Worcester Polytechnic Institute

Major Qualifying Project

Custom Autonomous Robotic Painter

Authors:
Katie Gandomi
Nick Panzarino
Odell Dotson
Alora Hillman

Advisors:
Professor Jie Fu
Professor Michael Gennert
Professor Kenneth Stafford

Submitted in partial fulfillment of the requirements for the degree of Bachelor of Science

April 26, 2017

Abstract

Today humans out perform robots in problem solving, adaptability, and creativity. The goal of this project was to bridge the gap between robotic and human capabilities through the development of an autonomous painting robot. The custom design of the mechanics, electronics, and software allowed for a versatile solution. Image decomposition techniques were used to break down input images into feature areas that were reconstructed by the robot. Vision feedback was also performed during the painting process to apply corrections to the artwork dynamically. Understanding the motions undertaken by painters and replicating it in a robotic platform can revolutionize the art form, contribute to the scientific advancement of robotic capabilities, and reduce the workload needed to construct paintings.

Acknowledgements

We would like to thank Automation Direct, Festo, Igus, and Performance Motion Devices for the donations made to our project. Without the engineering tools and materials supplied, the robot would not be where it is today.

We would also like to give a special thanks to our advisors Professors Jie Fu, Michael Gennert, and Ken Stafford for all their support during this project. Without their guidance, technical and non-technical, the project would not have been a success. Thanks as well to Katherine Crighton for her continued support across the duration of this project.

Thank you!

Contents

List of Figures

List of Tables

1 Introduction

In this chapter, the project's need, problem statement, and target audience are introduced.

1.1 Problem Statement

Advances in robotics and automation technology have reached a point where robotic solutions are better than humans at performing certain tasks. The advantages in speed, precision, repeatability and endurance provided by electromechanical systems make them well suited for simple, repetitive tasks. Humans however, still have major advantages over robots in terms of problem solving, adaptability and creativity. This means that humans can typically outperform robots in tasks which: have many unknowns, do not have a known solution, have many possible solutions, change over time, and/or require complex motions. Our project aims to bridge this gap between robot and human capabilities by demonstrating a robotic system capable of performing a task typically reserved for humans. The task chosen is painting. Painting is a difficult task for a robot for several reasons:

- There are many unknowns, such as the amount of paint on the brush, deformation of the bristles, the effects of mixing paint on the canvas, and even what the desired painting should look like.
- Painting is a form of artistic expression. It typically requires creativity, imagination and talent to produce high quality painting. These are human traits which cannot yet be duplicated by machines.
- Professional painters use various tools and techniques. To replicate such painters, the robot must be able to physically change between various tools and execute various actions, as well as be programmed with the ability to choose when to employ each.

By creating a robotic automation system which can perform a task such as painting, an advance in the capabilities of robots and automation is demonstrated, and providing evidence that it is possible for robots to perform other human tasks.

1.2 Project Justification

This robot can be used to create an artistic painting from a supplied photograph. There are two major groups of potential consumers that have been identified. The first group is people who want custom paintings. These paintings will be produced more cheaply than by professional artists. They can also be created simply from an input image, providing the consumer with a good understanding of what the final result may look like. Photographs of lost loved ones or childhood houses can thus be easily and cheaply turned into paintings.

The robot itself is not something that would be purchased by someone wanting a painting, though. The robot would likely be purchased by an art facility that would then charge consumers to contract the aforementioned paintings. These art facilities may also use the robot for research interests that may require multiple copies of the same painting. Selections of photographs paired with paintings created from the robot may also become something used in photography art installations.

Researchers could also use this robot as an open research platform, developing new modules or libraries for their own study. These may be as simple as libraries of brush strokes, or as complex as new styles of painting. The gantry is commercially available and the software is open source, which may draw researchers towards the platform. Existing robotic painting platforms lack the modularity and ease of transportation that this system has, making it ideal for multiple research installations.

2 Background Research

In this chapter, background research conducted on basic painting techniques are presented along with a literature review of prior robot painters.

2.1 Painting Techniques

2.1.1 Paints

When painting, there are a number of different strokes, brushes, mediums, and styles to take into consideration. The three most common paints are watercolors, oils, and acrylics. Each medium was reviewed for use in this robotic platform.

Watercolors are pigments mixed with gum that have been diluted with water. This creates a transparent medium that can be built up in multiple layers. They are the the least expensive of the three mediums because little pigment is required to cover a large area. However, watercolor is unforgiving to the canvas. First, spaces intended to be white are simply left without paint instead of being painted over. This is because the translucent nature of watercolors makes darker colors difficult to mask or lighten once they have been applied. Second, watercolor paintings can be easily ruined with a single drop of water due to the fact that watercolors do not permanently dry.

Oil paints are pigments that have been combined with oils to create a thick painting medium. Oil paintings are known for their ability to show light. They have a luminescent quality because the oil allows light to be reflected back to the viewer. The major drawbacks of oil paints are that they are more expensive than other paints and they take a very long time to dry. Often thick applications of oil paints can take over six months to dry completely.

Acrylics are water-based paints that combine pigments with synthetic resins to create a very versatile medium. Concentrated acrylics paints can have similar properties to oil paints because they are completely opaque and can be layered to create depth. Acrylics can also be diluted with water in order to mimic the transparencies of watercolor paints. The main benefit of acrylics is its fast drying time. However, the largest drawback of this medium is that dried acrylic can be difficult to remove from objects.

2.1.2 Painting Styles

During this project, two painting styles were focused on: realism and impressionism. These styles were chosen because realism depicts the subject accurately while impressionism uses loose brush strokes. A combination of these two styles achieves

the project goals of accurately depicting the subject while still creating a painting made up of visible brush strokes.

Realism focuses on achieving a work of art that accurately depicts the subject matter without any stylization. Often this style is used to depict mundane activities or ugly subject matters in uncomfortably detailed works of art. Realism artists typically use dark, earthy palettes that directly conflict with the Romanticism movement's ideals of beauty.

Impressionism focuses more on the overall feeling of a work of art rather than its details. Loose brush strokes are used along with intense colors in order to convey the passage of time and the effects of light. In impressionist artworks, the brush strokes are more important than the lines and contours of the subject. The short brush strokes used are intentionally not blended together. This gives impressionist paintings an unfinished look. Similar to realism, impressionism typically captures scenes from everyday life instead of grandiose, fictitious scenes.

2.1.3 Brushes and Strokes

The brush stroke that can be made is determined by the shape of the brush used. Figure 1 shows a graphic of the most common brushes used and their resultant strokes. For the painting styles being pursued in this project a small stroke is necessary. This is best obtained by using a round brush. Additionally, a round brush simplifies the painting experience by being able to make the same stroke regardless of what direction it is oriented.

While other brushes such as a flat brush would be able to apply two distinctly different strokes depending on the orientation of the bristles, the round brush is able to provide the necessary variety in brush stroke for these painting styles. The variety can be found in control of the vertical distance the brush is applied to the paper. If the brush is held above the paper with only a few bristles on the canvas, a thin stroke is created. This is distinctly different from brush strokes made by decreasing the distance between the brush and the paper, thereby causing a thicker mark.

Figure 1: Brush stroke styles [3]

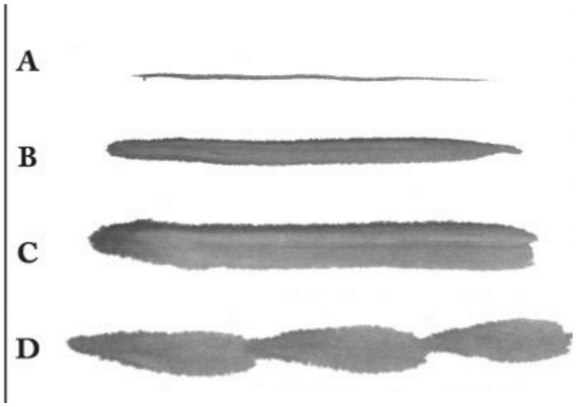

Figure 2: Round brush stroke styles [2]

In order to vary the thickness of each brush stroke, the z-distance of the paintbrush from the canvas needs to be carefully considered. When painting, this distance is measured using the built in potentiometer in the z-axis actuator.

2.2 Literature Review

Some important artistic robots and their contributions are highlighted below. Careful considerations to the problems each robot encountered were made while designing CARP.

2.2.1 eDavid

The eDavid project began in 2009 with the goal of creating artworks that mimic the human painting process. The robot uses a serial manipulator, and various sensors to accurately reproduce input images as paintings. The eDavid incorporates several techniques to achieve this functionality: iterative painting through camera-based vision feedback, predefined strokes, and dynamically generated strokes. By utilizing these techniques, the eDavid can produce detailed ink sketches and both binary and color paintings.

2.2.2 VangoBot

VangoBot was created by Doug Marx and Luke Kelly in 2008. This robot uses a gantry system that is capable of switching between 16 brushes and mixing its own paint from 8 canisters. The robot takes in a desired image that can be edited by the artist and replicates the product onto a canvas. Additionally, VangoBot is capable of taking suggestions and painting an image based on its own interpretation. One of the limitations with this design, however, is that the robot does not use any kind of vision feedback.

2.2.3 Painter Robot: Painting by Force and Vision Feedback

This robot was built in order to emulate human painting behavior as accurately as possible. The end effector is a multi-finger hand built up of the four fingers that humans use to paint. Three of the fingers have three degrees of freedom while the fourth has four degrees of freedom. Each finger has a force sensor at its tip in order to determine the force it is applying to the paintbrush handle and the paper. The system uses stereo vision with nine cameras in order to determine the position of the paintbrush and the strokes being made on the paper. The position of the tip of the brush is determined by detecting the position of the handle and extrapolating. Contact with the paper is sensed as a sudden increase in the force sensor outputs. The robot controls the brush strokes made by manipulating the angle of the brush, the amount of pressure being applied to the page, and the path of the brush. After creating a painting, the robot uses vision to determine what corrections need to be made in order to more accurately depict its subject.

2.3 Evaluating the Design

The design proposed in this project can be evaluated based on flexibility, extensibility, dexterity, complexity, and intellectual challenge. This section will go over each of these in detail and how they compare with previous solutions.

Flexibility

The overall modular construction of the custom robotic platform designed in this project allows for a lot of flexibility in the software, mechanical, and electrical systems. For instance the gantry used in this project can have its height adjusted, the camera used can be substituted for another model, and end effectors can be exchanged with ease. Finally, the software and control algorithms used in this robot are also easily interchangeable.

Extensibility

In addition to the platform being flexible in its hardware, electronics, and software, it is also easily extensible. Its compact form factor and modular design allow for new systems to be easily integrated or developed on top of the existing robotic platform. With the inclusion of force and distance sensors into the design, the platform can also be easily modified to handle different mediums such as crayons or color pencils. Other designs for painting robots have not shown high extensibility. For example, VangoBot would need a substantial amount of modifications to accommodate other mediums while eDavid simply needs to change end effectors.

Dexterity

The design of this robot platform lacks the dexterity of some of the existing painting machines. For example, the eDavid, being a six degree of freedom robotic arm, is quite good at maneuvering and carrying out strokes similar to a human. On the other hand, in this project the robot will have a dexterity similar to VangoBot, with a gantry system capable of moving in the x-y-z axes. Future work on the project could involve the introduction of an additional degree of freedom on the end-effector to control angling of the paint brush.

Complexity

The hardware used on the robotic platform is simple and orients the paintbrush in the x-y-z axes. The electronics used are also fairly straight-forward making use of a few stepper motors, a performance motion drive controller, and terminal blocks. The simplistic nature of the hardware and electrical systems on this platform give

way to the more complex programming component. The software is capable of obtaining images from a camera, extracting features from an image, determining the appropriate stroke to use, relaying stroke information to the low level code, and finally having the low level code move the motors appropriately to execute the action.

Intellectual Challenge

Since tactile painting is not a solved problem in the robotics community, there are a number of intellectual challenges to address in this project. The central one being the development of a vision feedback system for robotic painting. The eDavid makes use of vision to correct its paintings, but its artworks are mostly comprised of small short strokes. In this project, the team tackled the complexities behind performing large sweeping strokes during the painting process while also being able to perform precise corrections.

2.3.1 Pairwise Comparison Chart

To analyze the different criteria of importance in the project a pair wise comparison chart was created. The higher the score for a given metric the more it was valued in our decision making during the design process. From the table depicted below reliability and artistic quality were the two highest scoring criteria while affordability and speed were the two lowest metrics.

	Speed	Modularity	Reliability	Artistic Quality	Robust	Affordability	Score
Speed		0	0	0	0.5	1	1.5
Modularity	1		0	0	1	1	3
Reliability	1	1		1	1	1	5
Artistic Quality	1	1	0		1	1	4
Robust	0.5	0	0	0		1	1.5
Affordability	0	0	0	0	0		0

Figure 3: Decision matrix

3 Project Overview

In this chapter, the project requirements are presented.

3.1 Project Goals and Objectives

The goals for this project can be divided into three categories: goals which must be achieved in order for the project to be considered successful, goals to strive for, which will be attempted if required goals have been completed, and reach goals, which will be attempted if all other goals have been met.

Goals that must be achieved

- Development of a custom research platform for robotic art
  - Ability to move a paintbrush in the x, y, and z directions
  - Ability to sense the location of the end effector
  - Ability to access vision feedback
  - Ability to change end effectors
  - Ability to experiment with different control algorithms
- Achieve moderate artistic performance
  - Ability to deconstruct a given picture into brush strokes
  - Final painting resembles the original reference picture
  - Has clearly defined brush strokes (not just dots)
  - Uses a monochrome color scheme

Goals to strive for

- Ability to correct paintings according to the reference picture
- Picture paints in gray-scale and/or multiple colors

Reach goals

- Ability to learn from previous experience
- Ability to mix new colors on the palette

4 Mechanical Design

In this chapter, the mechanical design of the robotic painting platform is described. This includes a review of the platform itself and design iterations.

4.1 Robot Platform

The mechanical requirements for this system are relatively simple: linear actuation in three axes, and the ability to store and retrieve paint. The first design consideration was the method of actuation to employ on the robot. Previous painting machines have almost exclusively implemented one of two designs: a gantry system or a robotic arm. The advantages and disadvantages of each system with respect to this project are listed below:

Gantry System Advantages

- Modular and Compact
- Simple system dynamics
- Inexpensive

Gantry System Disadvantages

- Needs additional degrees of freedom added in Z direction

Robotic Arm Advantages

- Emulates human movement better
- Additional degrees of freedom do not need to be added

Robotic Arm Disadvantages

- Expensive
- Hard to modify

One of the main goals of this project was to make the robot a research platform for future use. With this in mind, the gantry system was chosen because it is a modular and compact system.

An H-style gantry was donated from Festo for use in this project, and it provides two of the three required degrees of freedom on the robot. In order to support the gantry above the work area, a frame was designed and built out of 80/20, a sturdy and modular material. The CAD model of the gantry and frame system is shown in Figure 4. The 80/20 frame allows for the height of the gantry to be adjusted easily. This creates a more useful research platform where different end effectors can be easily tested by adjusting the distance of the gantry from the work

space. The entire frame was attached to a wooden base to secure it and provide a platform to move the robot.

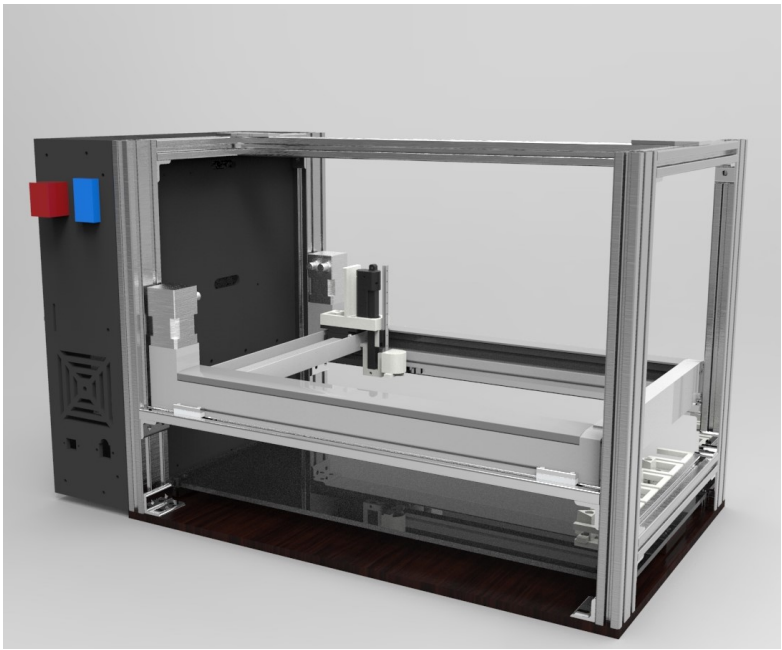

Figure 4: CAD model for the frame and gantry system

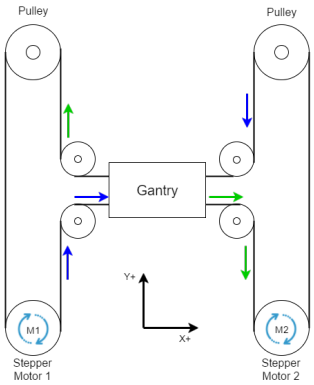

Figure 5: H-style gantry operation

The gantry operates through the use of two stepper motors attached to a single belt. Figure 5 illustrates the operation of an H-style gantry. Because of

this system, when only one motor is powered, the gantry moves at a 45° angle with respect to either the X or Y axis. The inverse kinematics for this system is explained in Section 6.1.

An additional degree of freedom was needed to actuate the paintbrush in the Z direction. Several options were considered for this mechanism: a ball screw, a lead screw, and a linear actuator. A ball screw would provide smooth and accurate movement but would be expensive and bulky. A lead screw would be less expensive but also has the problem of being too bulky for this system. Instead a micro linear actuator was chosen because it is lightweight, compact, easy to control, has built-in feedback, and is relatively inexpensive. The specific linear actuator chosen is shown in Figure 6.

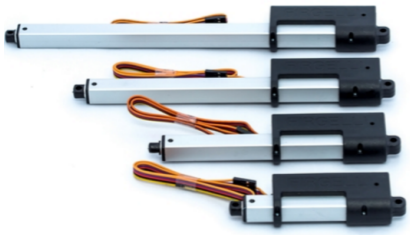

Figure 6: Firgelli linear actuator

4.2 End Effector Design

4.2.1 End Effector Design 1: Prototyping

For the first iteration of the end effector design, the following three parts were created to attach the paintbrush to the linear actuator. They were designed to be 3D printed in order to be manufactured quickly and inexpensively.

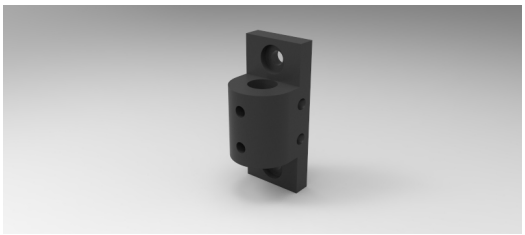

Figure 7: First iteration of paintbrush holder

Figure 7 shows the part created to hold the paintbrush. In this design, two screws applied pressure to the brush in order to hold it in place. This method allowed for the use of different sized paintbrushes. The problem with this approach was that the screws were not able to hold the brush in place when extreme force was applied to the paintbrush.

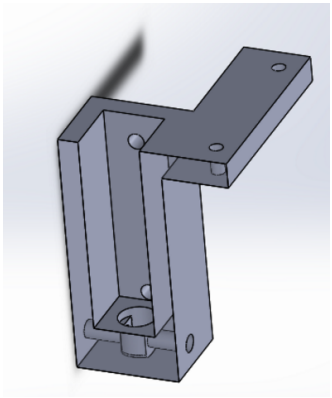

Figure 8: First iteration of Firgelli attachment to end effector

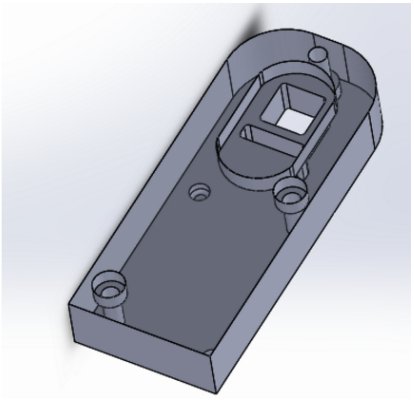

Figure 9: First iteration of Firgelli mounting

Figure 8 shows the method of attaching the Firgelli to the paintbrush holder. Because the Firgelli did not have any robust methods of attachment, it was pressure fitted into the bottom hole of this component and kept in place with a single screw.

Figure 9 shows the method of attaching the Firgelli to the gantry. Two screws were used to attach the component to the gantry and the top of the Firgelli was pressure fitted into this attachment.

There were several problems with the first iteration of end effector design. First, the paintbrush holder did not have the ability to change paintbrushes autonomously during operation. Second, throughout the painting process, it was found that the size of the brush holder was too large for the typical brush handle being used. Third, the end effector was not exact in its movement because the attachment to the Firgelli was held through a single bolt that was prone to rotating about its axis causing a significant amount of inconsistency to the system.

4.2.2 End Effector Design 2: Towards Improved Robustness

Figure 10 shows the second design of the end effector in action. The brush holder was redesigned to be smaller in order to hold the smaller brushes with ease.

Figure 10: Second iteration of design in action

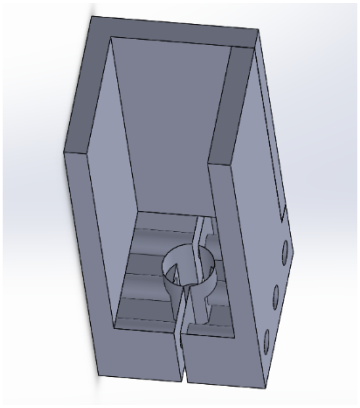

Figure 11: Second iteration of paintbrush attachment to Firgelli

The bottom of the end effector was redesigned to be two parts that were bolted together to reduce the slack in the system (Figure 11). This iteration of design fixed most of the problems of the first iteration but did not incorporate the IR sensor or autonomous brush changing.

4.2.3 Final End Effector Design: Tool Changing

During the design phase, the main goals for a tool changing system was to have a passive, reliable system. The passive nature of the system is beneficial because no additional motors or actuators need to be added to the system. The reliable nature of the system is important so that the robot can operate autonomously without failure.

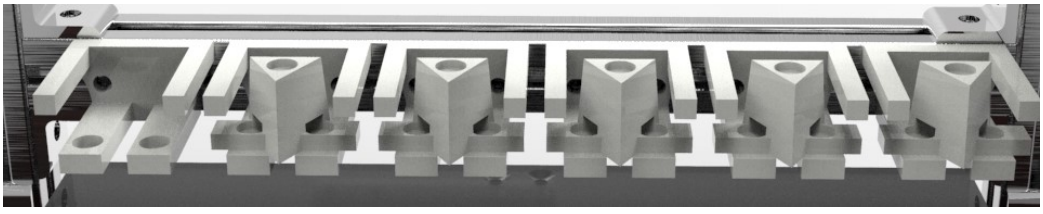

Figure 12: Tool changing dock

Figure 12 shows the final design of the tool changing docks and exchangeable end effectors. Six changing docks were fitted onto the robot to allow for a painting of six different colors.

Exchangeable Brush Holders
Each exchangeable brush holder had three embedded magnets that facilitated the exchanging of brushes. Additionally, their truncated pyramid shape ensured that the process of picking up or putting down a brush did not have to be exact. The magnets and pyramid shape corrected for any slack in the system by directing the brush holder into the end effector according to Figure 13. Additionally, the brush holders returned the tool to a set position in the dock through the use of magnets, as shown in Figure 14.

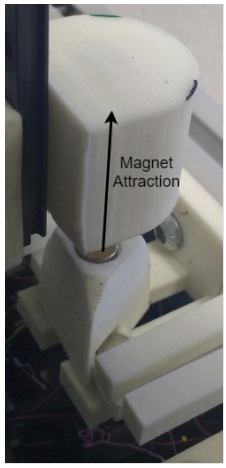

Figure 13: Paintbrush holder being picked up

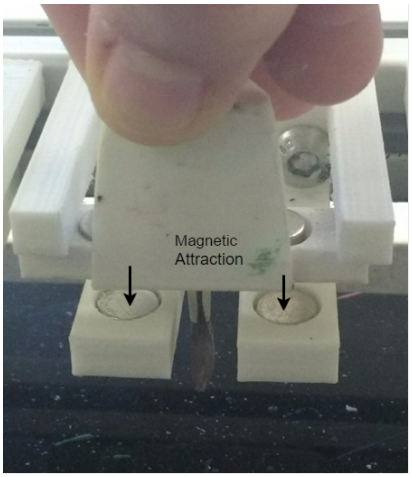

Figure 14: Paintbrush holder being put down

Paint Wells
Wells were created to provide a permanent location for the robot to receive paint. While the location of each paint wells was permanent, the wells themselves were purposely designed to be disposable. This enables the user to mix colors outside of the frame of the robot and insert them in the wells before painting. This also means that the wells can be disposed of after each use instead of requiring washing. Figure 15 shows the paint wells with four paints in disposable cups ready to be used by the robot.

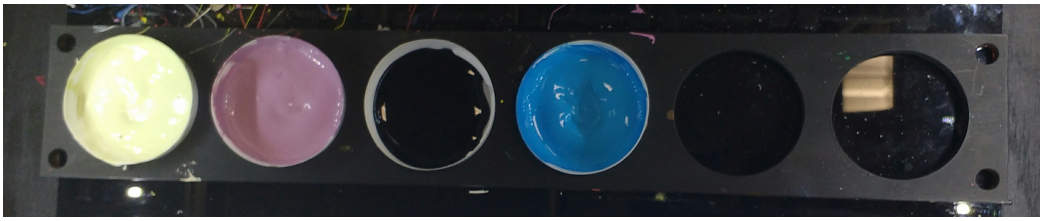

Figure 15: Paint wells with four paint colors

4.2.4 Rail System

One problem encountered with the addition of the Firgelli linear actuator was the accuracy of the end effector. Because of the long extension of Firgelli, a significant amount of slack was introduced into the system. A rail was integrated into the end effector in order to make the Firgelli more robust. There were two configurations that the rail could have been attached in shown in Figures 16 and 17. The first configuration was chosen because it did not leave the rail in the downwards position when the Firgelli was moved into the upper position. This allowed the gantry to be lower over the canvas area and reduced the space taken up by the robot.

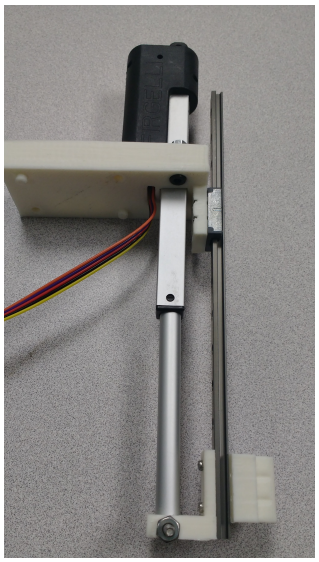

Figure 16: Configuration where the Firgelli controls the rail.

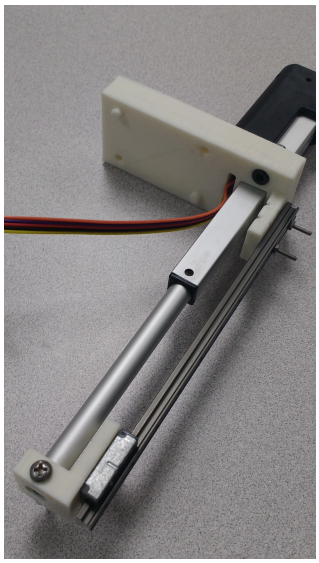

Figure 17: Configuration where the Firgelli controls the carriage.

4.3 Electronics Enclosure Heating Calculations

In order to make the electrical components more compact and professional, an enclosure was designed. Before construction, careful calculations were performed in order to ensure that the electronics would not overheat within the housing. The maximum heat loss for each component was measured and the change in temperature within the electronics box was calculated. For more information about the electrical enclosure see Section 5.2.

4.3.1 Power and Heat Loss Measurements

Table 1: Heat lost by electrical components

Electrical Component	**Input Power (W)**	**Output Power (W)**	**Heat Lost (W)**
Power Supply	231	192	39
48V to 12V Converter	33.6	21.6	12
PMD Control Board	67.2	0	67.2
Arduino Uno	0.2	0	0.2

Total heat released from electronics: 118.4W

4.3.2 Calculations

A simplified diagram of the heating elements within the electronics box is shown in Figure 18. The red area represents the electrical components, the blue area represents the air inside the box, and the black area represents the acrylic casing of the enclosure. Each variable is defined within the equation it is used.

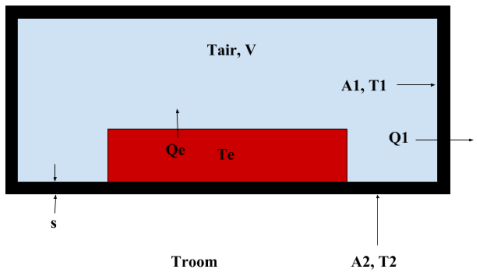

Figure 18: Electronics box with variable labels

$$Q - Q_1 = Q_e \tag{1}$$

Q = total heat released from the electronics
Q_e = amount of heat used to increase the air temperature inside the box
Q_1 = heat lost from the electronics box

$$Q_e = mc(\triangle T) \tag{2}$$

c = specific heat of air
m = mass of air in box
$\triangle T$= change in temperature of the air due to the heat released

$$Q_1 = h_c A_1(\triangle T) \tag{3}$$

h_c = heat transfer coefficient of acrylic
A_1 = internal surface area of the box
$\triangle T$= difference in temperature between the air in the box and the acrylic

$$Q - h_1 A_1(\triangle T) = mc(\triangle T) \tag{4}$$

$$\triangle T = \frac{Q}{h_c A_1 + mc} \tag{5}$$

Knowns:
$Q = 118.4\ W$
$m = 0.0245\ kg$
$c = 1.005\ kj/kgK = 1005\ J/kgK$
$h_c = 10.45\ W/m^2K$
$A_1 = 0.543m^2$

Calculations:

$\triangle T = \frac{118.4}{(10.45)(0.543)+(0.0245)(1005)}$

$\triangle T = 3.91K$

A change in temperature of 4 °C is well within the operating and storage ranges of all of the electrical components. Additionally vents were added to the sides and top of the electrical enclosure to provide additional airflow.

5 Electrical Design

In this chapter, the design of the electronics and the enclosure created are described .

5.1 Sensing on the Robot Platform

5.1.1 Camera and LEDs

A camera was mounted to the top of the frame in order to give visual feedback on the workspace. The image outputted by the camera was manipulated in order to detect the region of interest and ignore the rest of the workspace.

The lighting of the workspace provided some problems when it came to edge detection. Poorly lit areas of the workspace led to the robot not detecting the edge of the paper properly. The shadow of the gantry also interfered with the painting being recorded. In order to resolve these problems, LEDs were attached to the underside of the gantry to brighten the workspace.

The original camera did not capture the entire work space in its limited 60 degree field of view. To resolve this, a new camera with a 90 degree field of view was purchased. Because of the wider fish-eye lens, the dewarping transform was more dramatic for the new camera.

5.1.2 Infrared Distance Sensor

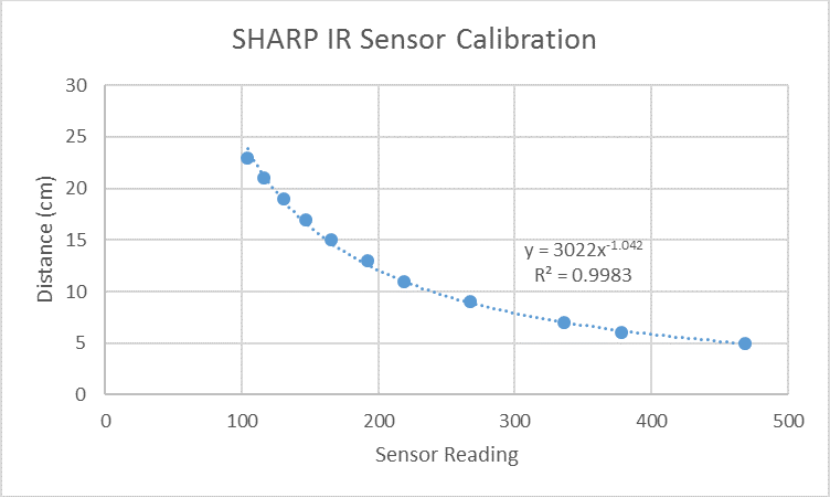

Figure 19: Sharp IR sensor calibrations

In order to use the Sharp IR sensor readings were taken at incremental distances and then curve fitted with an exponentially decaying equation (Figure 19). The R^2 value was very close to 1 which indicates that the equation fit has little error. Equation 6 was used to map the sensor readings to the distance the sensor was from the paper.

$$Distance = 3022(SensorReading)^{-1.042} \tag{6}$$

5.1.3 Force Sensor

In order to determine the force range required for the system, the paintbrushes were tested to determine how much force could be applied when their bristles were pushed down to a quarter, half, and three-quarters of their initial length.

The following paintbrushes were tested and the forces that were applied to them were recorded based on their brush number (Figure 20).

Figure 20: Test brushes

The range of forces recorded were from 0.392 N to 10.78 N (Figure 21). The useful range for the paintbrushes being used (numbers 1 through 4) was only 0.392 N to 2.94 N.

From this experimentation, it was determined that a force sensor for this system would only need to be able to sense up to 3 N of force. However, in the event of a failure, where the robot pushes the paintbrush into the paper past its expected position, it would be useful for the force sensor to detect this excess and indicate the failure to the system.

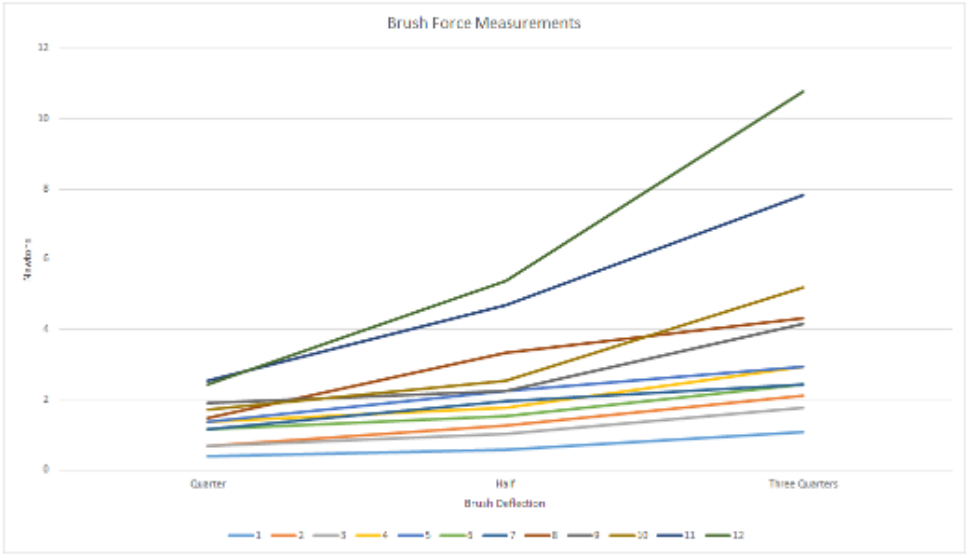

Figure 21: The force being applied by each brush

From this information, the TAL220 load sensor was chosen for the force plate on the robot, see (Figure 22). It can sense the required range of forces (up to 98N) and has a combined error range of ± 0.5 N.

The load sensor was then connected to the HX711 load cell amplifier in order to be able to translate the output strain of the load cell into readable values (Figure 23).

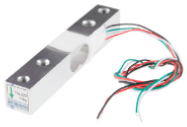

Figure 22: TAL220 load sensors

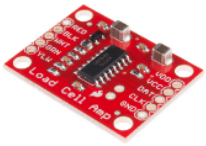

Figure 23: HX711 load cell amplifier

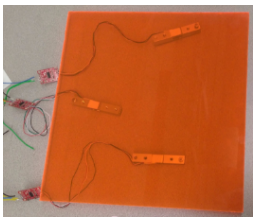

Figure 24: Test Force plate

Three of these load cells were wired in order to hold up a plate of acrylic (Figure 24). After being mounted on the edges of the acrylic, the load cells were summed to find the total force being applied to the plate.

5.2 Electrical Enclosure

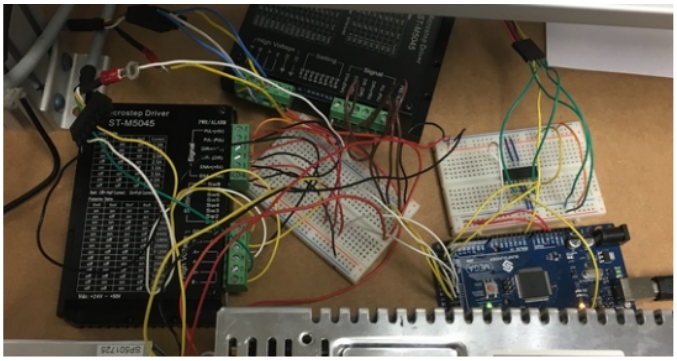

Figure 25: Original electronics setup

The original electronics setup shown in Figure 25 was reorganized to fit within a compact box. The goal of this box was to make the robot more professional and portable. Figure 26 shows the inside of the final box where the electronics were made to fit compactly together. Figure 27 shows the front side of the electronics box. An emergency stop button was mounted to the outside of the box to ensure safe operation for the user. Additionally, pause and resume buttons were mounted and implemented in software. Ports for the camera and Ethernet connection to the robot were mounted to the outside of the box. These additions ensured that the user did not need to open the electronics box except in the event of a failure.

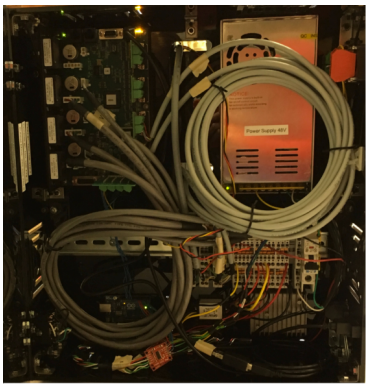

Figure 26: Inside of electronics box

Figure 27: Side of electronics box containing buttons and ports

Figure 28 shows all of the electronic connections within the electronics box and their interactions with the robot.

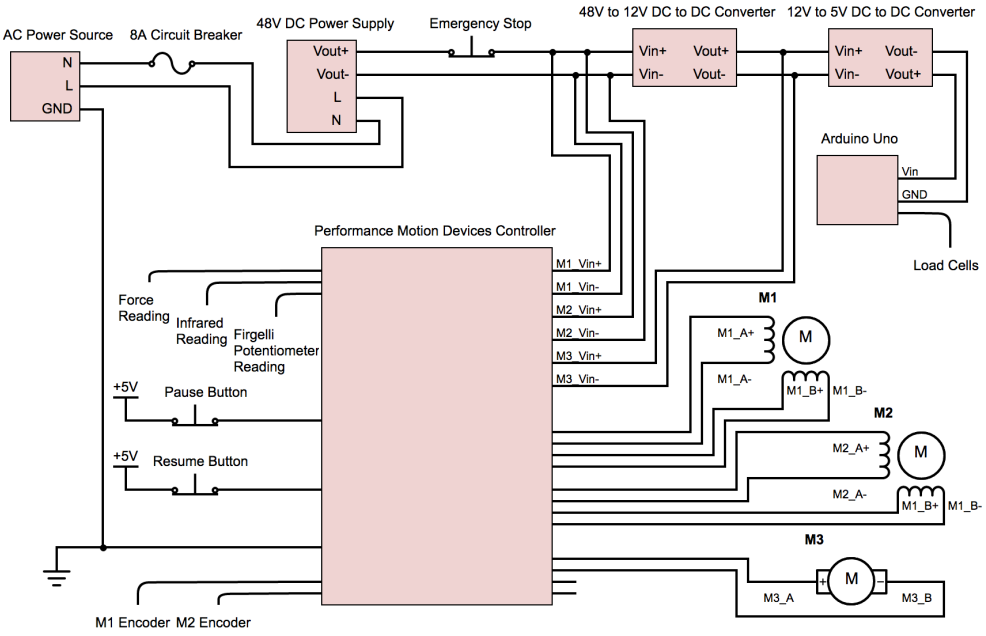

Figure 28: Basic schematic of the electrical enclosure

6 Low Level Software Design

In this chapter, the design of the low level software as well as the performance motion devices controller that runs it are described .

6.1 Low Level Software Functions

The software design of this project can be separated into two major components: High Level and Low Level (as seen in Figure 29). High Level code is written in python and can run on any standard computer. This code performs high level functions such as user interface, image processing and workspace path planning. Low Level code is written in functional C and is executed on the Machine Control Board donated by Performance Motion Devices (PMD). This code performs low level functions such as inverse kinematics, joint space trajectory planning, reading sensors, and closed loop motor control. Information is transferred between the two levels via Ethernet connection (TCP/IP protocol). The exact format of these packets is described in Table 2.

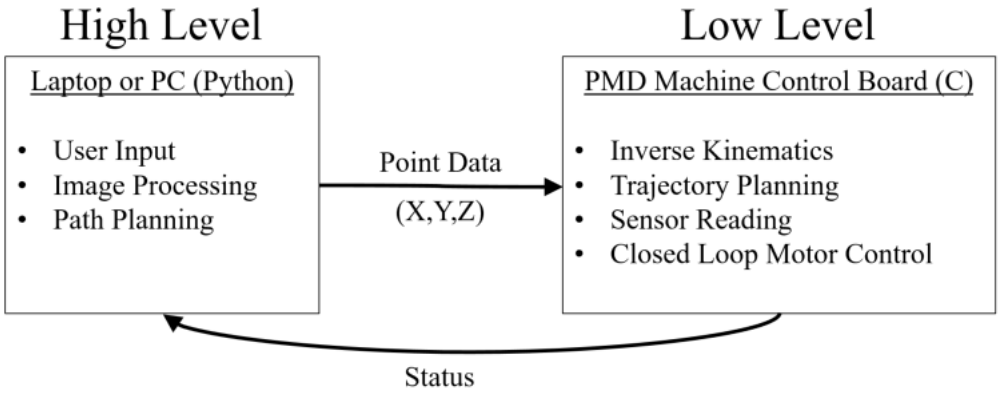

Figure 29: Software Design Overview

Table 2: Packet Description

Byte	Description
0	X position (high byte) (0.1 mm)
1	X position (low byte) (0.1 mm)
2	Y position (high byte) (0.1 mm)
3	Y position (low byte) (0.1 mm)
4	Z position (high byte) (0.1 mm)
5	Z position (low byte) (0.1 mm)
6	Minimum Step Time (high byte) (us)
7	Minimum Step Time (low byte) (us)
8	Bitmask of Flags: 1. Go flag: Indicates that the robot should execute all saved commands 2. XY abs flag: indicates that the xy motion in this command should be absolute, rather than relative 3. Z abs flag: indicates that the z motion in this command should be absolute, rather than relative 4. *Not Used* 5. *Not Used* 6. *Not Used* 7. *Not Used* 8. *Not Used*

The low level controller performs the following functions:

1. Inverse Kinematics: When packets are received, the low level controller performs inverse kinematics to convert points from workspace coordinates to joint space coordinates. See Figure 5 for a physical description of the gantry.

$$M1 = (y - x) * k$$
$$M2 = (y + x) * k$$
$$M3 = z * j$$

where x,y and z are workspace coordinates of the end effector in units of 0.1 mm, k is a constant scale factor to convert from units of 0.1 mm to steps, and j is a constant scale factor to convert from 0.1 mm to Analog-Digital Converter (ADC) counts.

2. Relative Motion: The low level controller can accept point information in terms of either relative or absolute coordinates. If relative coordinates are used, the motion will be performed relative to the robot's previous position.

3. Trajectory Planning: The low level controller receives path information in the form of points. In order to execute a motion, it must convert these paths into a trajectory by adding velocities and accelerations for each point. These values are determined by assigning a time stamp to each point based on the euclidean distance between them. A trajectory is then calculated automatically by the PMD Machine Controller to ensure the robot reaches each point in the specified time.

4. Sensor Reading: All sensors (except the vision camera) are connected to the low level controller, which is responsible for reading their values and acting on them as necessary. The list of sensors includes:

    - Encoders (2): These encoders are built into the FESTO stepper motors used to control motion in the X and Y axes. They feed directly into the feedback port for the corresponding motor on the Machine Control Board. Their values are available through PMD function calls in low level code.
    - Potentiometer: This potentiometer is built into the Firgelli linear actuator used to control motion in the Z axis. It is wired to Analog Input Pin 1 on the PMD Machine Control Board.
    - Sharp IR distance sensor: This sensor is attached to the end effector and can be used to measure the height of the paintbrush from the canvas. It is wired to Analog Input Pin 2 on the PMD Machine Control Board
    - Force Plate: The force plate consists of three load cells which can be used to measure the force being applied to the canvas. Raw data from the load cells is read and interpreted by an Arduino Uno, force values are sent to the low level controller in a parallel interface which uses digital I/O pins 1-4 on the PMD Machine Control Board.
    - Push Buttons: Two push buttons used for Pause/Resume functionality are wired to Digital Input Pins 1 and 2 on the PMD Machine Control

Board. When pressed, the appropriate action is automatically taken by the low level controller.

6.2 Performance Motion Devices Controller

All low level control and computation is performed by the Prodigy/CME Machine Control Board which was donated to the project by Performance Motion Devices. More information about the board as well as documentation can be found at on the PMD website [8].

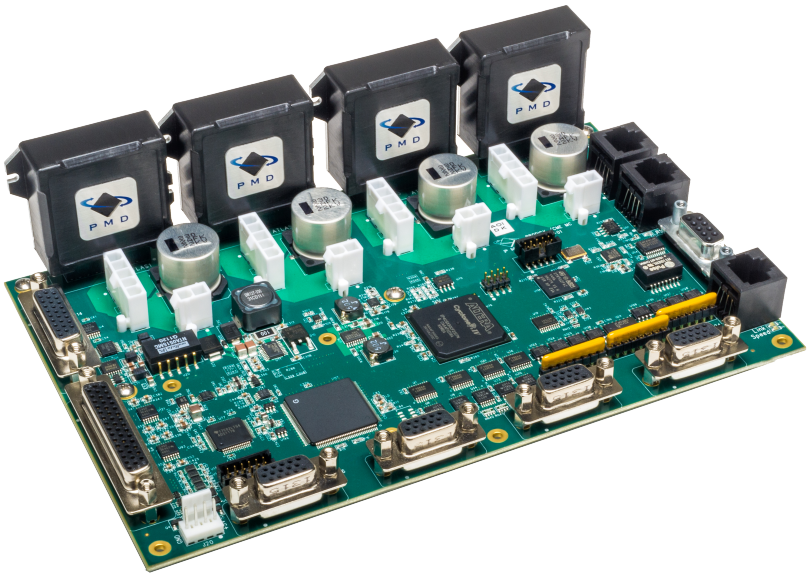

Figure 30: PMD Machine Control Board

7 High Level Software Design

In this chapter, the high level software developed for the painting procedure is outlined and each module in the process explained.

7.1 General Overview of Vision Feedback

In order to correct errors within the painting, a camera mounted with full view of the gantry workspace was used. The camera first isolates the canvas. The canvas is then classified into the colors that are used on this painting as well as the canvas color. The correction algorithm next examines differences between what has been found on the canvas and what areas belong to which color features in the decomposed image.

Areas found to be of an inappropriate coloring are then corrected in the next round of painting. Errors found may be where the brush has begun to run out of paint or where paint has been dripped or painted incorrectly. These areas found as error may come from a recomposition planner that is not intended to fill the image fully with color.

For example, the medial axis recomposer seeks only the midpoint to paint before it looks for error to correct. Here, the error correction allows for multiple passes of recomposition and painting to run iteratively. This allows recomposers such as the medial axis recomposer to make simple changes and have faster feedback and shorter painting steps.

7.2 Camera Transforms

In order to isolate the canvas, there are two major steps. The camera used on this robot utilizes a fish-eye lens, which allows a greater field of view but also warps the image. This warped image is corrected using OpenCV's perspective warping functionality. Once the image has been rectified, the canvas must be isolated.

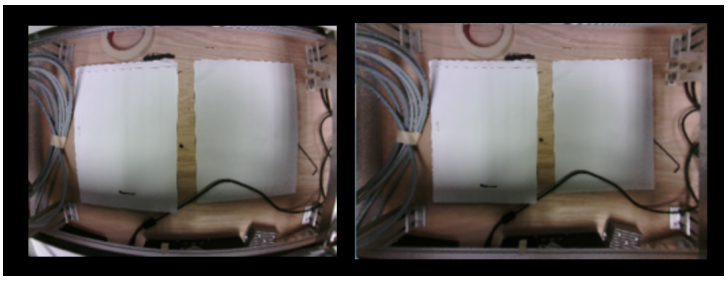

Figure 31: Camera Dewarping

In order to find the canvas, the dewarped image is first converted to a gray-scale, blurred, and then canny edge detection is used to isolate the canvas. Using the locations of the four corner points of the canvas, a transform is generated from the original camera image directly to the canvas-space. This allows the canvas to be easily extracted from new camera frames without having to rerun the canvas searching operation.

Figure 32: Canvas Isolation

7.3 Decomposition

The decomposition step in the painting process is characterized by extracting a number of distinct color layers from an input image. There are three ways this can be performed:

1. Input 'n' colors to decompose the image into. The robot performs k-means clustering and outputs the best 'n' colors to use. These colors are used to segment the image and the user can prepare these colors in the paint palette. This method will usually produce paintings which most closely match the original image, but requires users to mix paint colors themselves to match the ones generated.

2. Input the color values of the paint already in the palette. The robot decomposes the input image by mapping each pixel in the image to a paint

color based on similarity. The assigned pixels are then used as the color layers. This method requires the user to decide which colors best represent the image. The results can vary widely based on the user's choices and the composition of the input image. For example, it is possible that although the user provides six colors, only one or two are actually used.

3. Input both the number of colors to decompose the input image into and the color values of the paint already in the palette. The robot performs k-means clustering and outputs the 'n' best colors to use. Color matching is then used to map these best colors to those readily available in the paint palette. Unlike the previous method, this method will attempt to use as many unique palette colors as possible. This can be useful if the user already has a large selection of pre-mixed palette colors and wants to automatically select the best ones to use for a certain image.

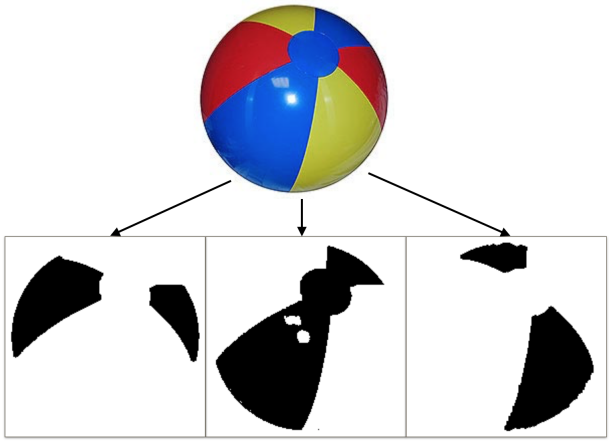

Figure 33: Decomposition performed through k-means

7.3.1 K-means Classification

One of the ways to perform color based segmentation is to use k-means. Given that a desired number of colors is provided, k-means starts with an initial guess, and iteratively updates the centroid of each color cluster until the algorithm converges. This generally happens when the sum of squared distances between clusters has been minimized to below a certain threshold value. The centroid of each cluster then represents the color to use for that image layer. Figure 33 shows decomposition of a beach ball performed through k-means. The three decomposed segments are clearly shown for the red, blue, and yellow color layers in the picture.

7.3.2 Color Matching

One of the problems encountered during the painting process was being able to relate colors in the input image to paint colors available in the palette.
This was needed for the following reasons:

- To allow the user to decompose an image with both k-means and pre-made palette colors
- To ensure color layers where appropriately matched to their respective k-means layer, since the decomposed cluster colors where not always returned in the same order
- To allow vision feedback to correctly identify the decomposed colors from the canvas

To resolve this problem, the colors from the image and the palette needed to be converted to the L*A*B* color space. This is due to the fact that the traditional RGB color space is linear and not perceptually uniform. Colors in L*A*B* space more closely mimic human perception and the distance here more accurately relates to the similarity between two shades. Converting a color from RGB to L*A*B* color space and vice versa can be performed using the equations shown in Appendix C.

The L*A*B* color space was first specified by the International Commission on Illumination (Commission internationale de l'éclairage or CIE) for the purpose of creating a 3D color representation that matches human perception. The CIE refers to the metric of similarity between two colors as ΔE. The first ΔE equation came in 1976 and simply calculated the Euclidean distance between two colors. However, L*A*B* space does not accurately consider differences in regions of high saturation. To compensate for this, ΔE76 was replaced by a new equation in 1994 and then again in 2000. ΔE00 is a much better metric for assessing the similarity between two colors. These ΔE equations where used to perform accurate color mappings from image colors to paint colors where the smaller the ΔE the more similar two colors appear visually. For more information of the different ΔE equations see [5].

7.4 Recomposition

The term Image Recomposition is used to refer to the process of generating strokes required to paint a given image. This can also be thought of as a motion planning process. Motion planning for a painting robot falls under the category of Coverage Path Planning algorithms. The goal of such algorithms is to find a path that fully

explores a search space, while optimizing variables such as time spent and distance traveled. In the case of a painting robot, the search space includes areas on the canvas where paint should be placed, while "obstacles" can be defined as areas where paint should not be placed. A description of some of these algorithms and how they work is provided in the following sections.

Each recomposition algorithm takes as input a binary image. '0' in the image represents areas which should be painted, and '1' or '255' represents areas which should not. These binary images can be generated by the Decomposition process (one for each color). Some recomposition methods also require other parameters such as brush size. Based on these inputs, the algorithm outputs a series of paths which if followed should result in the image being painted as desired. Specifically, this output is represented as a List of Lists of Tuples (LLT).

$$[[(x, y, w), ...], ...]$$

Each point in the LLT is represented as a Tuple containing 3 elements: x, y and w. x and y represent the 2-dimensional location in terms of image coordinates. w is used to quantify the "width" of the brush at a given point. It can also be thought of as the distance from the given point to the nearest edge. The w value is not currently utilized in the painting process, but in the future it could be used to modify the height or angle of the brush to change the apparent width while painting.

Each point is contained within a list which represents a stroke. The brush will be put down at the first point in a stroke and lifted up at the last point. Strokes may consist of any number of points, from "dots" which contain only one point, to complex paths containing hundreds. Since an image can contain any number of strokes, each stroke is also stored in a list. Thus the output of the recomposition process is an LLT containing all stroke/point information required to paint a single binary image.

7.4.1 Iterative Erosion

The iterative erosion algorithm works by generating a path from the edges of the desired painted shape. Next, the image is eroded by a fixed amount and the new edges are used to form a new path. The process repeats until the image has been fully eroded and there is no more area to paint. The result is a series of concentric paths which follow the outer/inner edges of the search space, as seen in Figure 34. While this algorithm works well for large filled regions, the nature of the erosion function causes it to miss smaller details which are still desired for painting.

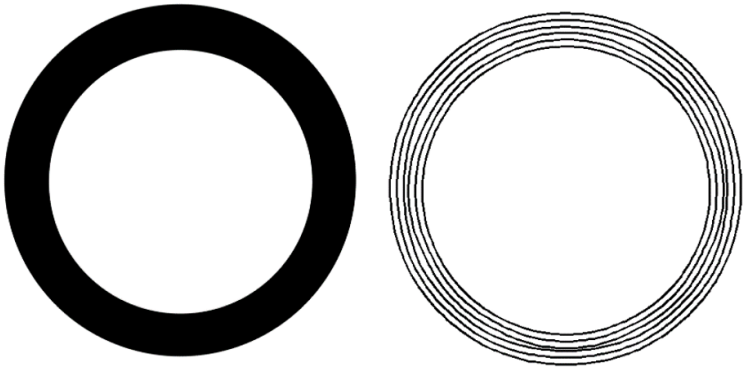

Figure 34: Iterative Erosion Recomposition

7.4.2 Medial Axis

The medial axis is defined as the set of all points having more than one closest point on an object's boundary. It can also be thought of as the "skeleton" of the object, or the points furthest from the edges. For the purpose of this project, it is effectively equivalent to the Voronoi diagram of the regions that should be painted. In narrow regions, the medial axis can be used to determine the path along which the brush should travel in order to paint the desired image. The result of this transformation on a simple circle is shown in Figure 35. While this method alone does not generate paths which completely fill the desired regions, it performs very well at efficiently painting thin, narrow segments, which would otherwise be ignored by other recomposition algorithms.

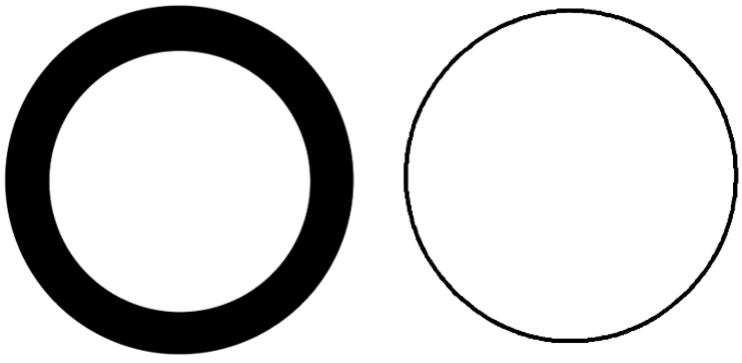

Figure 35: Medial Axis Recomposition

The medial axis transformation can be performed in numerous ways. Our final implementation consists of the following steps, which are visualized in Figure 36:

1. Create an image by assigning values to each pixel based on the distance to the nearest edge.
2. Compute the Scharr derivative of this image along x and y axes
3. Find the magnitude of these derivatives
4. Define Medial Axis as points where this magnitude is near zero.

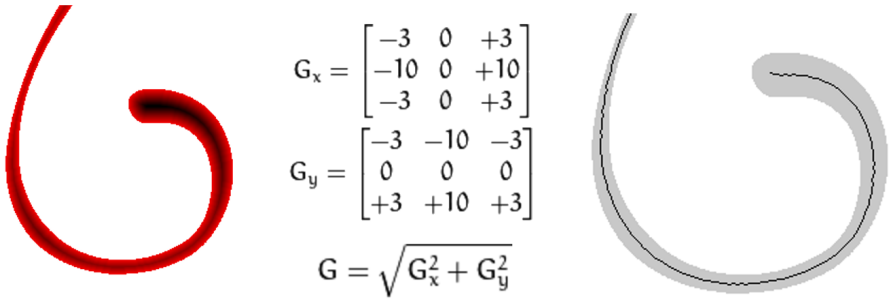

Figure 36: Medial Axis Transformation Process

Once the points which lie along the medial axis are found, the next step is to generate a path which traverses these points. This is done via the following method:

1. Convert the medial axis into a graph by treating each point as a node and connecting nearby points with edges
2. Choose an arbitrary starting point, which has not yet been visited
3. Run Breadth-First search from current node, but only to a certain depth (2-4 children). Record minimum known distance from each node to the start node. Add each newly discovered node to the frontier
4. Choose the frontier node that is **furthest** from the start node (Depth-First search). Check if it is an endpoint. The current node is an endpoint if:
    - All adjacent nodes have been visited or are in the frontier
    - Of all neighboring nodes, this one has the greatest distance from the start

5. If the point is **not** an endpoint run BFS again starting from this node (repeat step 3)

6. If the point **is** an endpoint, create a path by tracing parent nodes as far as possible. While tracing this path, flag all nodes adjacent to the path as "dead." Since they are close to an existing path they do not need to be considered further, and the algorithm will ignore them.

7. Repeat steps 3-6 while there are nodes remaining in the frontier

8. Repeat steps 2-7 while there are any points that have not been visited (there may be several unconnected graphs in a single image)

The result of this process is a series of paths which overall traverse the medial axis without necessarily visiting each node. Figure 37 shows the paths and endpoints produced for a portion of an image containing complex geometry with multiple cycles.

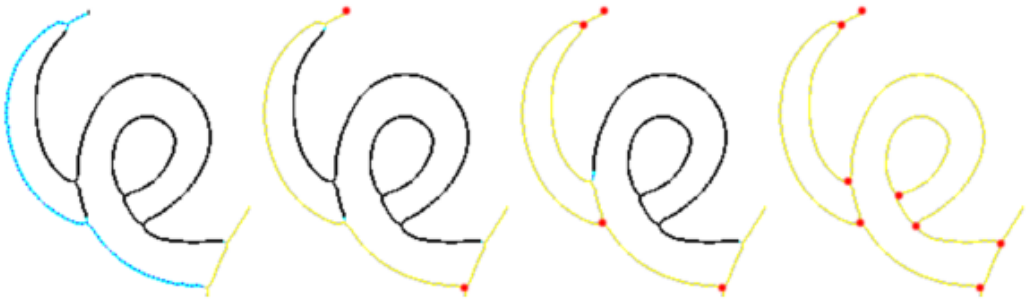

Figure 37: Graph Search Process

7.4.3 Blended

The final recomposition method incorporates both iterative erosion and medial axis transformation to create an algorithm which robustly generates paths along both large filled regions as well as thin narrow segments. The first step is to erode the image by given amount, usually the diameter of the brush. This eroded image is then subtracted from the original image, which produces an "outlined" image as shown in Figure 38. Paths are generated to paint this outer image using the medial axis recomposition method. The process repeats using the eroded image as the original, until there are no additional regions left to paint.

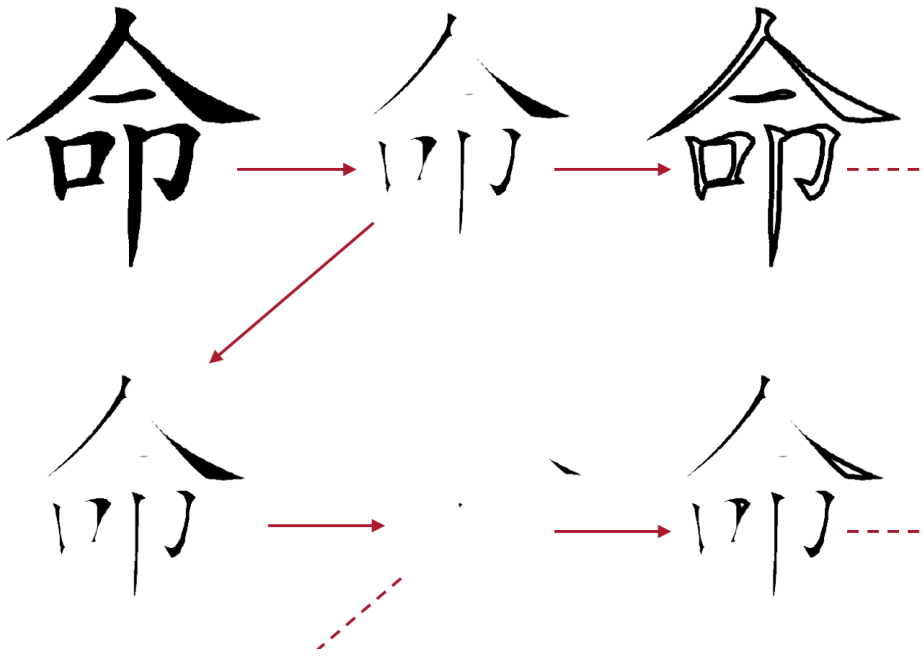

Figure 38: Blended Recomposition Process

Figure 39 shows the results of the iterative erosion method (left) compared to those of the blended method (right). It is clear that the blended recomposition method fills large regions more completely than iterative erosion. Furthermore, it is able to plan paths for small narrow regions just as well as the medial axis method alone, making it the most robust recomposition method for most images.

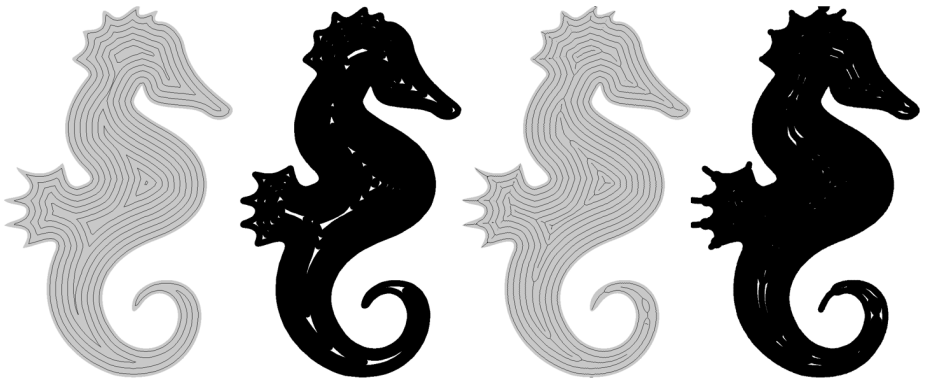

Figure 39: Iterative Erosion vs Blended Recomposition

7.5 Error Checking

The error checking process operates on a color-by-color basis, isolating error to specific color features. If the robot is provided with a paint color matching the canvas, it may use that paint color for correction as well. This would be used if paint splattered or dripped into areas of the canvas that should be untouched. If no canvas colored paint is provided, the robot simply ignores the canvas.

In order to examine error on the canvas, the canvas is first segmented into colors, using the same code that decomposes source images into specifically colored sections. These colored sections on the canvas are then each binarized and compared to the color sections of the decomposed source image.

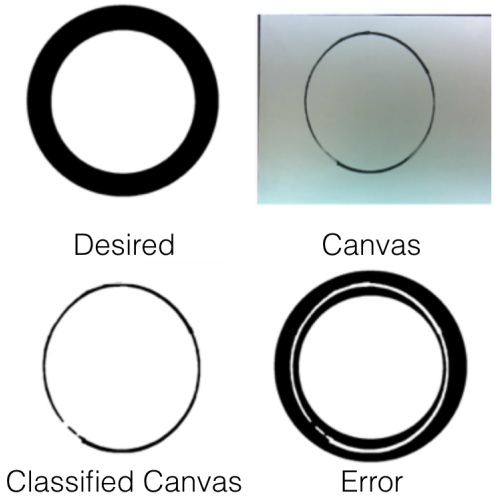

Figure 40: Error Calculation

The classified canvas and the desired image are compared using simple XOR logic. Areas of a color feature that are white in both binary images are left as white. Similarly, areas that are black both on the canvas and in the desired image are left white. However, any black areas of the desired image that are white on the canvas are considered error, and painted black. These are the areas that were missed in the most recent painting step. The above figure depicts this process occurring for a painting of a black circle utilizing the medial axis recomposer. Once these error regions have been identified, new paths are generated to paint over erroneous regions using the same image recomposition process as before.

7.6 User Interface

To facilitate the painting process and relieve the user from the code, a graphical user interface(GUI) was created. This GUI was made using wxPython, a cross-platform python library. The resulting program is depicted in Figure 41 and consists of one panel built from four main components described in the next few paragraphs.

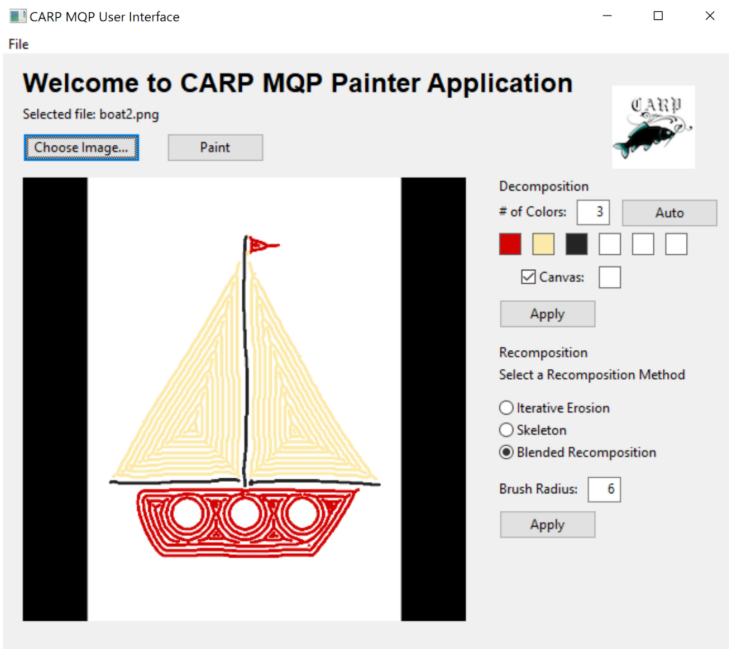

Figure 41: Graphical user interface for the painting robot platform.

Selection Component
This is the entry point for a user to interact with the GUI. Here the user can click the "Choose Image..." button and a dialog box will appear allowing the user to select an image file. Once a picture has been selected the name of the file is shown directly above the button, the file contents are shown in the "Image Display" component, and the Decomposition/Recomposition components become enabled. Additionally, there is also a paint button that becomes enabled once the Decomposition/Recomposition components have been completed. This button initializes the painting routine for the robot.

Image Display Component
This is where the input image is shown. This component provides feedback for the user to track how the software is interpreting and preparing the input image for painting.

Decomposition Component
In this component decomposition of the input image is performed. A text field is available for the user to input the number of colors to use for k-means. The user can also click on the 'Auto' button which will allow the software to select the 'n', up to six, best colors to use for the image. Additionally, the user can also directly input the colors they want to use in the palette. The color picker tool facilitates this process and allows for paint colors to be easily re-ordered to match the palette. Optionally a canvas color can be given to tell the robot not to paint the background. Hitting the 'Apply' button performs the decomposition with the selected settings and the results are shown in the Image Display component.

Recomposition Component
This component performs image recomposition. There are currently three different options for how an image can be recomposed: Iterative erosion, skeleton, and blended recomposition. For more details on these recomposes see section 7.3. Clicking on the radio button next to the option selects the method of recomposition. Then a brush radius can be specified to indicate stroke width. Hitting the 'Apply' button performs the recomposition and the results are shown in the Image Display component.

8 Robot Artwork

In this chapter, original artworks created from the "mind" of the robot are explored.

8.1 Abstract Art

While painting images from reference pictures was a major component of the project, abstract art was also investigated as a method for creating "original" robot art. One approach taken was to input music or other audio files for the robot to "interpret" and determine the layout of a painting based on the pitch and frequency of the sounds. Another technique was to input a Gaussian point cloud and allow the robot to randomly select points within the distribution to create abstract art. Both methods are detailed in the following subsections.

8.1.1 Art created with music

Abstract art was created with music as an input through the following method:

1. Finding the BPM of the song (Figure 42)
2. Dividing the song up based on its BPM
3. Determining the pitch, confidence level, and dB level at each point (Figure 43)
4. Graphing the pitch vs. the confidence level
5. Creating a number of rings corresponding to the dB level at each point

An example of artwork generated in this manner is shown in Figure 44.

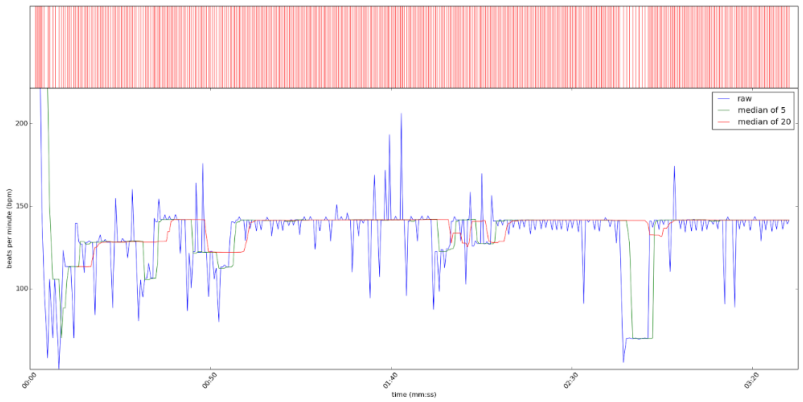

Figure 42: BPM calculations for a song

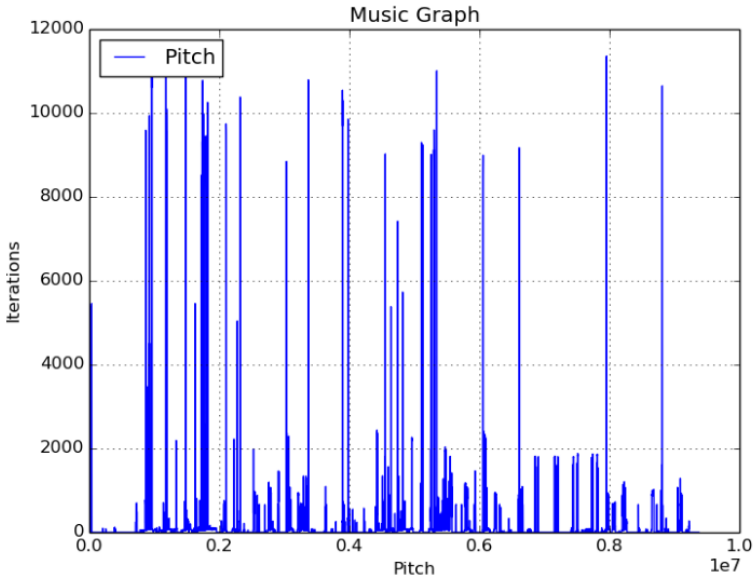

Figure 43: Pitch values of a song

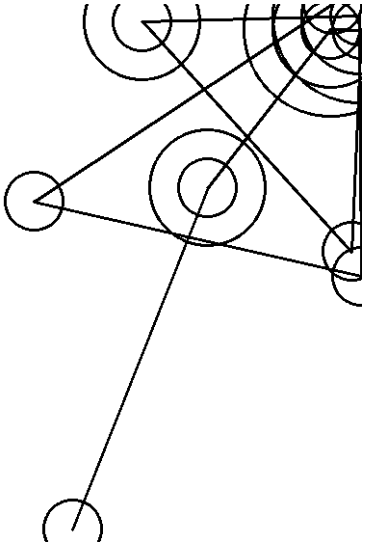

Figure 44: Art from Death of a Bachelor by Panic! at the Disco

8.1.2 Art created from a Gaussian

To create abstract art from a point cloud, two distributions were given to the robot a Gaussian along the x-axis and a Gaussian along the y-axis with respect to the canvas. Each of these were given a mu located at the center of the canvas and a sigma of half the canvas width/height respectively. The robot then selected a defined number of x,y pairs from the horizontal and vertical distributions to create a path to be painted. This was combined with the color palette code to allow for multiple distributions of different hues to be layered on top of one another. The results can be seen in Figure 45.

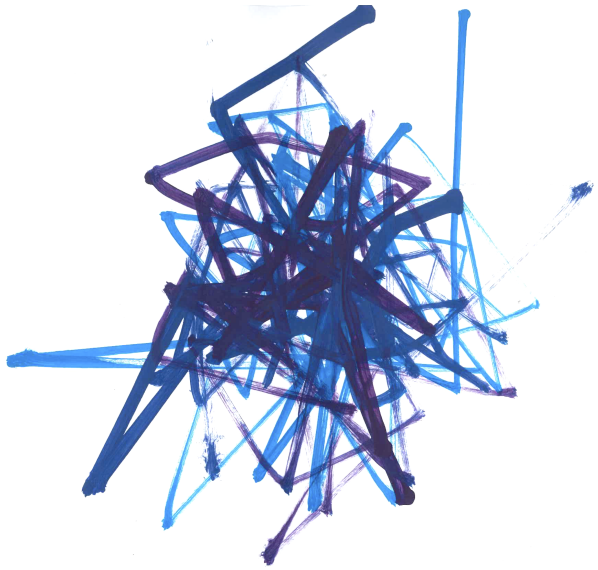

Figure 45: Abstract art created from a Gaussian distribution

9 Conclusions and Future Directions

A robotic painting platform was developed that successfully met both the required and strive goals for the project. The modular design allows the robot to be extended for research. Sensors allow the platform to locate the end-effector in the workspace and a camera allowed vision feedback to be accessed.

Paintings produced by the robot had distinct brush strokes greatly avoiding pointillist stylizations and mimicking a more human procedure. The final product resembled the input image and multiple colors could be used for more sophisticated paintings. Vision feedback was successfully incorporated into the high level code and the robot was able to perform multiple iterations of correction on its art. A user interface was developed to allow users to use the robot and generate their own artwork without any technical experience.

The project can be easily extended to a variety of engineering disciplines. Future recommendations were broken down into the following categories:

Mechanical Direction
The overall mechanical design in this project was simple, allowing for multiple areas of improvement. Additional degrees of freedom can be added to the end-effector to allow for angling of the paintbrush on the canvas. This would give a more "human" feeling to the produced paintings.

The platform is limited to six colors exclusively so improving upon the tool changing to incorporate more colors could be an interesting direction. Another possible area of exploration could be to create a paint mixing module that would allow the robot to be more independent from human actors who currently provide pre-mixed paints. Additionally an automated system of cleaning brushes or preventing them from drying out would decrease the need for human intervention in the painting process.

Software Direction
There are a number of different directions that can be pursued to improve the software on the robot. More sophisticated decomposition and recomposition techniques could be explored to allow the robot to layer paints or depict gradients, two limitations with current system. Machine learning could also be integrated to allow the platform to learn from past experiences or from human painters. Finally, in this project a single medium was considered. However, the platform can be easily extended to other art forms such as crayons, pencils, pens and others. Other forms may utilize different sensors such as force or IR distance which are already available on the robot.

Appendices

A Project Balance Sheet

Description	Debit	Credit	
University Budget		$ 1,000.00	
Order on 9/26	$169.02		80/20 Components
	$20.05		Infrared Sensor + Jumper cable
		$ 810.93	
Order on 9/26	$28.19		Art supplies
	$102.81		Firgelli Linear Actuator
	$20.99		Wood from Home Depot
		$ 658.94	
Order on 11/4	$50.70		Load Cells + Amplifiers
		$ 608.24	
Order on 12/9	$33.28		Heat shrink + Cable
	$94.85		Camera, 48V Powersupply, and 48V to 12 V Converter
		$ 480.11	
Order on 1/25	$48.95		Acrylic Sheet, Frames, cables
	$28.76		Small Components from 80/20
		$ 402.40	
Order on 1/25	$7.38		Screws from McMaster
		$ 395.02	
Order on 2/3	$32.61		More parts from 80/20
		$ 362.41	
Order on 2/6	$7.99		LED Strip
		$ 354.42	
Order on 2/14	$16.19		More screws from McMaster
		$ 338.23	
Order on 2/17	$6.87		12V to 5V Converter
		$ 331.36	
Order on 2/27	$19.62		Load Cell Replacement
	$24.40		Art supplies from C.C. Lowell and Lowe's
		$ 287.34	
Order on 3/1	$41.48		Enclosure components from Amazon
	$49.28		Enclosure components from McMaster
		$ 196.58	
Order on 3/3	$80.56		Update for Amazon order
		$ 116.02	
Order on 3/22	$37.61		Terminal Block + Din Rail mount + Dsub cable for cleaning up Firgelli
	$6.38		Art supplies from C.C. Lowell
		$ 72.03	
Order on 4/20	$30.00		Arduino Uno + Painting paper
		$ 42.03	

Figure 46: Balance Sheet for robot painting platform

B Color Matching Equations

B.1 RGB to XYZ

Given the chromaticity coordinates of an RGB system (x_r, y_r),(x_g, y_g), and (x_b, y_b) reference white (X_w, Y_w, Z_w):

$$\begin{bmatrix} R \\ G \\ B \end{bmatrix} = [M] \begin{bmatrix} X \\ Y \\ Z \end{bmatrix}$$

$$[M] = \begin{bmatrix} S_r * X_r & S_g * X_g & S_b * X_b \\ S_r * Y_r & S_g * Y_g & S_b * Y_b \\ S_r * Z_r & S_g * Z_g & S_b * Z_b \end{bmatrix}$$

$$X_r = x_r / y_r$$

$$Y_r = 1$$

$$Z_r = (1 - x_r - y_r)/y_r$$

$$X_g = x_g / y_g$$

$$Y_g = 1$$

$$Z_g = (1 - x_g - y_g)/y_g$$

$$X_b = x_b / y_b$$

$$Y_b = 1$$

$$Z_b = (1 - x_b - y_b)/y_b$$

$$\begin{bmatrix} S_r \\ S_g \\ S_b \end{bmatrix} = \begin{bmatrix} X_r & X_g & X_b \\ Y_r & Y_g & Y_b \\ Z_r & Z_g & Z_b \end{bmatrix}^{-1} \begin{bmatrix} X_W \\ Y_W \\ Z_W \end{bmatrix}$$

B.2 XYZ to L*A*B*

$$L = 116 f_y - 16$$

$$a = 500(f_x - f_y)$$

$$b = 200(f_y - f_z)$$

where

$$f_x = \begin{cases} \sqrt[3]{x_r} & \text{if } x_r > \epsilon \\ \frac{\kappa x_r + 16}{116} & \text{otherwise} \end{cases}$$

$$f_y = \begin{cases} \sqrt[3]{y_r} & \text{if } y_r > \epsilon \\ \frac{\kappa y_r + 16}{116} & \text{otherwise} \end{cases}$$

$$f_z = \begin{cases} \sqrt[3]{z_r} & \text{if } z_r > \epsilon \\ \frac{\kappa z_r + 16}{116} & \text{otherwise} \end{cases}$$

$$x_r = \frac{X}{X_r}$$

$$y_r = \frac{Y}{Y_r}$$

$$z_r = \frac{Z}{Z_r}$$

$$\epsilon = \begin{cases} 0.008856 & \text{Actual CIE standard} \\ 216/24389 & \text{Intent of the CIE standard} \end{cases}$$

$$\kappa = \begin{cases} 903.3 & \text{Actual CIE standard} \\ 24389/27 & \text{Intent of the CIE standard} \end{cases}$$

For more information on this process see [5]

C Code

For more information on the code used in this project please see the files attached to this submission or visit our repository at: https://github.com/Kygandomi/CARP

D Gallery

A few samples of paintings produce by the robot. For a complete collection see attached document.

Figure 47: Dog Painting

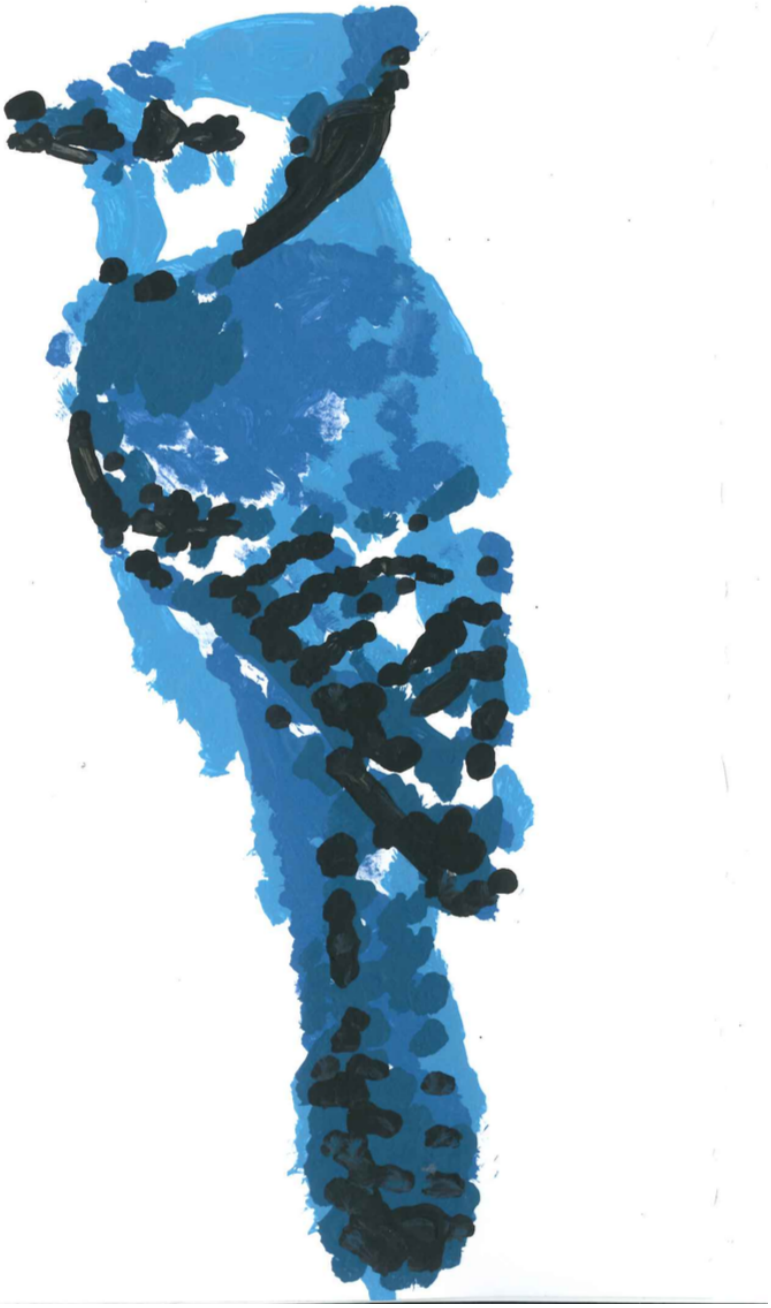

Figure 48: Bluejay Painting

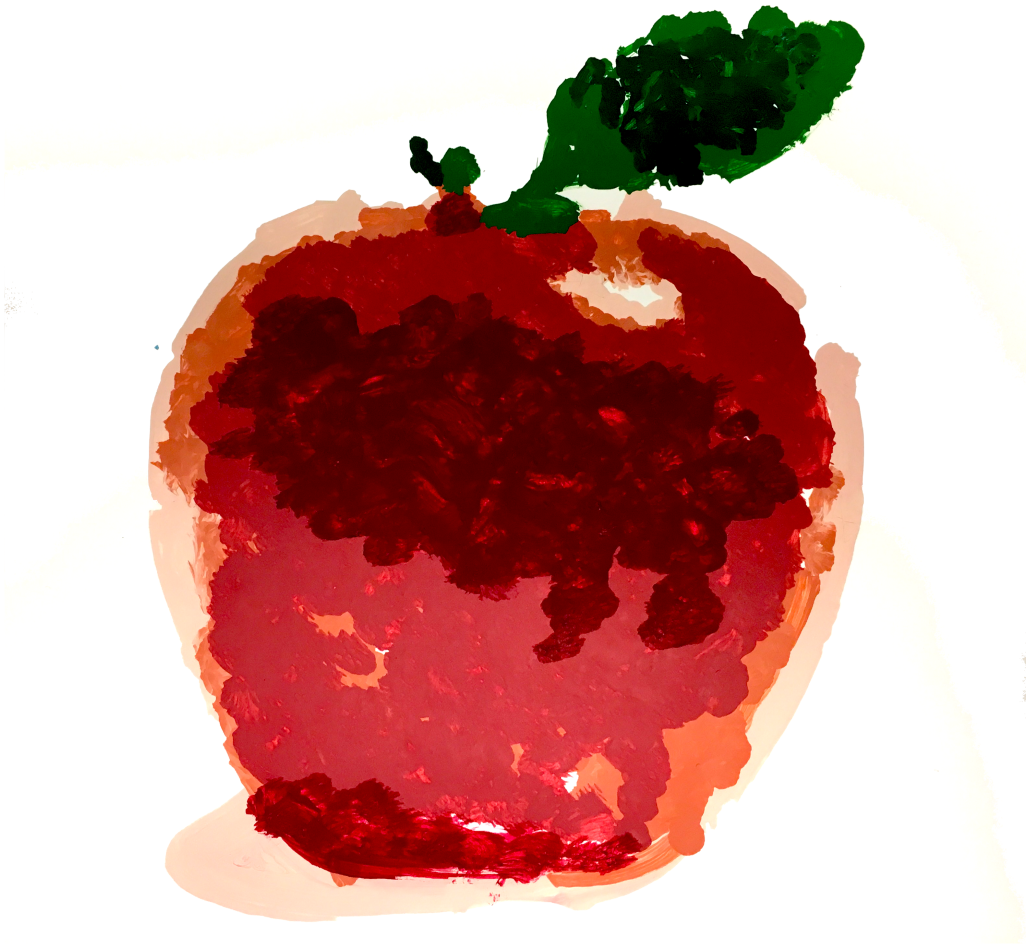

Figure 49: Apple Painting

E Technical Drawings

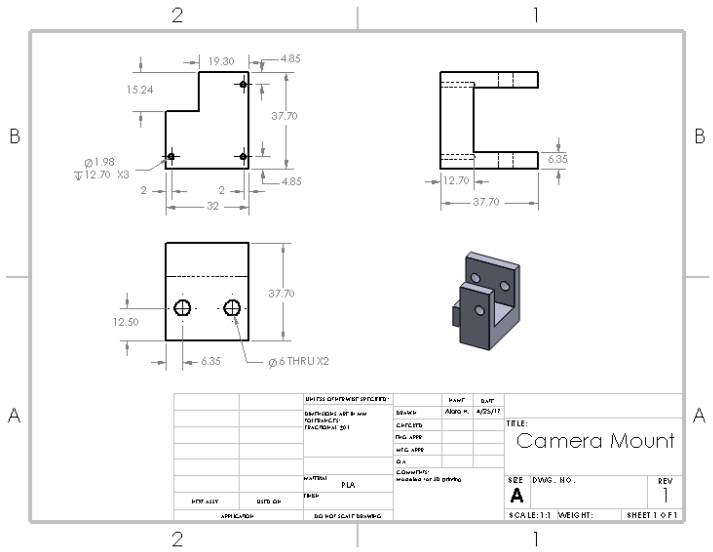

Figure 50: Camera Mount Drawing

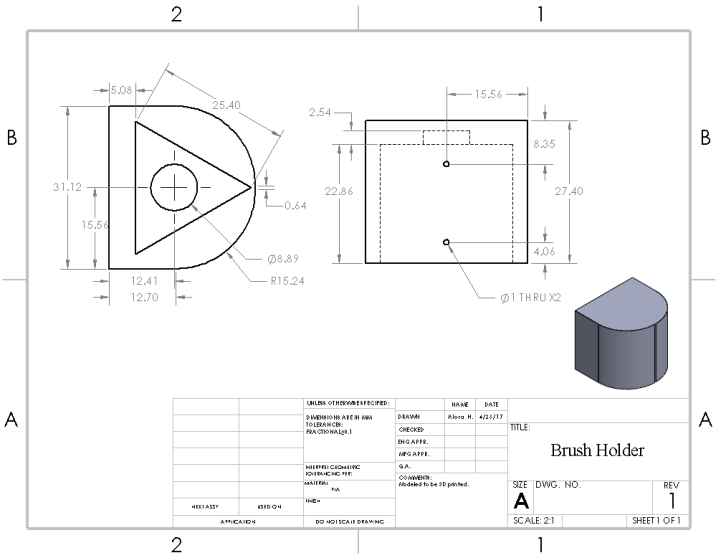

Figure 51: Brush Holder Drawing

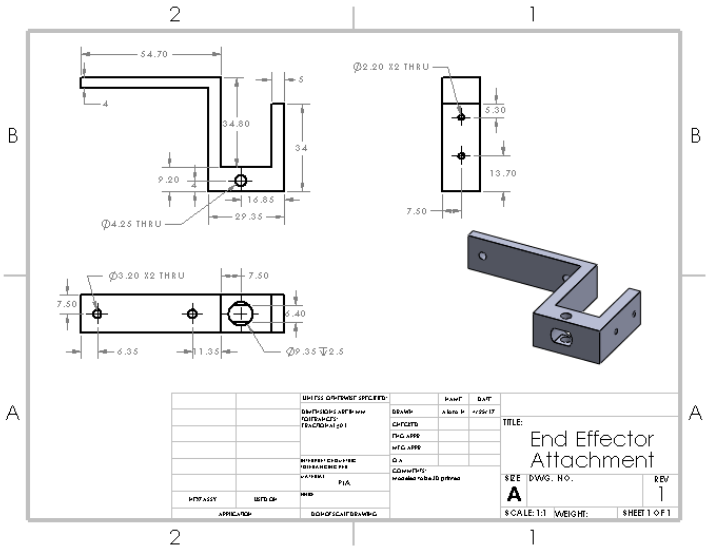

Figure 52: End Effector Attachment Drawing

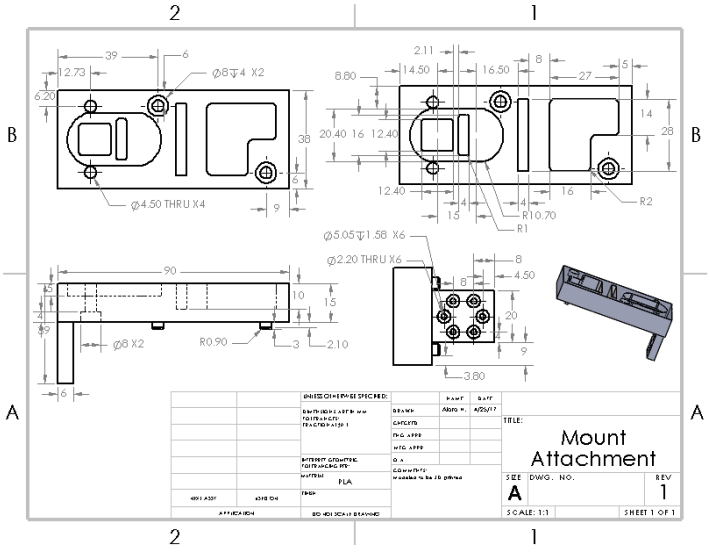

Figure 53: Mount Attachment Drawing

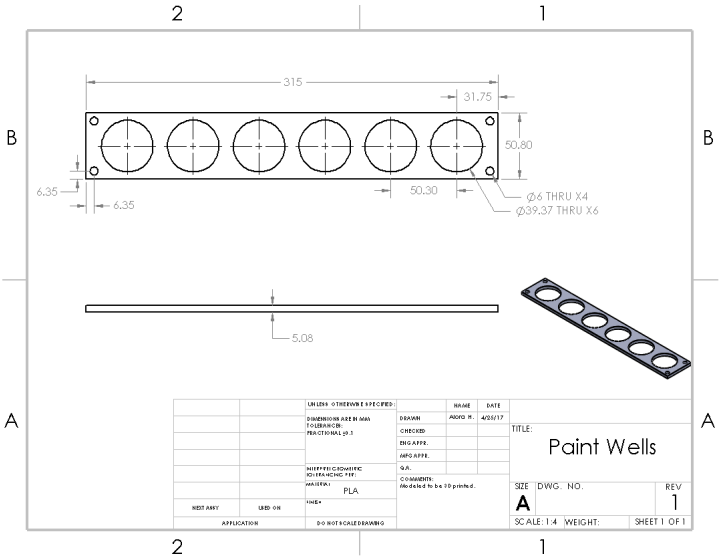

Figure 54: Paint Well Drawing

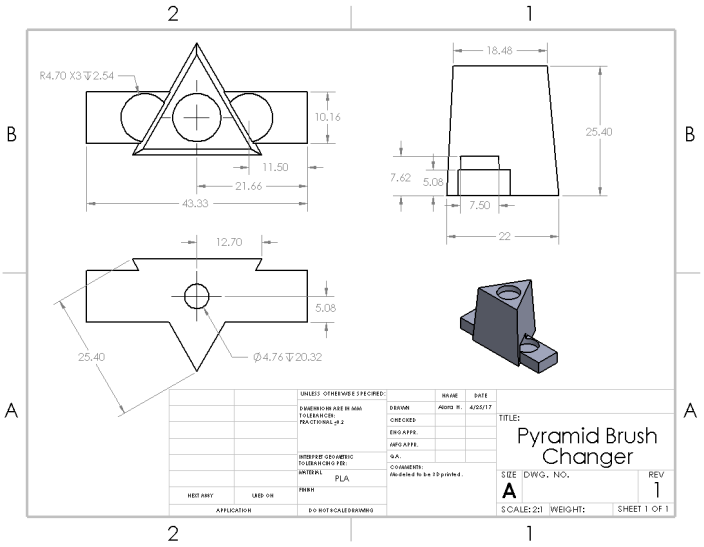

Figure 55: Pyramid Brush Changer Drawing

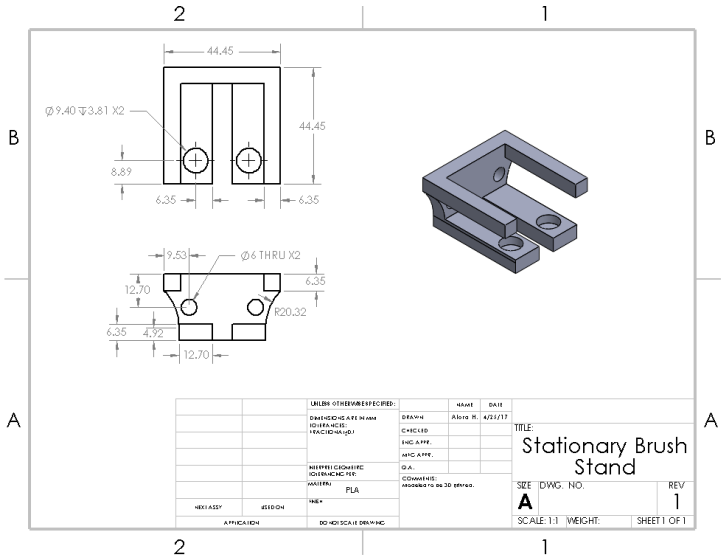

Figure 56: Stationary Brush Stand Drawing

F Authorship

Table 3: Authorship

1.1 Problem Statement	Team
1.2 Project Justification	Team
2.1 Painting Techniques	Hillman
2.2 Literature Review	Team
3.1 Project Goals and Objectives	Hillman
4.1 Robot Platform	Hillman
4.2 End Effector Design	Hillman
4.3 Electronic Enclosure Heating Calculations	Hillman
5.1 Sensing on the Robot Platform	Hillman
5.2 Electric Enclosure	Hillman
6.1 Low Level Software Functions	Panzarino
6.2 Performance Motion Devices Controller	Panzarino
7.1 General Overview of Vision Feedback	Dotson
7.2 Camera Transforms	Dotson
7.3 Decomposition	Gandomi
7.4 Recomposition	Panzarino
7.5 Error Checking	Dotson
7.6 User Interface	Gandomi
8.1 Abstract Art	Gandomi, Hillman
9.0 Conclusions and Future Directions	Gandomi

References

[1] Bouwmans, T., Baf, F. E., & Vachon, B. (2010). Background Modeling using Mixture of Gaussians for Foreground Detection - A Survey. Recent Patents on Computer Science CSENG, 1(3), 219-237. doi:10.2174/1874479610801030219

[2] An Introduction to Chinese Brushpainting Techniques. (n.d.). Retrieved July 28, 2016, from http://education.asianart.org/explore-resources/background-information/introduction-chinese-brushpainting-techniques

[3] Artist Paint Brushes: A Guide to Choosing the Right Paintbrushes for Painting with Acrylics. (n.d.). Retrieved July 28, 2016, from http://www.art-is-fun.com/artist-paint-brushes/

[4] Kudoh, Shunsuke, et al. "Painting robot with multi-fingered hands and stereo vision." Robotics and Autonomous Systems 57.3 (2009): 279-288.

[5] Lindbloom, Bruce Justin. Welcome to Bruce Lindbloom's Web Site. N.p., n.d. Web. 25 Apr. 2017.

[6] Lindemeier, Thomas, Soren Pirk, and Oliver Deussen. "Image stylization with a painting machine using semantic hints." Computers and Graphics 37.5 (2013): 293-301.

[7] Wang, Y. (n.d.). Tutorial: Image Segmentation. Retrieved from *http : //disp.ee.ntu.edu.tw/meeting/Segmentationtutorial.pdf*

[8] https://www.pmdcorp.com/product/board