientation. This result must be achieved 90 percent of the time.
Requirement(s) Verified: NFR 12

2.2.2 Functional Test: Speed

Test Question: Can the robot reach a desired speed?
Operational Procedure: The robot will drive along a straight line for 5 ft. during a timed trial.
Metric: The time required for the robot to reach the end of the line.
Acceptance Criteria: The robot must reach the end of the 5 ft. testing course in 20 seconds. This test must be repeatable 90 percent of the time.
Requirement(s) Verified: NFR 5

2.2.3 Functional Test: Omnidirectional

Test Question: Can the robot drive omnidirectionally?
Operational Procedure: The robot will drive along a path that has a turn of more than 120 degrees.
Metric: Whether or not the robot can make the turn.
Acceptance Criteria: The robot must be able to make an in-place turn more than 120 degrees. **Requirement(s) Verified:** FR 1, FR 9

2.3 Localization

2.3.1 Performance Test: Robot Position Accuracy

Test Question: Is the localization system able to accurately determine the position of each robot?
Operational Procedure: Both robots sit stationary within the working bounds. The localization system then attempts to determine their locations.
Metric: The difference of the robot's actual position from the position reported by the localization system.
Acceptance Criteria: The reported position must be within 1/10 in. of the actual position.
Requirement(s) Verified: NFR 6, FR 4

2.3.2 Performance Test: Bounds Accuracy

Test Question: Is the localization system able to accurately determine the boundaries of the workspace?
Operational Procedure: The localization system attempts to determine the bounds of the workspace.
Metric: The total difference in distance between the reported corners of the workspace and the distance between the actual corners.
Acceptance Criteria: The total difference must not exceed 1 in.
Requirement(s) Verified: FR 4

2.3.3 Functional Test: Robot Position

Test Question: Can the localization system find the robot?
Operational Procedure: With a single robot within the working bounds, the localization system attempts to determine the robot's location. Robot operation out of bounds is also considered out of scope, and is undefined behavior.
Metric: Whether or not a location is returned by the localization system.
Acceptance Criteria: The system must return a location for the robot.
Requirement(s) Verified: FR 4, NFR 6, FR 3, NFR 13

2.3.4 Functional Test: Bounds

Test Question: Can the localization system find the working bounds?
Operational Procedure: The localization system attempts to find all four corners of the working bounds while they are all in its field of view.
Metric: Whether or not locations are returned for all four corners.
Acceptance Criteria: The system must return locations for all four corners of the working bounds.
Requirement(s) Verified: FR 4, NFR 6, FR 3

2.4 Image Processing

2.4.1 Performance Test: Accuracy

Test Question: How closely does the image processor output resemble the original image?
Operational Procedure: The image processor takes in a valid input image from the example input set (Appendix A) and produces a result.
Metric: The percentage of lines that were accurately captured by the image processing system.
Acceptance Criteria: The system must succeed in capturing 95% of the lines in the input image.
Requirement(s) Verified: NFR 6

2.4.2 Functional Test: Return Data

Test Question: Does the image processor return data usable to the planner?
Operational Procedure: The image processor takes in a valid input image from the example input set (Appendix A) and attempts to produce a series of lines from it.
Metric: Whether or not usable output is produced.
Acceptance Criteria: The image processor must be able to produce a series of lines for the planner.
Requirement(s) Verified: FR 11

2.4.3 Functional Test: Reject Improper Input

Test Question: Is the image processor able to detect and reject improper input?
Operational Procedure: The image processor takes in an invalid input image (not an image file or contains components other than lines).
Metric: Whether or not input is rejected.
Acceptance Criteria: The invalid input must be rejected.
Requirement(s) Verified: FR 11

2.4.4 Functional Test: Keep Lines Within Bounds

Test Question: Does the image processor generate drawing lines within the working bounds?
Operational Procedure: The image processor takes in a valid input from the example input set (Appendix A) and produces an output in the context of the bounds.
Metric: Whether or not all lines lie within the working bounds.
Acceptance Criteria: All lines must lie within the working bounds.
Requirement(s) Verified: FR 4, FR 11

2.5 Work Scheduling, Distribution and Planning

2.5.1 Performance Test: Executable Plans

Test Question: How consistent is the planner at generating executable plans, ie those that avoid collision and stay within bounds?
Operational Procedure: Using the example input set (Appendix A), run each input and check the plan for potential robot-robot collisions and out of bounds driving.
Metric: Ratio of number of unacceptable plans, those that would involve collision or driving out of bounds, over the total number of plans.
Acceptance Criteria: Almost all, 99% of plans would not involve collision or out-of-bounds if executed.
Requirement(s) Verified: FR 4, NFR 11

2.5.2 Performance Test: Execution Distribution

Test Question: How efficiently is execution time, i.e. the total time robots spend moving, distributed?
Operational Procedure: Using the example input set (Appendix A), run each input and record the total time each robot spends moving.
Metric: We define execution efficiency as $\frac{\min(execution(R_0), execution(R_1))}{\max(execution(R_0), execution(R_1))}$ where execution(R_0) refers to the execution time of robot 0 and execution(R_0) refers to the execution time of robot 1
Acceptance Criteria: Execution efficiency of 0.75.
Requirement(s) Verified: NFR 5, NFR 9

2.5.3 Performance Test: Drawing Distribution

Test Question: How efficiently is drawing time, i.e. the total time robots spend drawing, distributed?
Operational Procedure: Using the example input set (Appendix A), run each input and record the total time each robot spends drawing.
Metric: We define drawing efficiency as $\frac{\min(draw(R_0), draw(R_1))}{\max(draw(R_0), draw(R_1))}$ where draw(R_0) refers to the drawing time of robot 0 and draw(R_0) refers to the drawing time of robot 1
Acceptance Criteria: Drawing efficiency of 0.75.
Requirement(s) Verified: NFR 5, NFR 9

2.5.4 Performance Test: Speedup

Test Question: What speedup is achieved by using two robots instead of one?
Operational Procedure: Using the example input set (Appendix A), run each input first with one robot and then with two. Time the execution time of each variant.
Metric: The comparison of duration, i.e. $\frac{executiontimewith2robots}{executiontimewith1robot}$.
Acceptance Criteria: According to our requirements we expect a speedup of 2x.
Requirement(s) Verified: NFR 5

2.5.5 Functional Test: Collision Free

Test Question: Does the planner and executor generate collision free plans?
Operational Procedure: Using the example input set (Appendix A), run each input and check for any robot-robot collisions during execution.
Metric: Boolean across each plans on whether a collision occurred.
Acceptance Criteria: We only accept if collisions were avoided on 95% of our test cases.
Requirement(s) Verified: NFR 11

2.5.6 Functional Test: Autonomy

Test Question: Does the system require no user input beyond adding the image to be drawn (except for error handling)?
Operational Procedure: After having input a plan, press "Run" on the system and observe if the system requires user input to finish the drawing.
Metric: Boolean on whether user input was required, excluding input relating to errors.
Acceptance Criteria: Accept only if no input was required.
Requirement(s) Verified: FR 2

2.6 Communication

2.6.1 Performance Test: Uptime

Test Question: What is the uptime on our ability to communicate data to between the robots and the offboard system?
Operational Procedure: Run the system for a significant period of time (several hours) and record any communication downtime or data loss during communication.
Metric: Time duration of down communication and packet loss.
Acceptance Criteria: Operational 95% of the time.
Requirement(s) Verified: FR 8, FR 12, NFR 8

2.6.2 Functional Test: Sending and Receiving Data

Test Question: Can the robot send and receive data from the off-board device and can the off-board device send and receive data to the robot?
Operational Procedure: Send data from the off-board device to the robot and verify the robot received it. Send data from the robot to the off-board device and verify the off-board device received it.
Metric: Four booleans on whether the data is successfully sent and recieved on both ends.
Acceptance Criteria: We must succeed on all four accounts.
Requirement(s) Verified: FR 8

2.6.3 Functional Test: Data Parsing

Test Question: Can the data on each side (robot, off-board device) be parsed by each other?
Operational Procedure: Send data from the off-board device to the robot and verify the robot received it and can execute it. Send data from the robot to the off-board device and verify the off-board device received it and can respond to it.
Metric: Check whether the data was successfully parsed on all sides.
Acceptance Criteria: We require all data be parsable.
Requirement(s) Verified: FR 8

2.7 User Interface

2.7.1 Performance Test: Emergency Stop Speed

Test Question: How fast does the emergency stop shut down the system?
Operational Procedure: While the system is in use, press the emergency stop button and time how long it takes for everything to completely shut down.
Metric: Elapsed time.
Acceptance Criteria: It is vital to safety that our emergency stop shuts everything down within a

second.
Requirement(s) Verified: NFR 11, FR 13

2.7.2 Performance Test: Error Reporting Delay

Test Question: What is the delay between an error occurring and that error being reported to the user?
Operational Procedure: Given a list of known operational errors, intentionally trigger each error within the system and report the time between causing the error and it being reported to the user.
Metric: Averaged elapsed time across error reporting.
Acceptance Criteria: The average time to detect and report an error should be within 3 seconds.
Requirement(s) Verified: NFR 11, NFR 2

2.7.3 Performance Test: Error Understandability

Test Question: How understandable and informative are error messages?
Operational Procedure: Given a list of known operational errors, intentionally trigger each error while a non-developer user is using the system (while masking the error cause) and evaluate how well the user can determine the error. For example, while the system is drawing the user could be in a different room with only the error reporting device, making the user unable to see what errors the robots are facing.
Metric: Determine if the user can determine the error and knows how to react to or correct the error.
Acceptance Criteria: The user should be able to determine and react effectively for 90% of the errors.
Requirement(s) Verified: NFR 2, NFR 1, FR 14, NFR 7

2.7.4 Functional Test: Emergency Stop

Test Question: Does the emergency stop fully stop the system?
Operational Procedure: While the system is in use, press the emergency stop button and check if all systems halt their operation.
Metric: Boolean on whether every subsystem stops or not.
Acceptance Criteria: It is only successful if the boolean metric is true.
Requirement(s) Verified: FR 13

2.7.5 Functional Test: Error Reporting

Test Question: Is each operational error reported to the user?
Operational Procedure: Given a list of known operational errors, intentionally trigger each error within the system and report whether the error caused it reported to the user.
Metric: Each error must be reported correctly. Hence we can divide the number of correctly reported errors by the number of total errors caused to determine an error-reporting score.
Acceptance Criteria: Considering error handling is critical to performance, our system should have an error-reporting score of 90%.
Requirement(s) Verified: NFR 2, NFR 1, FR 14, NFR 7

2.8 Power System

2.8.1 Performance Test: Battery Duration

Test Question: How long can an individual robot run for on a single battery charge?
Operational Procedure: Charge a robot fully. Using the example input set (Appendix A), continue to input drawings until the robot is fully drained of power. Time how long this takes.
Metric: The duration of operational time given one charge
Acceptance Criteria: We accept this if the operational time exceeds the necessary duration time of 90% of our test drawing inputs.
Requirement(s) Verified: FR 15

2.8.2 Functional Test: Battery Life

Test Question: Can the robots complete a drawing from a single charge?
Operational Procedure: Using the example input set (Appendix A), we want to test the robots ability. For each input, fully charge each robot, send the input and keep track of whether the drawing is fully complete before the battery on either robot is fully drained.
Metric: The ratio of the number of completed drawings to the total number of drawings.
Acceptance Criteria: We want to be able to successfully draw 90% of the drawings in our example drawing input set.
Requirement(s) Verified: FR 15

3 Full System Validation

3.1 Performance Test: Painting Accuracy

Test Question: How closely does the drawn image resemble the original image?
Operational Procedure: Using the example input set (Appendix A), input each for the system to complete. After completion, overlap the original image with the image of final drawing captured from overhead camera. Rescale the two images so that they are in the same size. Evaluate the coherence of the two images.
Metric: The percentage of drawn lines that were within 3 pixels of difference compared to those of the original image.
Acceptance Criteria: The system must successfully and accurately draw 95% of the lines in the original image.
Requirement(s) Verified: NFR 6

3.2 Performance Test: Reliability

Test Question: How reliable is the system in terms of successfully complete a series of drawing tasks?
Operational Procedure: Command the system to finish the example input set (Appendix A). Measure the number of consecutive successful completion. Successful completion is defined as the system autonomously finishes painting and the painting process is free of errors including but not limited to localization breakdown, motor breakdown, or painting mechanism breakdown. Calling human interference with switching battery and drawing utility does not count as unsuccessful run.
Metric: Number of consecutive painting completion.
Acceptance Criteria: The minimum acceptable number of consecutive completion is 5.
Requirement(s) Verified: NFR 8

3.3 Functional Test: Size

Test Question: Is the robot agent too big to be portable, i.e. carry the robot through a standard door?
Operational Procedure: Measure the physical dimensions of the robot in terms of width, length, and height or in terms of diameter and height.
Metric: Numeric value of each length measurement; robot footprint; robot volume.
Acceptance Criteria: Must be less than 80 in. x 36 in. x 36 in.
Requirement(s) Verified: NFR 4

3.4 Functional Test: Weight

Test Question: Is the robot agent too heavy to be portable, i.e. able to be lifted by a normal person?
Operational Procedure: Measure the mass of the robot.
Metric: Numeric value of robot mass.
Acceptance Criteria: Must be less than 50 pounds.
Requirement(s) Verified: NFR 3

3.5 Functional Test: Budget

Test Question: Does the cost for developing this robotic system exceed our budget?
Operational Procedure: Document total amount of money spent for designing and constructing this robot system. This includes machining expense, part cost, and etc.
Metric: Total amount of money spent.
Acceptance Criteria: Total developing expense has to be less than $2500.
Requirement(s) Verified: NFR 10

3.6 Functional Test: Safety

Test Question: Is the robot safe during operation? Specifically, when collision happens, will the robot harm other robots, external environment, or human?
Operational Procedure: Count the number of sharp edges on the exterior of the robot. Also, measure the time it takes from the overhead camera detects collision to robot agent stops moving motors. Intermediate steps involved are: camera sends collision signal to system controller and system controller commands involved robot agent to stop its current action.
Metric: Number of sharp edges (angles less than 90 degrees); amount of time takes from detection to action.
Acceptance Criteria: Values for these two metrics need to be as small as possible. Zero sharp edges can be on the exterior - any edges from, for example, a rectangular chassis, should be rounded. The maximum amount of time is 1.5 seconds.
Requirement(s) Verified: NFR 11

3.7 Functional Test: Documentation

Test Question: Is the documentation for the developing process comprehensive and replicable?
Operational Procedure: Give the full documentation to another design group or stakeholder and inquiry if they can duplicate the project with those documents.
Metric: Boolean on whether or not reviewers can replicate the system development process.
Acceptance Criteria: Reviewers are confident to replicate system development process based on the documentation.
Requirement(s) Verified: NFR 1

4 Failure Modes

4.1 Writing Implement

4.1.1 Out of Writing Material

Description: This failure mode describes the situation when a writing implement is loaded inside a robot agent, and runs out of writing material, i.e. ink or chalk.
Cause: Overuse of the writing implement.
Effects: The robot agent moves around and attempts to continue drawing, without making physical marks.
Criticality: This is a critical failure, as it requires the user to replace the implement before drawing can continue. If the user fails to replace the implement, lines will be missing from the drawing.
Safety Hazards: There is no safety hazard associated with this failure mode.
Mitigation: Routine inspections of the writing material level can help mitigate this failure mode.

4.1.2 Writing Mechanism Failure

Description: This failure occurs when the mechanism that raises and lowers the writing implement does not work. This causes the robot to be incapable of either drawing a line or even moving on the writing surface.
Cause: This is caused by a communication failure, as described by Sec. 4.6.1, or more likely, by the raise/lowering mechanism breakdown.
Effects: The robot agent will be unable to alter whether or not it is writing as it moves. This can cause either missing lines or incorrect ones, depending on the state of the writing mechanism.

Criticality: This is a critical failure as it directly affects the quality of the lines being drawn. To continue opertion, users can resolve this issue after receiving an error message. However, it is possible that the drawing has already been compromised.
Safety Hazards: There is no safety hazard associated with this failure mode.
Mitigation: Routine inspections and careful handling of the writing mechanism can help mitigate this failure mode.

4.2 Locomotion

4.2.1 Inaccurate Motion

Description: This failure mode describes the situation in which a robot agent is unable to accurately follow motion commands. An example of this is if the robot is commanded to move 10 inches, but localization detects it only moving 4 inches.
Cause: Inaccurate motion could be a result of slippage of wheels, failed motor encoders, or failed driving motors. Encoders and the localization system can be used to determin which of these causes occured.
Effects: Inability to move accurately can cause a user-reported error, which the user can resolve to continue operation.
Criticality: This is a minor failure, as it does not end system operation and can be resolved by the user.
Safety Hazards: The only safety hazard is with regard to the drawing surface; slippage could cause minor destruction of the surface.
Mitigation: Proper maintenance on the driving mechanism and using high-quality, reliable components can help mitigate this failure mode.

4.2.2 Failure to Move Omnidirectionally

Description: Failure to make omnidirectional movements means a robot agent cannot move in an arbitrary direction on the flat plane represented by the drawing surface.
Cause: Similar to Sec. 4.2.1, this could be caused by wheels slippage. Alternatively, a broken wheel or motor could have this effect as well.
Effects: Failing to move omnidirectionally could result in incorrect drawings - the robot agent can no longer move along sharp curves to faithfully recreate the input image. When detected, robot operation should halt and robot should report to the user of this failure.
Criticality: This is a critical error, as it has a direct influence on the quality of the drawing.
Safety Hazards: As with Sec. 4.2.1, the only safety hazard is that a broken wheel could damage the drawing surface.
Mitigation: Proper maintenance on the driving mechanism and using high-quality, reliable components can help mitigate this failure mode.

4.3 Localization

4.3.1 Camera Failure

Description: A camera failure occurs when the localization camera, mounted above the drawing surface, is incapable of gathering and/or sending data to the off-board processor.
Cause: Two potential causes for a camera failure are insufficient power supplied to the camera, or improper mounting. Improper mounting can cause the camera to fall or hang, which results in skewed and mis-calibrated camera data.
Effects: The effect of camera failure results in localization being poor or impossible, which can halt operation. This can be temporary, as a user-reported error would be generated to resolve this issue.
Criticality: This failure is of medium importance, as, while it halts operation, it can be resolved by the user to continue the drawing process.
Safety Hazards: The only safety hazard exists if the camera falls entirely from its mount, in which case it may fall on a person below.
Mitigation: A stable and sturdy camera mount design and using a reliable camera can help mitigate this failure mode.

4.3.2 Unusable Localization Data

Description: This failure mode exists when the off-board processing system is unable to localize.
Cause: Causes include mis-calibrated camera data or incorrectly placed bounds tags. For example, the bounds tags could be placed in a shape that does not reflect the drawing surface accurately, resulting in incorrect localization. Blurry data could also result in misreading localization tags.
Effects: If localization cannot be completed, robot operation will halt to avoid performing undefined actions. This error can be resolved by the user recalibrating or fixing the source that causes bad data.
Criticality: Similar to Sec. 4.3.1, this failure is of medium criticality and can be resolved by the user.
Safety Hazards: There are no safety hazards that result from this failure mode.
Mitigation: Concrete setup procedures and multiple checks before operation can help mitigate this failure mode.

4.3.3 Vision Tag Occlusion

Description: Occlusion of the vision tags is when the camera does not have direct line-of-sight of any vision tag used for localizing robots and bounds.
Cause: Tag occlusion is likely the result of an obstacle unexpectedly entering the scene. This could be a person walking over the drawing surface or over the edges of the camera view, where the vision tags representing the surface bounds are located.
Effects: Inability to find a tag results in incomplete localization, and will pause operation until the user can resolve the issue. This guarantees all robots are tracked continually during operation, as well as staying within bounds of the drawing surface.
Criticality: This is a minor failure, as robot operation can easily be corrected and operation can continue.
Safety Hazards: There are no safety hazards that result from this failure mode.
Mitigation: Proper security of the area around the work surface can help mitigate this failure mode.

4.4 Image Processing

4.4.1 Unable to Process Input Image

Description: Inability to process user input refers to the image processing subsystem failing to determine a set of lines usable for work distribution, planning, and scheduling.
Cause: This could be caused by an unreadable input or input drawings that do not conform to requirements. An example of this would be an input that contains background noise, making it unsuitable for processing and drawing.
Effects: The effect is that another input, a corrected version of the initial input, will have to be supplied for the system to continue operation.
Criticality: This failure is critical to system operation, as no drawing can be made until the input can be properly processed.
Safety Hazards: There are no safety hazards involved in this failure mode.
Mitigation: A clear set of guidelines and directions for the human providing the input image can help mitigate this failure mode.

4.5 Work Scheduling, Distribution and Planning

4.5.1 Fail to Plan

Description: This failure mode occurs when the offboard system is unable to generate a valid plan for the robot agents. This means the main controller is unable to command the robots to successfully complete the input drawing.
Cause: Failure to create a valid plan could arise from an out of bounds drawing. Other reasons include the image processing result being incorrect, which forces the work planner to incorrectly assign and generate plans.
Effect: The system is unable to complete an invalid drawing, and cannot begin autonomous operation.
Criticality: This is a system-critical failure due to the fact that the system cannot recreate the drawing if it cannot generate a robot motion plan to do so.
Safety Hazards: There are no safety hazards associated with this failure mode, given that it is entirely software-based.

Mitigation: Clear guideines and instructions for the human providing the image can help mitigate this failure mode.

4.6 Communication

4.6.1 Loss of Connection

Description: A loss of connection occurs when a robot agent and the off-board processing unit are unable to send data between each other. As per Sec. 8, the robot system expects consistent communication. This means that a failure resulting in intermittent or sparse connection will be treated equivalently to no connection.
Cause: This failure mode is the result of a robot agent and the off-board unit being unable to connect. This could be the result of a hardware failure, in which the either of the robot or off-board device's transmitter fail. Other causes could be loss of signal due to distance between the two devices, or obstacles that attenuate or disturb communication.
Effects: In the case that the off-board device cannot communicate with the robot, robots should be aware of a dropped connection and cease all locomotion. This will prevent robots from moving out of bounds or into collision, as without connection they can no longer localize. If the robot cannot communicate with the off-board device, locomotion should also end. The robot cannot report errors or sensor information to the off-board device for planning and scheduling, which then risks incorrect drawing and motion.
Criticality: Loss of connection is high-risk with regard to completing the drawing task. Requirements do not specify the ability to recover a connection, so processing and drawing will end on signal loss.
Safety Hazards: The only risk is the robots going out of bounds, or colliding with each other. Both of these pose little hazard to bystanders, as the robots are designed to be safe in the event of human-robot collision (Sec. 11).
Mitigation: Redundant communication systems and robust communication protocols can mitigate this failure mode.

4.6.2 Incorrect Data

Description: This failure mode occurs when the robot receives bad data from the off-board device, or when the off-board device receives bad data from a robot agent. Bad data here refers to data that cannot be parsed by either end.
Cause: Garbage data could be the result of a low-quality connection with high noise, or if data being sent becomes corrupted. It could also occur due to controller inability to parse the data being sent.
Effects: Invalid and incorrect commands and information should be ignored by the robot agent or the off-board processor.
Criticality: Incorrect data is noncritical to task completion as incorrect commands will be reacted to accordingly and resent. For example, if the off-board device sends a locomotion command to a robot, but the command becomes corrupt. The localization system will see the robot not move, and attempt to send a similar motion command again.
Safety Hazards: There is no risk to unparseable data being sent between robot and off-board device.
Mitigation: Redundant communication systems and robust communication protocols can mitigate this failure mode.

4.7 User Interface

4.7.1 UI Navigation

Description: This failure occurs when a user is unable to navigate the UI to setup and begin the autonomous drawing process.
Cause: Causes of this effect could be an unintuitive user interface, a UI that lacks features necessary to run the system, or lack of user training to use the interface properly.
Effects: The only effect is that the system is unable to begin the drawing process.
Criticality: UI failure is noncritical to system operation, as the system can run without a graphical interface. However, it is critical for demo purposes as a demo user must be able to begin system operation.
Safety Hazards: No safety hazards are posed by this failure mode.
Mitigation: A clear and intuitive UI design as well as a short training phase for human operators can help mitigate this failure mode.

4.8 Power System

4.8.1 Insufficient Battery

Description: This failure mode arises when the battery for an individual robot agent is low or out of power.
Cause: Insufficient battery power is a result of overuse of the battery from autonomously drawing for too long.
Effect: The result is a robot agent being unable to move, draw, or communicate with the offboard system, as it has no subsystems being powered.
Criticality: This failure is critical, as it can pause and/or end robot operation. Robot agent battery must be replaced before operation can continue.
Safety Hazards: There are no safety hazards that result from the battery being low or out of power.
Mitigation: Regular battery level inspections and using reliable batteries can help mitigate this failure mode.

4.8.2 Battery Explosion

Description: This failure mode arises when the battery experiences catastrophic failure and EXPLODES.
Cause: Improper charging procedures or unsafe battery handling can cause this failure mode to occur.
Effect: Battery explosion can cause irreparable damage to the entire system, as well as to the humans in the area.
Criticality: This failure is of high importance. Battery explosion will cause the entire system to become unusable, and will undoubtedly require an extensive amount of work to repair.
Safety Hazards: There is a significant safety hazard for this failure mode, as with all situations involving fire and explosions.
Mitigation: Strict procedures for battery charging and handling can help mitigate this failure mode.

4.9 Full System

4.9.1 Out of Bounds

Description: This failure describes the situation when any robot agents move beyond the bounds of the drawing surface, as described by the boundary vision markers.
Cause: This error can be caused by either poor localization, or poor locomotion. If localization software believes the robot to be somewhere it is not, it may command the robot out of bounds. If the robot motors or wheels are not working properly, it may move out of bounds, despite being given correct motion commands.
Effect: The robot moving out of bounds introduces undefined behavior that could result in collision, incorrect drawings, or making marks with the writing tool that are not on the appropriate writing surface.
Criticality: This is a critical error, as it results in incorrect localization, drawing marks, and system operation. Entering this failure mode results in system operation ending.
Safety Hazards: This failure poses a safety hazard of collision, as robots that exist the bounds may collide with objects or people that it is not expecting.
Mitigation: Robust localization system and driving mechanism designs can help mitigate this failure mode.
Failure Tree: See Fig.1

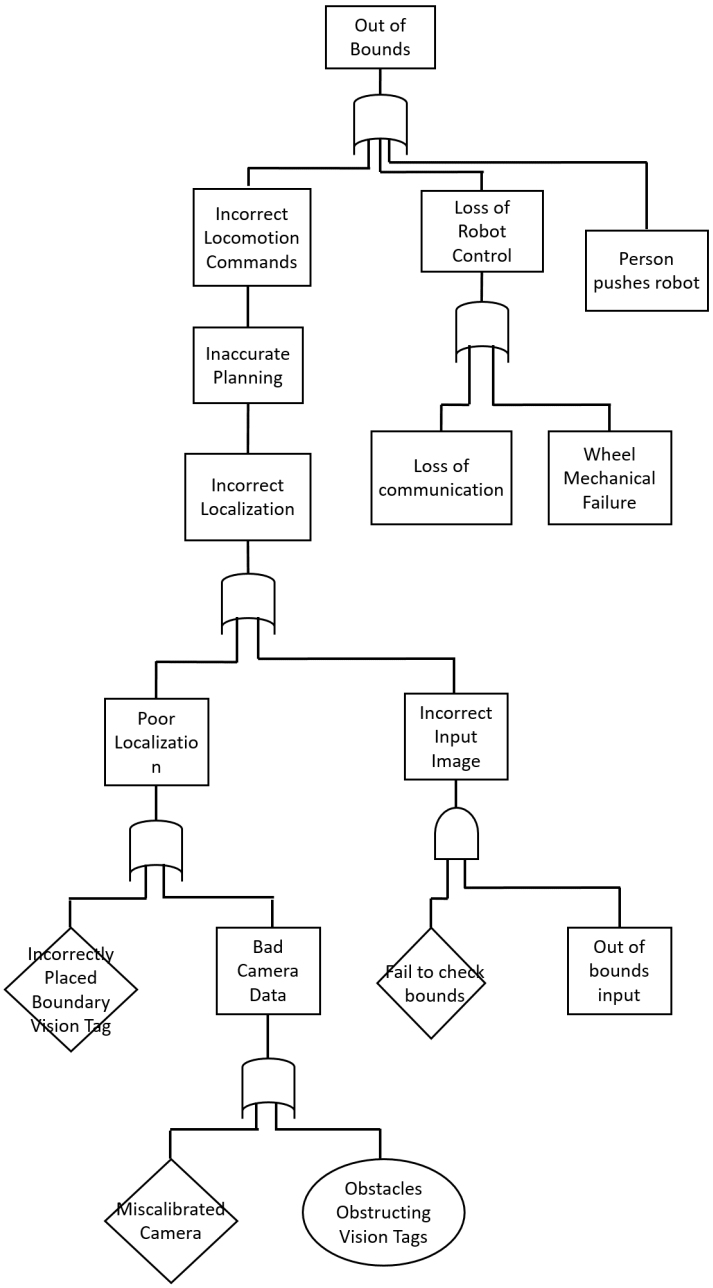

Figure 1: Fault Tree of the Failure Mode of the Robot going out of bounds.

4.9.2 Incorrect Markings

Description: Incorrect markings are made when a robot agent lowers the writing implement, and makes a mark in a location that does not match with the input drawing.
Cause: The cause of incorrect markings could be a result of the writing implement, locomotion, or localization subsystems. The writing implement subsystem may malfunction and lower the tool at an incorrect time. The locomotion subsystem may move the robot incorrectly while the writing implement is lowered, making markings where they are not expected. Localization error can cause markings to be in places that the system thinks are correct, but do not line up with the input drawing.
Effect: The effect of this failure is that the output drawing by the system is incorrect.
Criticality: Given that the goal of this robot subsystem is to accurately recreate an input drawing, failure to do so is a critical error.
Safety Hazards: There are no safety hazards associated with incorrectly marking the drawing surface.
Mitigation: A fail-safe writing implement design and robust localization/driving mechanism designs can help mitigate this failure mode.

4.9.3 Robot Collision

Description: This failure exists when robot agents collide with any obstacle, including each other or external obstacles.
Cause: Robot collision is a result of the locomotion, localization, or work scheduling subsystems failing. Locomotion failure could cause undefined motions by a robot agent, causing it to hit another robot, or move out of bounds (Sec. 4.9.1) and collide with an obstacle. Localization failure could result in incorrect motion commands, resulting in collision with unintended objects. Finally, poor motion planning has potential to require the robots to move into collision with each other.
Effect: Robot collision could damage the robot agents, or hurt human observers who are hit.
Criticality: This failure is of medium importance. The robots are designed to be safe (NFR 11), and therefore are unlikely to hurt or be significantly damaged by a collision.
Safety Hazards: There is a hazard of minor human injury, but as stated above the robots are designed to minimize human injury in the case of collision.
Mitigation: Robust localization and driving mechanism designs, as well as short-rage collision sensors can help mitigate this failure mode.

4.9.4 Intruder Collision

Description: Robot agents run the risk of colliding with intruders when humans or other objects unexpectedly move into the robot's workspace during operation.
Cause: Intruder collision can occur when humans and other objects are allowed to enter the workspace.
Effect: Intruding objects prevent robots from moving in their intended directions, severely affecting effectiveness of operation.
Criticality: This failure is of high importance. If an intruder occupies the same area as a part of the drawing, it will be impossible for the system to complete the task.
Safety Hazards: There is a hazard for minor human injury if very young humans are caught in the wheels or motors.
Mitigation: Tight security around the workspace, whether through physical barriers or signs, can help mitigate this failure mode.

4.9.5 Finger Jam

Description: This failure occurs when a human user's fingers are jammed in the motors for chalk lifting or locomotion.
Cause: Finger jamming can occur as result of improper operation of the robot, or if untrained users are allowed into the workspace during operation.
Effect: The motors could damage jammed fingers, causing injury to humans. The motors themselves would likely also be damaged.
Criticality: This failure is of high importance. It directly affects human safety and the operation of the entire system.
Safety Hazards: There is a hazard for severe human injury in the worst case.

Mitigation: Strict procedures and instructions for handling the robot can help mitigate this failure mode. Additionally, safety housings over moving parts will also decrease the likelihood of the failure mode occuring.

5 Risk Management

We assess risk through the following questions.

- Likelihood: how likely is this failure mode?
- Detectability: how likely can this failure mode be noticed?
- Criticality: how serious are the potential results caused by this failure mode?
- Priority: how much attention should be put in to address this failure mode?

The scales for priority is assigned based on likelihood, detectability, and criticality. We define the following scales and rank our failure modes accordingly.

Scale	Likelihood
1	Impossible (close to 0% chance)
2	Unlikely (25% chance)
3	Possible (50% chance)
4	Certain (75% chance)
5	Inevitable (100% chance)

Scale	Detectability
1	Impossible to detect (close 0% chance)
2	Unlikely to detect (25% chance)
3	Might be detected (50% chance)
4	Easy to detect (75% chance)
5	Obvious (100% chance)

Scale	Criticality
1	Harmless
2	May result in minor pain
3	May result in minor injury
4	May result in moderate injury and result in system immobility
5	May result in serious injury and death

Scale	Priority
1	Negligible
2	Could be addressed
3	Necessary to be addressed
4	Important to be addressed
5	Has to be addressed and avoided

Failure Mode	Likelihood	Detectability	Criticality	Priority
FM 4.1.1	2	4	2	4
FM 4.1.2	1	3	4	5
FM 4.2.1	1	2	4	3
FM 4.2.2	1	3	4	2
FM 4.3.1	1	5	5	4
FM 4.3.2	2	2	3	3
FM 4.3.3	3	5	3	4
FM 4.4.1	3	5	1	1
FM 4.5.1	2	5	4	5
FM 4.6.1	3	3	5	5
FM 4.6.2	1	3	5	4
FM 4.7.1	1	5	3	1
FM 4.8.1	3	4	5	4
FM 4.8.2	1	5	5	5
FM 4.9.1	2	4	4	3
FM 4.9.2	3	1	3	1
FM 4.9.3	1	5	3	3
FM 4.9.4	1	2	5	2
FM 4.9.5	2	1	5	5

6 Requirements Traceability Matrix

Requirement	Test
FR 1	T2.2.3
FR 2	T2.5.6
FR 3	T2.3.3, T2.3.4
FR 4	T2.3.1, T2.3.2, T2.3.3, T2.3.4, T2.4.4, T2.5.1
FR 5	T2.1.3
FR 6	T2.1.3
FR 7	T2.1.3
FR 8	T2.6.1, T2.6.2, T2.6.3
FR 9	T2.2.3
FR 10	T2.1.4
FR 11	T2.4.2, T2.4.3, T2.4.4
FR 12	T2.6.1
FR 13	T2.7.1, T2.7.4
FR 14	T2.7.3, T2.7.5
FR 15	T2.8.1, T2.8.2
NFR 1	T2.7.3, T2.7.5, T3.7
NFR 2	T2.7.2, T2.7.3, T2.7.5
NFR 3	T3.4
NFR 4	T3.3
NFR 5	T2.2.2, T2.5.2, T2.5.3, T2.5.4
NFR 6	T2.1.2, T2.1.5, T2.1.6, T2.1.7, T2.3.1, T2.3.3, T2.3.4, T2.4.1, T3.1
NFR 7	T2.7.3, T2.7.5
NFR 8	T2.6.1, T3.2
NFR 9	T2.5.2, T2.5.3
NFR 10	T3.5
NFR 11	T2.5.1, T2.5.5, T2.7.1, T2.7.2, T3.6
NFR 12	T2.2.1
NFR 13	T2.3.3
NFR 14	T2.1.1

Appendices

A Planner Inputs

The following are a set of sample drawing inputs that will be used to test the system. Some test inputs have been randomly generated while others were designed to stress test a particular feature.

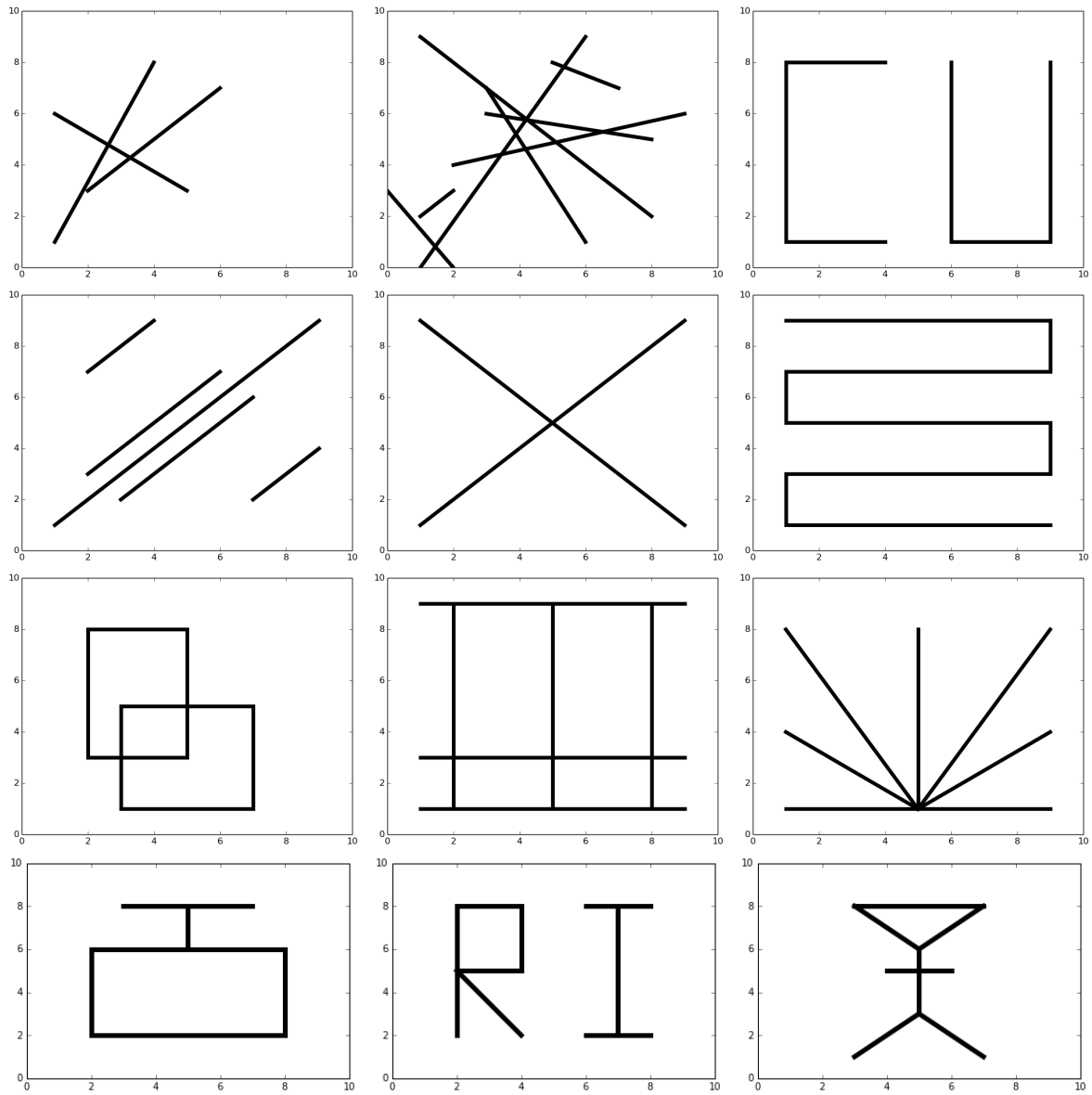

Figure 2: Planner Test Cases.

Carnegie Mellon University

Robotics Capstone Project

Detailed Design

Friction Force Explorers:

Don Zheng
Neil Jassal
Yichu Jin
Rachel Holladay

supervised by
Dr. David Wettergreen

Version 2.0
December 14, 2016

Contents

List of Figures

1 Introduction

This document provides implementation details for our robotics capstone project. This involves software, hardware, and overall system design. It includes parts lists and software libraries to be used, as well as assembly and setup instructions. Software architecture is described in detail enough to recreate using the same architecture and libraries. This document serves as a detailed description that can allow a 3rd party to recreate the project. In addition, system designed is traced back to the original requirements, and includes models to better define system architecture and usage.

2 System Diagrams

In this section, CAD diagrams for full system and subsystems and electronics diagram will be presented. Subsystems include localization system, locomotion system, painting mechanism, and power system. Detailed explanation of fabrication and assembly processes will be discussed in Sec. 4.

2.1 Full System Diagrams

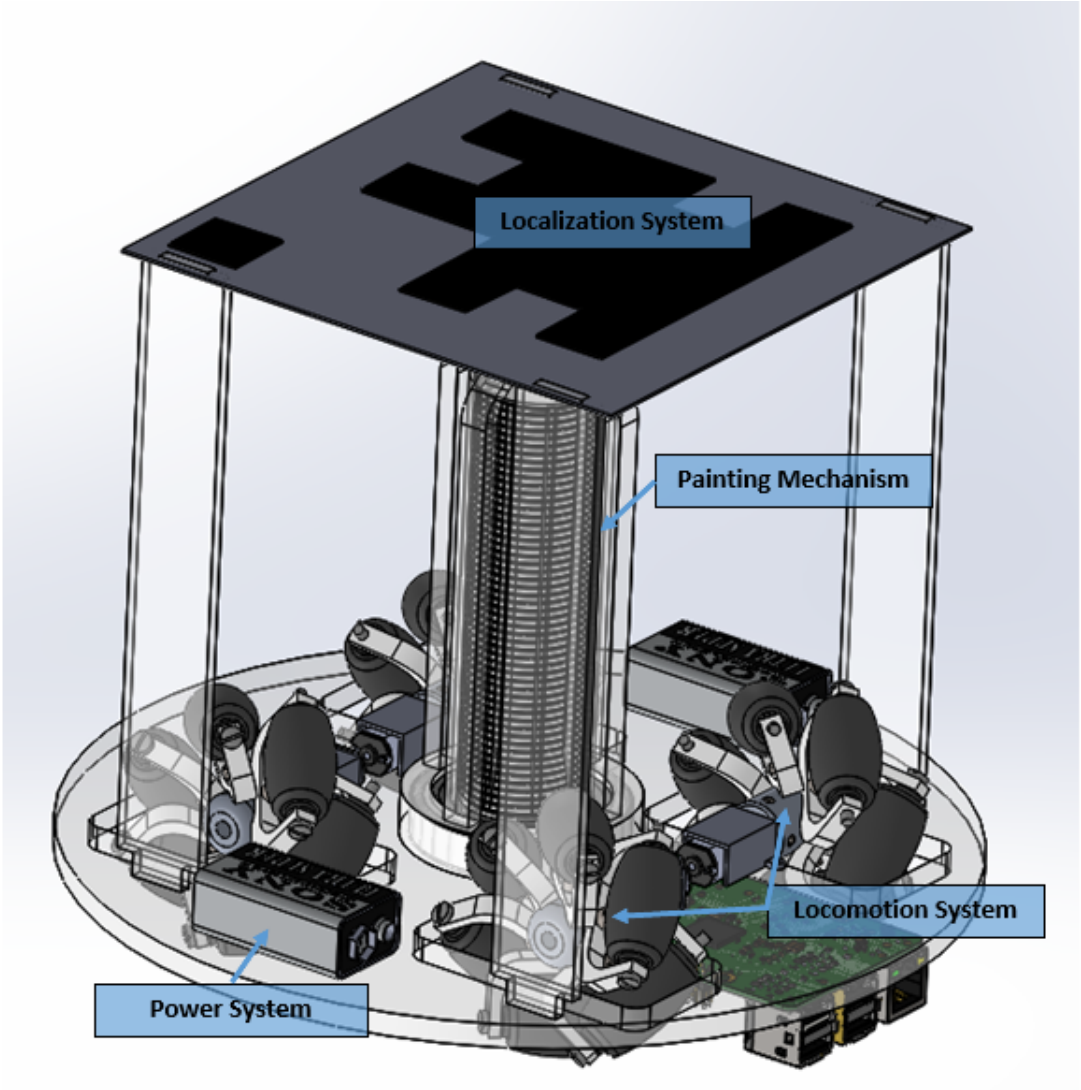

Figure 1: Full system diagram from isometric view.

Above labeled images show different views of the robot, including isometric, top, bottom, front, and side views. The labels indicate the relative positioning of each subsystem. As indicated in the CAD, the system is 23 cm in diameter and 19 cm tall. The total mass of the system is estimated to be 1.5 kg.

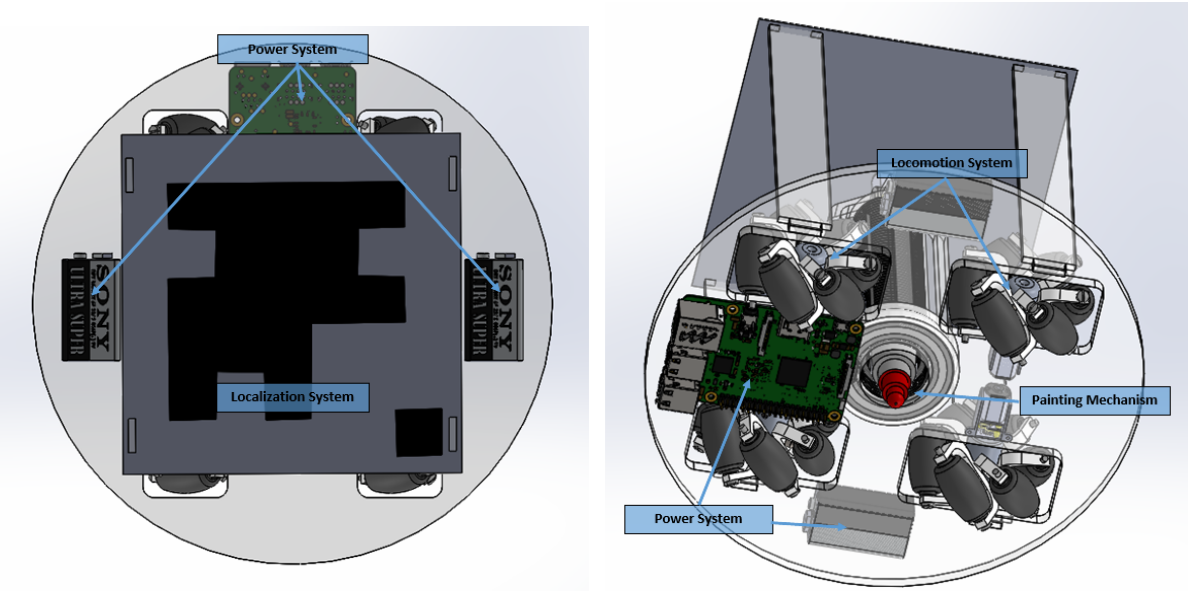

Figure 2: Full system diagram from top view (left) and from bottom view (right).

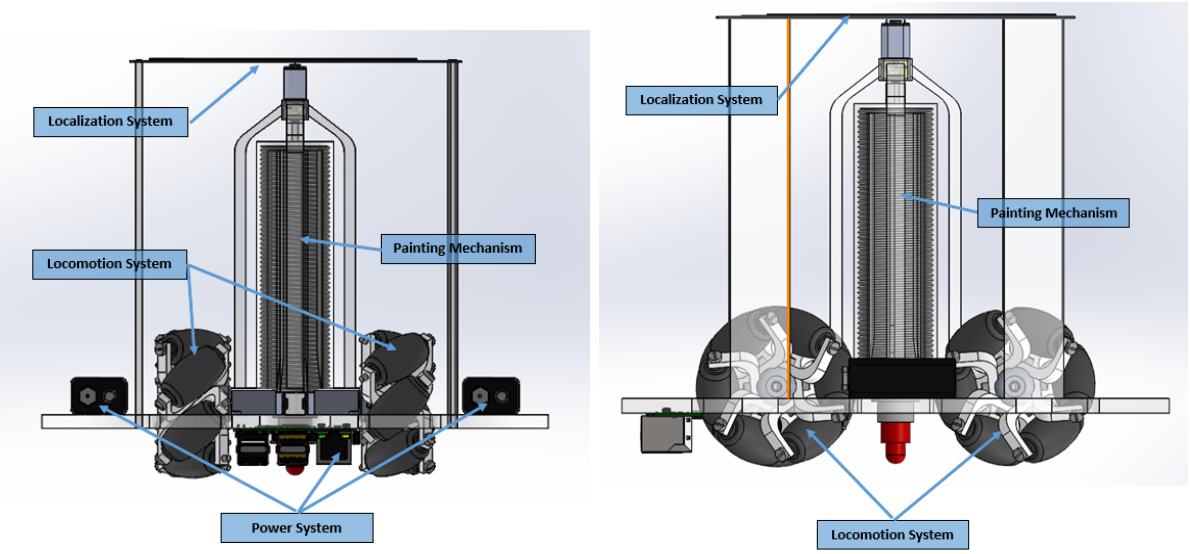

Figure 3: Full system diagram from front view (left) and from side view (right).

2.2 Component Diagrams

Above images show the assembled and exploded views of each subsystem. This section intends to give readers the basic familiarity with the functionality and degree of complexity contained in each subsystem. Sec. 4 will explains the fabrication and assemnbly procedures.

Painting mechanism is the most complicated mechanical system on the robot. Chalk holder is a 3D printed part with external thread. Rotor is another 3D printed part with internal thread. The subsystem uses a motor to control the lifting of the marker. Since the motor is fixed on the motor holder, as it activates, rotor rotates inside the motor holder along with the bearing and threads out the chalk holder, which lifts or lowers the installed chalk marker.

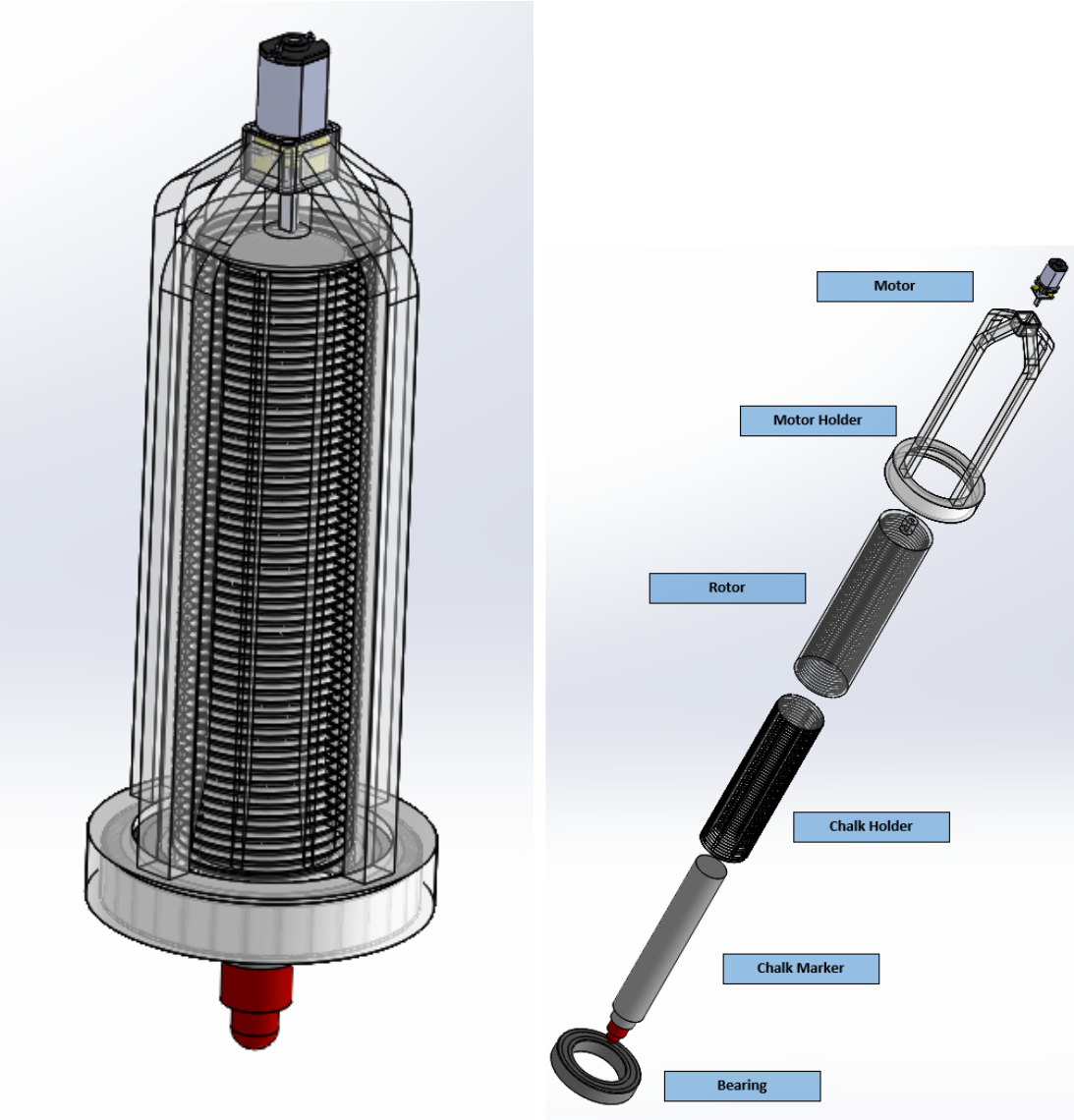

Figure 4: Painting mechanism from isometric view (left) and from exploded view (right).

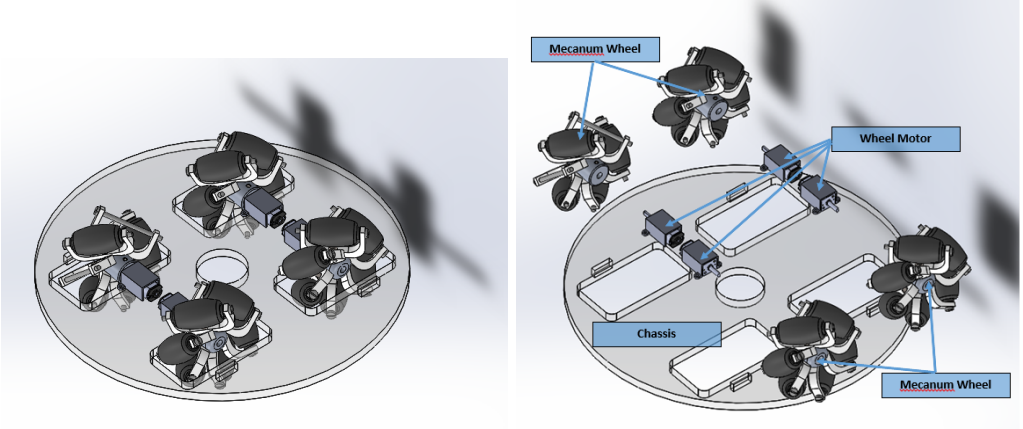

Figure 5: Locomotion system from isometric view (left) and from exploded view (right).

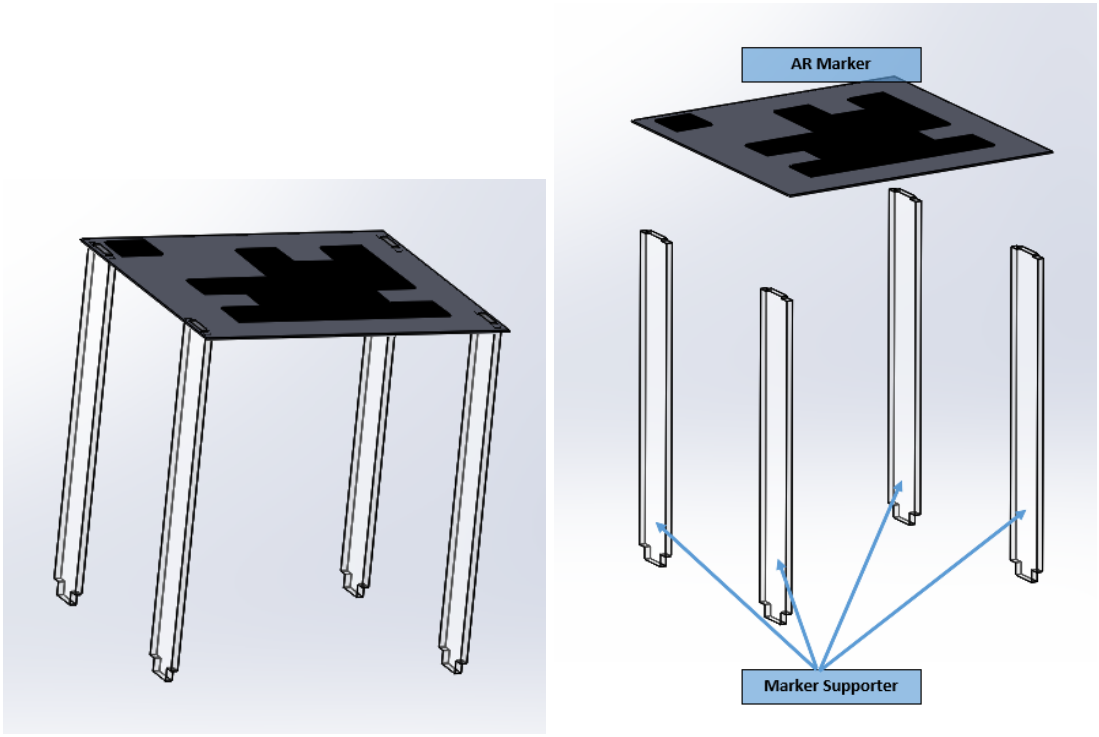

Figure 6: Localization system from isometric view (left) and from exploded view (right).

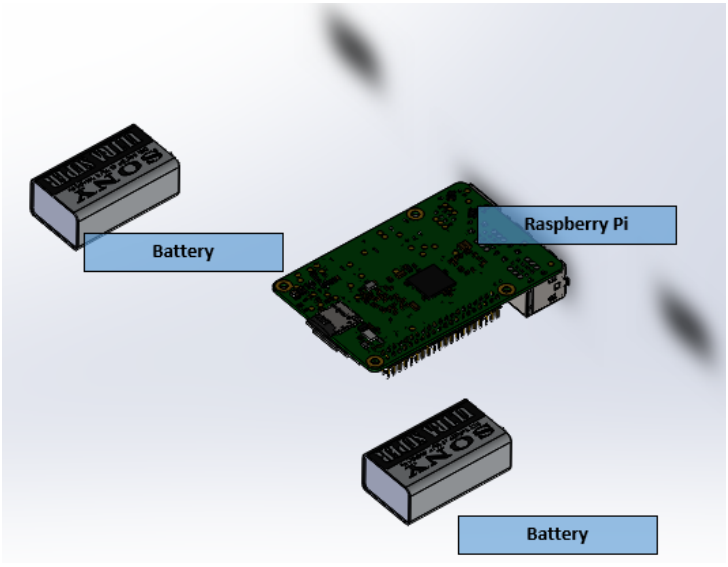

Figure 7: Power system.

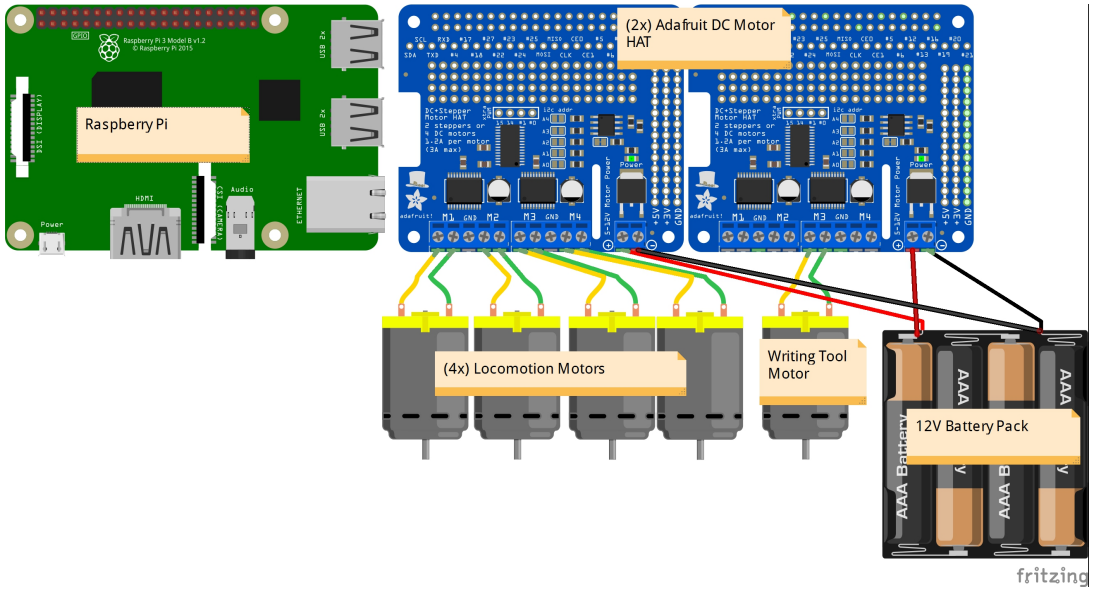

Figure 8: Full electronics diagram for a single robot.

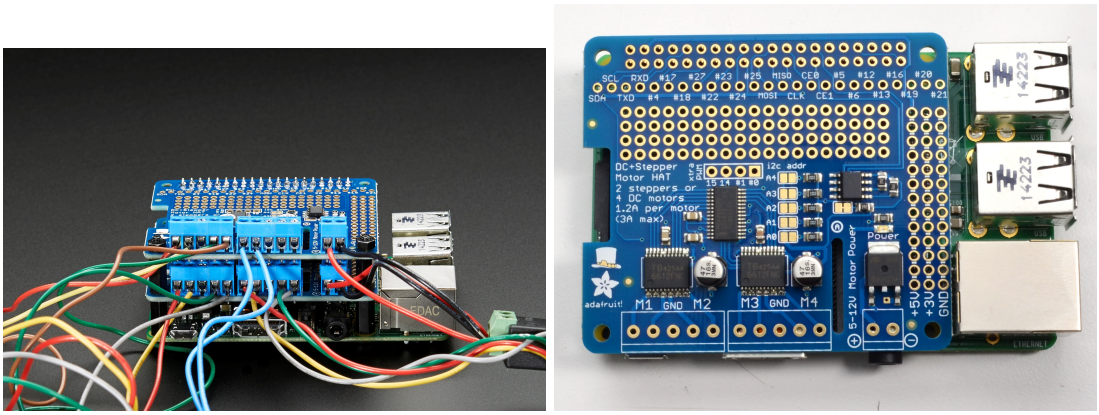

Figure 9: Approximation of what the Raspberry Pi and Motor HAT assembly will look like from the side (left) and the top (right).

2.3 Electronics Diagrams

The electronics system on each robot is detailed in the full electronics diagram Fig.8.

Two motor HATs will be stacked onto the Raspberry Pi for each robot, as illustratred in Fig.9.

For a port-to-port connection table for the electronics in the system, refer to Fig.10.

3 Operational Modes

This robot system can be defined by only a few operational modes. These modes represent various states that the system can take from the point of view of a user.

3.1 Mode: User Inputs

The first mode a user encounters is the user input mode. This mode involves the user creating and inputting an image to be drawn. This is all done via the user interface. In addition, this operational mode requires the user to setup the drawing surface, and place the robots in bounds.

Component Name	Connected to	On port
Raspberry Pi	Motor HAT 1	GPIO array
	Portable power brick	USB power header
Motor HAT 1	Motor HAT 2	GPIO extender
	Locomotion Motor 1 +/-	M1 +/-
	Locomotion Motor 2 +/-	M2 +/-
	Locomotion Motor 3 +/-	M3 +/-
	Locomotion Motor 4 +/-	M4 +/-
	Raspberry Pi	GPIO header
	12 Battery Pack +/-	Power +/-
Motor HAT 2	Motor HAT 1	GPIO header
	Writing motor	M3 +/-
	12V Battery Pack +/-	Power +/-

Figure 10: Electronics connections table.

3.2 Mode: Preprocessing and Planning

After the system is set up, the offboard processing system runs preprocessing to connect wirelessly to the robots, calibrate any localization, and determine the trajectories for the robot agents to follow. To do this, the planning system uses the user-generated drawing and parses it into a series of lines. It then runs these lines through the path planner to determine a series of trajectories for each robot. While these may be optimized to avoid collision, minute differences in performance and actual-robot speed require real-time collision detection as well.

3.3 Mode: Driving and Drawing

This is the main operational stage, in which the actual result of the system is generated. The offboard system will continually use updating localization and odometry to send commands to navigate the robot agents around the drawing space. In addition, the offboard processor will send commands to raise and lower the writing implement to create the drawing based on the robot position. Robot agents are responsible for parsing data to actuate motors, and sending odometry and other logging information back to the offboard processor. During this stage, the user observes the drawing process, pausing appropriately if they want to pause or stop execution.

3.4 Mode: Error Handling

The error handling operational mode does not occur regularly. This mode occurs whenever the offboard system receives information (from localization, or the robot agents) that an error has occurred. Depending on the issue, the system will shut down or pause. If the system pauses and it is possible to recover from the error, the user is responsible for correcting any issues and resuming operation. The user is unable to assist the system from recovering from fatal errors. Any errors and warnings are displayed on the UI for the user to see.

4 Hardware Fabrication Process

4.1 Writing Implement

4.1.1 Fabrication Procedure

As shown in the exploded view of the painting mechanism Fig.4, this subsystem contains six componenents: bearing, liquid chalk marker, chalk holder, rotor, motor holder, and motor. Bearing, liquid chalk marker, and motor will be purchased online. These off-the-shelf components are from vendors: McMaster, Amazon, and Pololu, respectively. The other three components, due to their complex geometry, will be 3D printed.

4.1.2 Assembly Instruction

1. Screw the chalk holder inside the rotor.
2. Slide the rotor into the motor holder.
3. Align the motor with the D-shape hole on the top of the rotor and with the motor housing on the top of the motor holder.
4. Press fit the bearing into the motor holder.
5. Asseble the mechanism on the chassis.
6. Due to the 3D printed ribs inside the chalk holder, users can then install and uninstall liquid chalk markers with ease.

4.2 Locomotion

4.2.1 Fabrication Procedure

As indicated in the exploded view Fig.5, the locomotion system contains four mecanum wheels, four motors, and a chassis. Mecanum wheels and motors will be purchased from the vendors Roboshop and Pololu. The robot chassis will be laser cut with acrylic.

4.2.2 Assembly Instruction

1. Attach mecanum wheels to motors.
2. Slide motor cases on the four motors.
3. Screw in the four motor-wheel assemblies to the acryilc chassis.

4.2.3 Electronics

On each robot, the Raspberry Pi will be connected to an Adafruit DC Motor HAT, which is able to controll all 4 DC motors on the robot. Communication between the Raspberry Pi and the Motor HAT will be handled through the GPIO pins on the Pi. For specific information on the pinouts, see Sec. 2.3.

4.3 Localization

4.3.1 Fabrication Procedure

The localization system is comprised of a ceiling-mounted webcam with full view of the workspace and AR markers (AprilTags) mounted the robots and on the four corners of the workspace. A ceiling-mounted webcam will be purchased online from Amazon. AprilTags will be printed on paper and attached to the robots and on the bounds of the writing area. As shown in the exploded view Fig.6, four marker supporters are required to mount the AR marker on each robot. These supporters will be made out of laser patterned acrylic.

4.3.2 Assembly Instruction

1. Mount the webcam on the ceiling.
2. Place and secure the printed AprilTags around the four corners on the workspace.
3. Glue four laser patterned marker supporters to the AprilTag for robot.
4. Glue the marker supporters to the robot chassis.

4.4 Communication

The built-in WiFi card on each robot worker's Raspberry Pi 3 will communicate with the offboard laptop computer. The WiFi card is 802.11n compliant, ensuring that packet loss will be minimized and high speed is achieved.

4.5 User Interface

The hardware for the user interface is the laptop on which the central processing is done. Users will be able to use the laptop's mouse and keyboard to specify parameters of operation. The laptop's monitor will be used to observe the workers' progress and view any errors during operation.

4.6 Power System

4.6.1 Fabrication Procedure

As shown in Fig.7, power system contains a Raspberry Pi and two power supplies. All of them will be purchased online.

4.6.2 Assembly Instruction

1. Tape Velcro strips on the chassis, the Raspberry Pi, and the batteries.
2. Attach the Rasberry Pi and the batteris to the chassis via Velcro.

4.6.3 Electronics

The Raspberry Pi will draw its power from a 5V USB power bank. The battery pack can hold 3400 mAh of charge, which should allow more than 2 hours of operation. The battery packs are easily swappable and cheap, meaning that our budget allows for multiple packs. Additionally, battery packs can charge each other, making it uncessary to power down the Raspberry Pi when additional charge is needed.

A separate locomotive battery pack will provide power to the motors. This battery pack contains 8 rechargable AA batteries, giving 12V and containing 2000 mAh of charge. This allows around 30 minutes of continuous operation for the motors, requiring only a quick battery swap when charge is depleted.

4.7 Full System

4.7.1 Assembly Instruction

To reiterate:
1. Localization system is attached to the chassis via glue.
2. Motor-wheel assembles are attached to the chassis via screws.
3. Painting mechanism is attached to the chassis by gluing on the bottom side of the motor holder.
4. Power system is attached to the chassis via velco strips.

4.7.2 Computation

A WiFi-compatible laptop computer will handle all offboard computation. This includes planning, scheduling, localization, locomotion command generation, and managing communication with the robot workers. Only communications, a computationally light task, must be managed in real time. This means that the computational speed of the laptop is largely unimportant to proper operation of the system.

Each robot worker will be equipped with a Raspberry Pi 3 to manage onboard computation, motor controller operation, and communication with the offboard system. The Raspberry Pi 3 is a fully functional Linux system with easily accessible GPIO pins, which can be used for the motor controller. The 1.2GHz quad-core processor will be more than sufficient for the required onboard computation. It also has a built-in WiFi card, which simplifies communications and decreases the number of necessary parts.

Part name	Link	Purpose/notes	Price/Unit	Quantity	Subtotal
Raspbery Pi 3	https://www.amazon.com/Raspberry-Pi-RASP-PI-3-Model-Motherboard/dp/B01CD5VC92	Computing platform	$35.99	3	$107.97
Raspberry Pi 3 case	https://www.amazon.com/Eleduino-Raspberry-Model-Acrylic-Enclosure/dp/B01CQRROLW	Pi protection	$7.69	2	$15.38
Pi cobbler Kit	https://www.adafruit.com/products/2029	Breakout to breadkb	$6.95	2	$13.90
MicroSD card	https://www.amazon.com/SanDisk-microSDHC-Standard-Packaging-SDSQUNC-032G-GN6MA/dp/B010Q57T02/ref=sr_1_3	Pi storage	$10.99	2	$21.98
Power Bank	https://www.amazon.com/dp/B01CU1EC6Y/ref=psdc_7073960011_t2_B005X1Y7I2	Pi Power	$16.99	4	$67.96
Webcam	https://www.amazon.com/dp/B0080CE5M4/?tag=techuflistsum-20	Localization	$32.50	1	$32.50
USB extension cable	https://www.amazon.com/AmazonBasics-Extension-Cable-Male-Female/dp/B00NH12O5I/ref=sr_1_8?s=electronics&ie=UTF8&qid=1480795100&sr=1-8&keywords=USB%2Bextension&th=1	Camera mounting	$5.52	2	$11.04
Breadboards	https://www.amazon.com/Aketek-Solderless-BreadBoard-tie-points-power/dp/B01258UZMC/ref=sr_1_4?s=electronics&ie=UTF8&qid=1480795347&sr=1-4&keywords=breadboard	Electronics	$8.99	1	$8.99
Adafruit DC Motor HAT	https://www.adafruit.com/product/2348	Motor control	$22.50	6	$135.00
Wires	https://www.amazon.com/Adafruit-Accessories-Hook-up-Spool-22AWG/dp/B00XW2O95A/ref=sr_1_4?ie=UTF8&qid=1480796696&sr=8-4&keywords=solid+core+wire	Electronics	$19.95	1	$19.95
Motors	https://www.pololu.com/product/3043	Locomotion	$17.95	10	$179.50
Rechargable Batteries	https://www.amazon.com/EBL-Rechargeable-Batteries-2800mAh-Battery/dp/B00RVH0WA8/ref=sr_1_14_a_it?ie=UTF8&qid=1480899394&sr=8-14&keywords=rechargeable+AA	Locomotive power	$31.99	1	$31.99
Battery holder	https://www.adafruit.com/products/875	Power	$5.59	2	$11.18
Mecanum Wheel Set (2x Left	http://www.robotshop.com/en/60mm-mecanum-wheel-set-2x-left-2x-right.html	Locomotion	$70.04	1	$70.04
Acrylic 3/16" by 12" by 12"	https://www.mcmaster.com/#8560k211/=15b7ebv	Chassis material	$10.13	1	$10.13
Artista Pro Water Resistant M	http://www.chalkink.com/Artista-Pro-6mm-Chisel-Tip-Markers-s/31.htm	Marker	$4.99	2	$9.98
Pro Ink Container	http://www.chalkink.com/category-s/202.htm	Marer ink	$69.99	1	$69.99
Tygon PVC Clear Tubing 1/4'	https://www.mcmaster.com/#6516t22/=15b7mzg	Tube for ink reservoi	$14.15	1	$14.15
Ball Bearing	https://www.mcmaster.com/#5972k172/=15bduc0	Painting mechanism	$19.41	1	$19.41

Figure 11: Full Parts List

5 Parts List

Our full parts list and associated budget is provided below in Fig.11. Our total predicted budget comes out to $851.04, not including shipping and handling costs

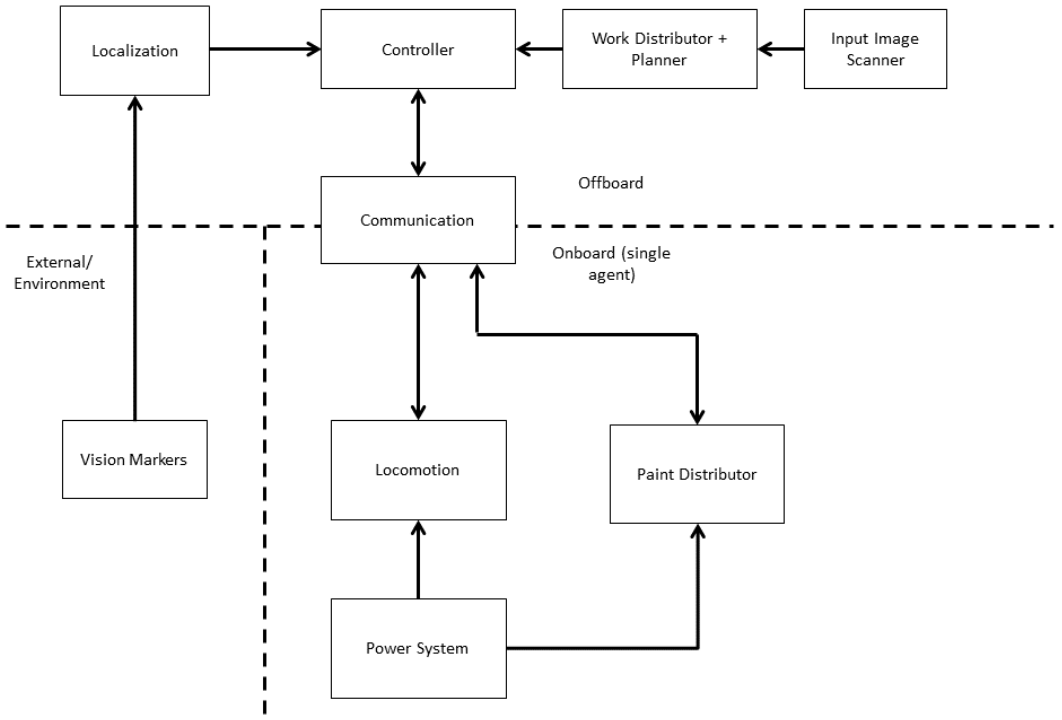

Figure 12: Software architecture model for offboard and onboard system.

6 Software Implementation Process

This section describes the software implementation of the robot system. Each software subsystem is described in detail, with control and temporal flow models as necessary.

Fig.12 represents the overall software system architecture. This flow chart describes the passing of information between modular software subsystems.

6.1 Software Libraries and Packages

- C++14
- Python 2.7 [**?**]
- OpenCV 2.4 [**?**]
- Protocol Buffers 3 [**?**]
- AprilTags C++ Library [**?**]
- Python RPi.GPIO 0.6.3 [**?**]
- Boost.Python 1.60 [**?**]
- Python Tkinter [**?**]

6.2 Full System

Full system architecture is split into offboard and onboard, and handles the coordination between the various subsystem software. This section will describe the controller for the offboard and onboard systems separately.

Algorithm 1 Full System

```
Given: Input Image I
planner_input = ImageProcessing.processInputImage(I)
paths = Planner.planRobotTrajectories(planner_input)
Communication.connectToRobots
procedure !Planner.isFinishedDrawing
    robot_info = Communication.getTCPMessage(robots)
    localization = Localization.localizeRobots(camera_data, robot_info.odometry)
    paths = Planner.updatePaths(localization)
    locomotion_message = Locomotion.generateCommand(localization, paths)
    writing_message = WritingImplement.generateCommand(localization, paths)
    error_message = Communication.processErrors
    Communication.sendMessage(locomotion_message, writing_message, error_message)
    UI.displayErrors(error_message)
```

6.2.1 Offboard Controller

The offboard controller's primary function is to continually send updated locomotion commands to the robot agents. This controller will operate based on the pseudocode listed below. Functions are classified by the subsystem they are a part of (Planner refers to the planning subsystem Sec. 6.7, Locomotion refers to the locomotion subsystem Sec. 6.4, etc.). This pseudocode acts as a temporal model that describes offboard control flow.

It is important to note that while locomotion, writing implement, and error-related messages are all sent together, any action based on error reports executes with the highest priority on the onboard robot controllers.

The main loop will run continuously at a fixed rate until system operation is finished and the drawing is complete. In order to ensure a fixed-rate operation, large processing steps - such as updating paths, updating the UI, and localizing - will be done concurrently in separate threads. This will ensure the main controller is free to send any error messages immediately, or send an emergency kill command on user input.

6.2.2 Onboard Robot Controller

The onboard robot controller is responsible for parsing incoming commands, and sending odometry, writing implement, and other logging information back to the offboard system.

The robot will parse commands as it recieves them, and use them to actuate motors for locomotion or raising and lowering the writing implement. It will also parse error information, for messages such as emergency stop, pause, or other noncritical errors. Error processing and action occurs with a higher priority than logging and motor actuation.

In order to send logging information in a way that is easiest for the offboard controller to process, the onboard system will only send logging information when requested by the offboard controller. Log information includes motor encoder data for odometry, battery state, and any debugging information.

The onboard robot controller will operate based on the following pseudocode. Functions are classified by the subsystem they refer to, similar to Sec. 6.2.1. The pseudocode below describes the temporal model of onboard robot controller operation.

6.3 Writing Implement

Writing implement software is run both on the offboard system as well as on individual robot agents. This software will be a part of the main control system on the offboard system, written in Python [**?**]. Onboard, it will be parsed from the communication system Sec. 6.8 and used to command the writing implement.

The offboard system will combine localization and planning data to determine when the writing implement should be raised or lowered. Once the controller decides the state of the implement, it uses logging information from the agent to determine how far up or down the implement must be moved. This delta is computed as a part of the proto3 message packets [**?**] and sent to the agent for command.

Onboard a robot agent, the robot controller is responsible for sending the current state of the writing implement back to the offboard system. This is sent using proto3 messages [**?**] as a part of the logging

Algorithm 2 Offboard Processing

```
1: Communication.connectToOffboard()
2: procedure TRUE
3:     message = COMMUNICATION.GETTCPMESSAGE(offboard)
4:     locomotion = COMMUNICATION.PARSELOCOMOTIONMESSAGE(message)
5:     writing = COMMUNICATION.PARSEWRITINGMESSAGE(message)
6:     errors = COMMUNICATION.PARSEERRORMESSAGE(message)
7:     if errors.emergencyOff then
8:         LOCOMOTION.STOPOPERATION
9:         WRITING.STOPOPERATION
10:    if errors.pause then
11:        LOCOMOTION.PAUSEOPERATION
12:        WRITING.PAUSEOPERATION
13:    LOCOMOTION.PARSEERRORS(errors)
14:    WRITING.PARSEERRORS(errors)
15:    LOCOMOTION.ACTUATEMOTORS
16:    WRITING.ACTUATEMOTORS
17:    if message.requestsLogging then
18:        COMMUNICATION.SENDMESSAGE(Power.batteryLevel, Locomotion.encoders, Writing.encoders)
```

information. When the agent receives a message, it parses the writing implement command, and then commands the writing implement motor appropriately. This part of the onboard controller is written in Python [**?**], using the RPi GPIO Library for interfacing [**?**].

6.4 Locomotion

The software involved in locomotion exists both on the offboard processor as well as on individual robot agents. Robot agents must process commands sent from Sec. 6.8 and command the motors. The offboard processor must use localization from Sec. 6.5 and planning data from Sec. 6.7 to determine the current motor commands.

6.4.1 Libraries

Locomotion software makes use of Python 2.7 scripts [**?**] for commanding motors via Raspberry Pi [**?**] and generating proto3 commands offboard by combining localization and planning

6.4.2 Robot Agent

Robot agents read proto3 messages coming from the TCP connection, as specified in Sec. 6.8. These messages contain locomotion commands in the form of specific motor velocitoes; these motor values are relative to each other in value and together, will command the robot in the required direction. The robot parses this proto3 message, and commands each of the four motors via the Raspberry Pi GPIO pins [**?**]

6.4.3 Offboard Controller

The offboard controller is in charge of combining localization and planning information to create correct locomotion commands for each robot agent. The system must take into account the robot agent's current position and orientation, and then generate locomotion commands that will move the robot along the specified path. Python scrips will generate locomotion commands [**?**].

Omnidirectional motion with mecanum wheels must have a specific controller, as directional movement is not as straightforward as with traditionally-wheeled robots. Controllers already exist, and reduce the problem to selecting a robot velocity, rotational velocity, and desired angle to be rotated to [**?**]. Localization provides information about the robot's current position and orientation. The path planning algorithm can be used in conjunction with localization to determine the next expected location and orientation of the robot. Given a current robot position and expected, the delta can be computed to represent the position and angle to be moved. Using speed constraints that align with NFR 6, velocity

commands for each motor can be computed. These are then added the the proto3 message, and sent to a robot for execution.

6.5 Localization

Localization uses data gathered from real-time overhead-mounted camera data to localize the drawing area bounds, and robot positions and orientations. Important locations are marked by AprilTags. AprilTags are the equivalent of 2D barcodes, which can be calibrated and detected at range to determine their orientation and position [**?**]. Localization software will use the AprilTags C++ library. In order to provide compatibility with the Python scripts used for locomotion and control Sec. 6.4, Boost Python will be used to wrap C++ functions into the Python layer [**?**].

Four AprilTags placed at the four corners will represent the drawing boundary, which is static throughout the course of a single drawing. During the localization loop, these will be found and will remain the same at every iteration. By detecting all four corners, the systen can find the maximum and minimum coordinates of tags in camera image space. This will then define the bounds for computing locomotion commands. For the bounds, the orientation of the tags make no difference.

In contrast, tracking the robot agents requires both position and orientation tracking. Positions will be reported relative to the bounds, with the bottom left corner bounds marker being designated as (0,0). Similarly, orientations will be reported as an angle relative to the base of the detected bounds.

6.6 Image Processing

Image processing involves the user inputting a series of lines, and converting it into a format usable by the planner. The planner takes an ordered series of lines, and the image processing subsystem will perform this task. This system will create the ordered lines using Python [**?**].

6.7 Work Scheduling, Distribution and Planning

Our scheduling and planning process, as detailed in Algorithm 1, happens entirely offline. Thus it is the task of the controller to monitor the trajectory execution. In Algorithm 3 we decompose planning those trajectory into: distributing the work between the two robots, planning each path independently and coordinating the trajectories to avoid collision.

Algorithm 3 Planner.planRobotTrajectories

1: **Given:** Set of Line Coordinates L
2: $\{L_{R0}, L_{R1}\}$ = SCHEDULER.DISTRIBUTEWORK(L)
3: P_{R0} = PLANNER.GENERATEPLAN(L_{R0})
4: P_{R1} = PLANNER.GENERATEPLAN(L_{R1})
5: $\{T_{R0}, T_{R1}\}$ = PLANNER.GENERATETRAJECTORIES(P_{R0}, P_{R1})
6: **return** $\{T_{R0}, T_{R1}\}$

The work distribution algorithm is given in Algorithm 4. The idea is to spatially segregate the lines such that one robot primarily operates on the left and the other operates on the right. By alternating assignment, rather than assigning down the middle, we hope to better balance the distribution.

Algorithm 4 Scheduler.DistributeWork

1: **Given:** Set of Line Coordinates L, Number of lines n
2: **Initialize:** Line Sets $L_0 = L_1 = \emptyset$
3: **procedure** FOR I=0:$\frac{n}{2}$
4: L_0 += SCHEDULER.FINDNEXTLEFTMOST(L)
5: L_1 += SCHEDULER.FINDNEXTRIGHTMOST(L)
6: **return** $\{L_0, L_1\}$

Given a robot's set of lines, we next need to plan a path that traverse through all the lines, given in Algorithm 5. We think of the overall plan as a series of segments, planning from where we left off to the start of the next line, then planning along the line. We define a home pose H for each robot where it starts and ends. To minimize collision we start each robot from opposite sides of the drawing surface.

Algorithm 5 Planner.generatePlan

Given: Set of Line Coordinates L, Number of lines n, Home Pose H
Initialize: Reference Path $P = H$
procedure FOR I=0:N
    P += PLANNER.GENERATECONTROLS(P.end, L[i].start)
    P += PLANNER.GENERATECONTROLS(L[i].start, L[i].end)
P += PLANNER.GENERATECONTROLS(P.end, H)
return P

We generate a reference trajectory for each robot in Algorithm 6. We generate uniformly timed straight line "Snap" trajectories. At each time step, we check for collision and simply insert a waiting pause to prevent the robots from colliding. Hence this is equivalent to one robot waiting at a invisible stop sign, allowing the other robot to pass. To allow a rough notion of fairness we alternate which robot should pause.

Algorithm 6 Planner.generateTrajectories

Given: Plan P_0, P_1
Initialize: Traj $T_0 = T_1 = \emptyset$, Pause Traj $T_p = T_0$
Total Time T= MAX(P_0, P_1)
procedure FOR I=0:(T-1)
    if PLANNER.CHECKCOLLISION($P_0[i], P_1[i]$) **then**
        PLANNER.INSERTPAUSE(T_p)
        T_p = SWAP(T_0, T_1)
    T_0 += PLANNER.SNAPTRAJ($P_0[i], P_0[i+1]$)
    T_1 += PLANNER.SNAPTRAJ($P_1[i], P_1[i+1]$)
return $\{T_0, T_1\}$

6.8 Communication

6.8.1 Libraries & Protocols

The communication software subsystem handles sending information to and from the robot agents and the offboard system. The offboard system will be tasked with mainly with sending locomotion and emergency commands to the robot agents. The robot agents will send logging information and odometry measured from motor actions back to the offboard system for processing. The offboard system can use received data to determine if a robot agent has fallen into any error states. Fig.13 is a model describing the flow of information between the offboard and onboard system. This model is described in detail throughout this section.

Communication will be handled via a TCP connection between a robot agent and the offboard system. Given that the robots only receive and process commands relating to their own motion, there is no need for the robots to be able to directly communicate with each other.

Once a TCP connection is established, data will be sent using Protocol Buffers, a Google-designed standard for serializing structured data [**?**]. For this project, Protocol Buffers language version 3 (proto3) will be used due to its improved speed and features over the previous version, proto2.

Using proto3 will allow the offboard system to send commands in packets, called ḿessages.́ This way, the system can organize locomotion and emergency commands into separate messages or sections of a message, which can then easily be parsed by the robot agents.

6.8.2 Message Design

Message design will be addressed first with regard to messages sent to robot agents, then messages sent from robot agents to the offboard system.

All messages sent to the robot agents will involve emergency information commands or locomotion. Emergency commands can be separated into full system stop, and system pause. System stop is the emergency kill switch, which occurs by user command or when the system encounters a fatal error. This will end onboard robot operation. System pause occurs for errors that are recoverable; an example of this

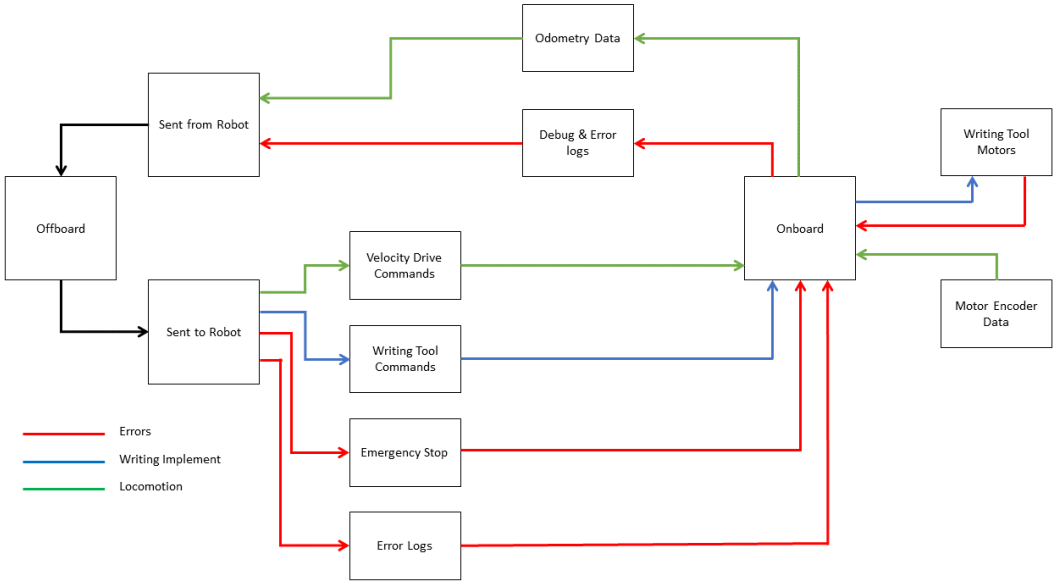

Figure 13: Model of system communication across controllers.

would be robot collision. The user has the option to assess and correct any issues, and resume operation. Both system stop and pause will halt all locomotion commands until the robot agent is commanded otherwise.

Locomotion commands will be continually sent to robot agents. These commands will come in the form of relative motor powers to command each of the four mecanum wheels. These messages will be sent continually, to allow for stable corrections to robot motion to maintain smooth lines, which increases the overall quality of drawing. Below is a table representing data fields that will go into the proto3 messages being sent to onboard robot controllers.

Message Name	Data Type
Emergency State	Stop, Pause, Resume
Wheel Motor 1	Motor power
Wheel Motor 2	Motor power
Wheel Motor 3	Motor power
Wheel Motor 4	Motor power
Writing Implement Motor	Motor power

Robot agents are responsible for sending odometry and logging information back to the offboard processor. Odometry information is sent for each of the four motors on the robot, by sending the encoder ticks since the last communication. This odometry information can be incorporated into localization. Motor encoder movement for the writing implement motor will also be sent. Logging information can be used for debugging or system analysis. An example of logging information is the robot battery level. Below is a table representing data fields that will go into the proto3 messages being sent back to the offboard master controller.

Message Name	Data Type
Wheel 1 Odometry	Encoder value
Wheel 2 Odometry	Encoder value
Wheel 3 Odometry	Encoder value
Wheel 4 Odometry	Encoder value
Writing Implement Motor Odometry	Encoder value
Battery Level	Voltage from battery

6.9 User Interface

The user interface is a graphical interface that provides the user the option to enter settings, start, pause, and stop drawing execution, and display error messages. The interface will be written in Python using the Tkinter library [**?**]. This library makes for a straightforward interface to set user options, as well as display any warnings or errors during the drawing process. The interface will be run concurrently with the offboard control software.

6.10 Power System

Power system software is in charge solely of checking the battery power level of robot agents. As a part of the communication pipeline, battery information is sent as a part of logging information back to the offboard system Sec. 6.8. The offboard system will compare current battery level to the maximum level, and will pause or stop system operation when the battery falls below a threshold. This will prevent the system from entering a state of undefined behavior when not enough current can be supplied. Power system control is handled by Python [**?**] in the main controller. This is done for simplicity, as proto3 messages are parsed in the same place Sec. 6.8.1.

7 Failure States and Recovery

7.1 Degraded Mode: Driving but Not Drawing

This failure state occurs when a robot agent is able to drive properly using localization and planning but the writing implement is not working properly. It can be caused by one of two factors: a broken actuation mechanism, or the writing material is out of ink. The writing implement running out of ink is significantly more likely, as it is a common occurrence and the system is designed around this limitation. For either degraded mode, the only solution is for the user to pause system operation, resolve the issue, and resume operation. The user can either replace the writing implement or replenish ink, or fix the actuation mechanism. This mode can be traced back to the following failure modes from the Test Plan: FM 4.1, FM 4.5.1, FM 4.6.1.

7.2 Degraded Mode: Erratic Driving or Drawing

Erratic driving or drawing occurs when the robot is moving, but not responding properly to the given commands. It can also occur when the commands being sent are incorrect, or when the drawing actuation commands or actuation is incorrect. Invalid motion can be attributed to the system failing to move omnidirectionally (likely due to issues with one or more mecanum wheels), or due to external factors such as slippage. Incorrect commands and undefined behavior can begin as a result of incorrect localization, loss of connection between offboard and onboard systems, battery problems, or the robot agent moving out of bounds. In these situations, the solution is to end system operation. Undefined behavior cannot be resolved without resetting the system. This mode can be traced back to the following failure modes from the Test Plan: FM 4.1.2, FM 4.2.1, FM 4.2.2, FM 4.3.2, FM 4.3.3, FM 4.6.2, FM 4.8.1.

7.3 Degraded Mode: Planning Failure

Planning failure is a degraded mode that occurs when the input image is unable to be processed into a path. When this happens, the planner is unable to generate a set of paths for the robot agents. The system cannot begin to draw the image without a valid path, so the user is required to enter a new image to begin system operation. This test plan can be traced back to the following failure modes from the Test Plan: FM 4.5.1.

8 Installation Plan

The system needs a large (at least 6ft by 6ft), flat indoor surface on which to draw. To ease cleanup, we will cover the working surface with blackboard paper. The corners of the workspace will be fixed with AprilTags. There also must be a ceiling above the workspace low enough to mount the webcam, but high enough to allow the webcam a full view of the workspace. This means that the ceiling must be between 7 ft and 15 ft above the floor.

We will also need a single outlet to plug in the power strip used for charging the central processing laptop and the battery chargers for the robot.

9 Traceability Matrix

Requirement	Section
FR 1	Sec. 4.2, Sec. 6.4.2, Sec. 6.4.3
FR 2	Sec. 6.5, Sec. 6.4, Sec. 6.7
FR 3	Sec. 4.3, Sec. 4.2
FR 4	Sec. 6.5, Sec. 6.7, Sec. 6.8, Sec. 8
FR 5	Sec. 4.1
FR 6	Sec. 4.1
FR 7	Sec. 4.1
FR 8	Sec. 4.6, Sec. 6.8
FR 9	Sec. 6.7, Sec. 6.4, Sec. 6.5
FR 10	Sec. 4.1, Sec. 6.3
FR 11	Sec. 6.9
FR 13	Sec. 4.7, Sec. 6.2,Sec. 6.8, Sec. 6.9
FR 14	Sec. 4.5, Sec. 6.9
FR 15	Sec. 4.6, Sec. 6.10
NFR 1	Sec. 4.7.1, Sec. 6.2, Sec. 6.1
NFR 2	Sec. 6.2, Sec. 6.8, Sec. 6.5, Sec. 7
NFR 3	Sec. 2.1
NFR 4	Sec. 2.1
NFR 5	Sec. 4.2, Sec. 6.7
NFR 6	Sec. 6.4, Sec. 6.7, Sec. 4.2, Sec. 6.3, Sec. 4.6
NFR 8	Sec. 4.1, Sec. 4.6, Sec. 4.2, Sec. 6.5 Sec. 6.8
NFR 9	Sec. 6.7, Sec. 6.8
NFR 10	Sec. 5
NFR 11	Sec. 4.7, Sec. 4.6 Sec. 6.4, Sec. 6.7, Sec. 6.8
NFR 12	Sec. 4.2, Sec. 4.3, Sec. 6.4, Sec. 6.5
NFR 13	Sec. 4.2, Sec. 4.3, Sec. 6.4, Sec. 6.5
NFR 14	Sec. 4.1, Sec. 4.7