MAGNETO
Control Interface for Autonomous Delivery of Magnetically Stimulated Particles

Team Name:
Team 7 (Nson)

Team Members:
Victor Huynh, Bassam Mutawak, Elizabeth Ankrah, Minh Quan Do

Weinberg Medical Physics Sponsors:
Dr. Lamar Mair, Dr. Chad Ropp, Olivia Hale, Dr. Irving Weinberg

Faculty Advisors:
Dr. Nathalia Peixoto, Dr. Qi Wei

Course Title:
BENG 493: Senior Advanced Design Project II

Date:
10 May 2019

Executive Summary:

This document provides a final design report for Project Magneto, the Bioengineering senior design project in collaboration with George Mason University (GMU) and Weinberg Medical Physics LLC (WMP) from Fall 2018 - Spring 2019. Before the start of Spring 2019, the scope of Magneto was updated after advising and discussion with both our GMU advisors and WMP sponsors. Magneto is a C++ software interface that controls WMP's 4-coil magnet array system for autonomous delivery of a magnetic object to user-defined locations. The project's primary purposes are:

1) To be a software precursor to WMP's future magnetic particle delivery system for non-invasive surgeries via MRI imaging and electropermanent magnet activation.

2) To improve control and versatility of WMP's 4-coil magnet array system used for pilot research experiments.

Magneto can be broken into 4 overarching components:

1) **Graphic User Interface (GUI)** - This is the user's main tool for system operation. The GUI implements and integrates all other components to perform particle delivery. Major functionalities include intuitive delivery path drawing, customized data collection, and system error detection.

2) **Image Segmentation Module** - This receives and processes optical images streamed via camera. It calibrates a consistent coordinate system and physical field-of-view regardless of the hardware orientation under the camera. It also performs detection of the magnetic object of interest through all streamed frames.

3) **Physics Module (Particle Translation Model)** - This determines electrical current values and their applied duration in order to translate the magnetic object along the delivery path. A translation command is generated from the model and sent to the motor controller hardware for execution.

4) **Hardware Setup** - This is the configuration and setup of the motor controller hardware needed to interface with WMP's 4-coil magnet array system. The control scheme allows for communication from the GUI to the controllers and vice versa.

This report includes descriptions of the final component designs with rationale against alternatives. Comparison and testing results are provided to address project metrics and evaluate overall project success. Project impact and next steps are also discussed. In summary of the major design improvements over the Fall 2018 semester:

- The GUI received a streamlined layout and additional functionality for path drawing, system control, and more.

- Image Segmentation module now employs ArUco fiducial markers to rapidly calibrate system orientation and utilizes a more dynamic implementation of image subtraction that does not require acquisition of a true background image.

- The Particle Translation Model is now implemented via Regression Neural Network for handling of multiple feature inputs. The model displayed high translation consistency.

- Hardware Setup utilizes packet serial communication protocol, allowing for bidirectional communication with the software. Simultaneous control of multiple motor controllers was achieved via multithreading.

The project budget total is $919.28. However, the majority of costs were covered by Weinberg Medical Physics, LLC (see Appendix C).

*Note: From here on, the word “particle” will be used in place of “magnetic object.”

Project Contributors:

Team	
Victor Huynh	Project Manager, Front-End Design Lead
Bassam Mutawak	Back-End Design Lead
Elizabeth Ankrah	Testing & Evaluation Lead
Minh Quan Do	System Design Lead
Advisors: George Mason University	
Dr. Nathalia Peixoto	*Electrical & Computer Engineering Department*
Dr. Qi Wei	*Bioengineering Department*
Sponsors: Weinberg Medical Physics LLC	
Dr. Lamar Mair	Physicist
Dr. Chad Ropp	Physicist
Olivia Hale	Bioengineer
Dr. Irving Weinberg	Chief Executive Officer

Table of Contents

1. Problem Definition:

Despite recent advancements in research, major limitations are holding back clinical implementation of nose-to-brain magnetic particle delivery. These include physiological obstructions to particle transport, magnetic field strength attenuation as particles travel to deeper targets, and impracticality of manually operating magnet arrays in real-time. Weinberg Medical Physics is constructing an MRI-guided, electropermanent magnetic control system (MCS) to address many of these issues.

Automation of MCS operation is key for effective implementation as manual manipulation of magnet arrays during surgeries is highly impractical. Magneto serves as a software precursor to the MCS, laying the foundation for automation behavior, control flow, and user interaction. It will provide relevant, portable user-interface and backend features for the MCS, as well as automate and improve the accuracy of magnetic object translation via WMP's 4-coil system.

A. Objectives

1. Design a software control interface whose features are both relevant and easily portable to WMP's future electropermanent MCS. Some of these features include a user-friendly and fully functional graphic user interface (GUI), intuitive path drawing controls, delivery automation protocol, and data collection protocol.

2. Improve control of WMP's coil-based hardware to support future pilot experiments (for the MCS and otherwise). This includes automating control of the hardware to eliminate human error and improving particle translation accuracy of translation via a robust and reproducible translation model.

B. Requirements

GUI

- Must be coded in C++ via Qt API (The Qt Company, Espoo, Finland)
- Must be able to stream and display optical images from a camera at > 30FPS
- Must consistently detect the particle through all streamed frames
- Must be able to calibrate for hardware orientation under the camera
- Must be able to receive, display, load, and save 2D path input from the user
- Must discretize user-specified path based on a specified interpolation amount
- Must include operation controls for 1) start/pause, 2) stop, and 3) force stop options
- Must collect and export data to an output file with no gaps in collection
- Must run efficiently on WMP's proprietary computers
- Must automate the entire particle delivery operation

Particle Control

- Must be able to control multiple hardware components (motor controllers and solenoid coils) simultaneously
- Must allow for bidirectional hardware communication with GUI
- Must be implemented via an operating system independent control scheme
- Must enable consistent and accurate particle translation along a user-defined path

C. Metrics

GUI

- **Software Speed** - measured via component lag times determined through strategically placed elapsed timer functions within the GUI code; major values include but are not limited to the times (with nanosecond precision) for:
    - Startup
    - Camera Connection
    - Motor Controller Connection
    - Starting/Pausing System Operation
    - Applying Changes Made in Settings Window
    - Stopping Further Hardware Execution

- **Software Functionality** - measured by calculating the Task Completion Rate (taking the number of functional tasks completed successfully divided by the total number of tasks). A functional task is defined as interacting with a GUI component produce a desired output (ex. pressing "Connect Camera" button results in camera images being streamed and displayed on the GUI in real-time).

$$Task\ Completion\ Rate\ =\ \frac{Number\ of\ Tasks\ Successfully\ Completed}{Total\ number\ of\ Tasks}$$

Particle Control

- **Motor Controller Communication Rate** - defined as the time between two consecutive commands sent to the motor controllers; measured via strategically placed elapsed timer functions within the GUI (with millisecond precision).

- **Simultaneous Command Execution Delay** - defined as the time between two simultaneous commands (one to each motor controller); measured via strategically placed timers within the GUI (with millisecond precision).

- **Consistent Particle Path Translation** - defined as the standard deviation between multiple path completions of an identical desired path; measured via standard deviation of the particle's location throughout the path (with micrometer precision).

2. Final Design Description

This section describes the final design of the major project areas. Rationale for the chosen designs is provided and, where applicable, alternative designs are described in order to characterize the decision making process for certain project areas.

A. GUI Final Design & Description

Objective: To satisfy all the established GUI requirements and achieve high-performing metrics (see previous Problem Definition section). Every UI component must contribute to an intuitive user experience. Major areas of consideration include:

- **UI Layout**
- **Setup Procedure**
- **Path Drawing,**
- **Operation Control and Automation**
- **Data Collection**

Fall 2018 GUI Assessment: The GUI from the previous semester met basic requirements but was limited in its functionality (i.e. path drawing convenience, image segmentation integration) and its design intuition. It featured 3 full-sized windows (main, path drawing, and particle detection). Each window was cluttered with various buttons (some unnecessary) and had large lag times for displaying, making UI navigation time-consuming. Because the camera viewport displays were different sizes between the windows, several data fields needed to be resized when shuffled across the code. This resulted in convoluted code and generated several bugs.

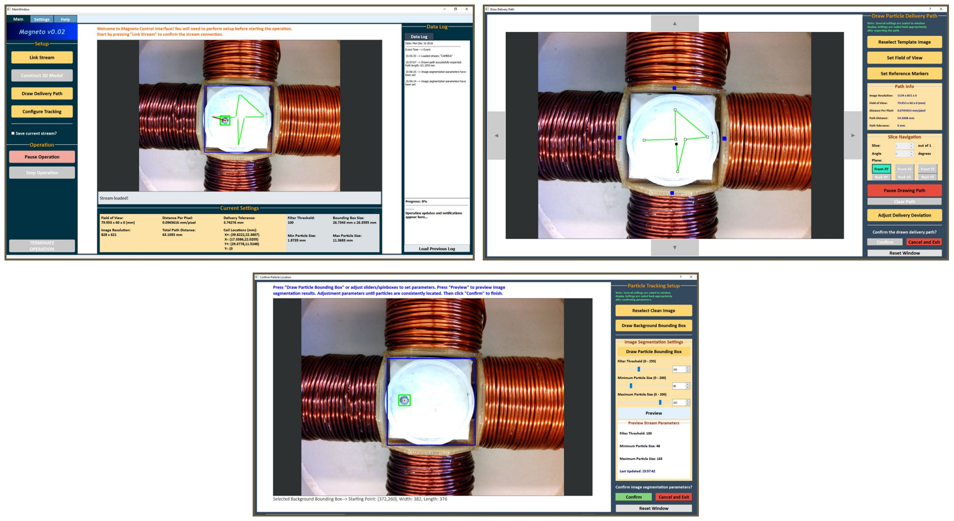
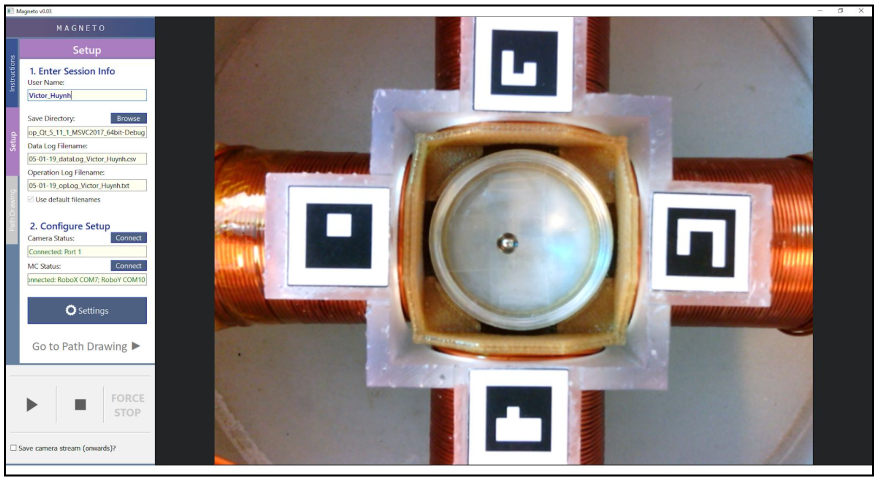


Figure 1. Fall 2018 GUI Windows (main window - top left, path drawing window - top right, particle detection window (bottom). From a functionality standpoint, 3 full-sized windows was unnecessary.

UI Layout

Based on the team's assessment of the Fall 2018 GUI, the following layout objectives were established:

1. Maximize screen space for camera image display
2. Keep all other display components contained in one defined screen region (not surrounding the camera image display on multiple sides)
3. Have operation control buttons always visible to the user
4. Minimize the number of windows that need to be opened/viewed



Setup

1. Enter Session Info

User Name: Victor_Huynh

Save Directory: op_Qt_5_11_1_MSVC2017_64bit-Debug

Data Log Filename: 05-01-19_dataLog_Victor_Huynh.csv

Operation Log Filename: 05-01-19_opLog_Victor_Huynh.txt

☑ Use default filenames

2. Configure Setup

Camera Status: Connected: Port 1

MC Status: onnected: RoboX COM7; RoboY COM10

☐ Save camera stream (onwards)?

Figure 2. New GUI layout. Functions are condensed to the left side of the screen, allowing for increased space reserved for stream display.

The new UI design now utilizes a 3-tab tabwidget (Instructions, Setup, Path Drawing) on the left side of the screen. This condenses the majority of GUI functionality into a single defined region without introducing clutter (as the user simply switches between tabs). As a result, the number of necessary full-sized windows is reduced to just 1 (the main window) and the space reserved for the camera stream display is increased. The tab functions are:

- <u>Instructions</u>: This tab lists in-order the general steps a user needs to take to set up, start, and complete a delivery operation.

- <u>Setup</u>: This tab is where the user 1) enters in his/her information, 2) specifies the save directory and name of output data files, 3) confirms software connection to the camera and motor controllers, and 4) accesses the settings window.

- <u>Path Drawing</u>: This tab is where the user can load a previously saved delivery path and draw and save the current path. The user is provided instructions on how to draw (i.e. the correct mouse clicks to add/remove and drag/drop path points).

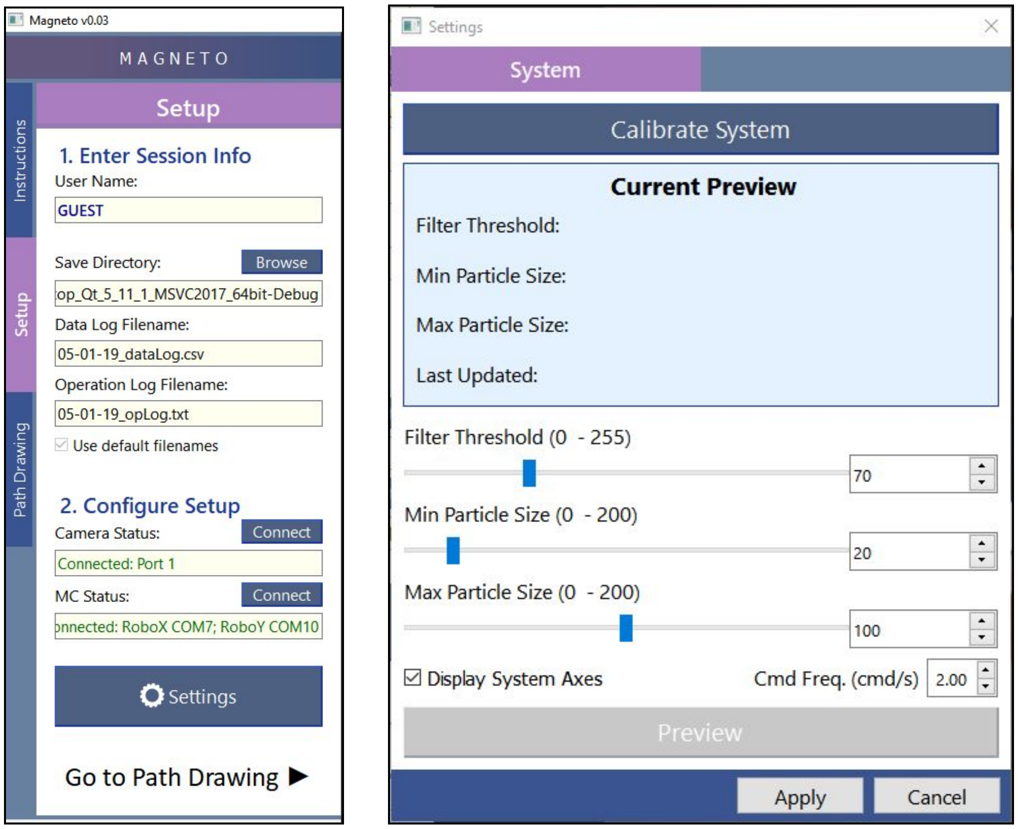
The settings window is the other openable window. It is non-full sized, making it much less intrusive, and allows the user to calibrate the system and make settings changes areas (see Setup Procedure).

The operation control panel is now in the bottom-left and is always visible to the user (see Operation Control below for more).

Setup Procedure

Before a delivery operation is started. the user must complete a brief setup procedure to ready internal data and then draw/load the delivery path. Because of the previously mentioned UI layout improvements and the one central settings window, this setup procedure has been streamlined for user convenience. The core procedure is as follows:

1. Specify save directory for export data files (or use the default directory)
2. Connect GUI to camera and motor controllers (1 button press each)
3. Open settings window and calibrate system coordinates and field-of-view (1 button press)
4. Configure particle detection parameters and confirm detection within the settings window (adjust detection parameters via sliders/spinboxes)



Setup

1. Enter Session Info

User Name: GUEST

Save Directory: op_Qt_5_11_1_MSVC2017_64bit-Debug

Data Log Filename: 05-01-19_dataLog.csv

Operation Log Filename: 05-01-19_opLog.txt

☐ Use default filenames

2. Configure Setup

Camera Status: Connected: Port 1

MC Status: onnected: RoboX COM7; RoboY COM10

Current Preview

Filter Threshold:

Min Particle Size:

Max Particle Size:

Last Updated:

Filter Threshold (0 - 255): 70

Min Particle Size (0 - 200): 20

Max Particle Size (0 - 200): 100

☑ Display System Axes

Cmd Freq. (cmd/s): 2.00

Figure 3. Setup Tab (left) and Settings Window (right).

Path Drawing

Path drawing was previously implemented via etching the path points and their connecting lines onto the image. To display the path throughout a stream of images, the program would need to perform repeated draw-calls for each displayed frame. This increases the software's CPU usage especially for paths with a large number of specified points. The etching method also prevented the user from modifying any previous path points without first removing all points that come after (since the points and their connecting lines are static pixels in the displayed image). This is an incredibly tedious constraint when attempting to change an early point in a long path.

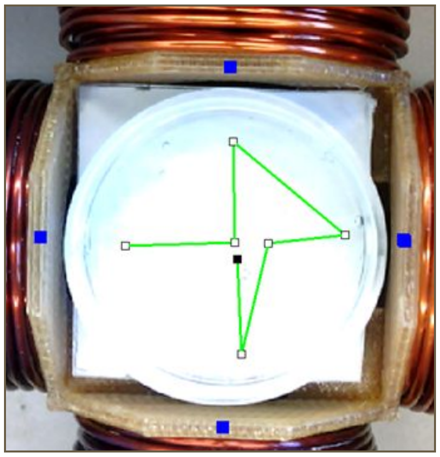
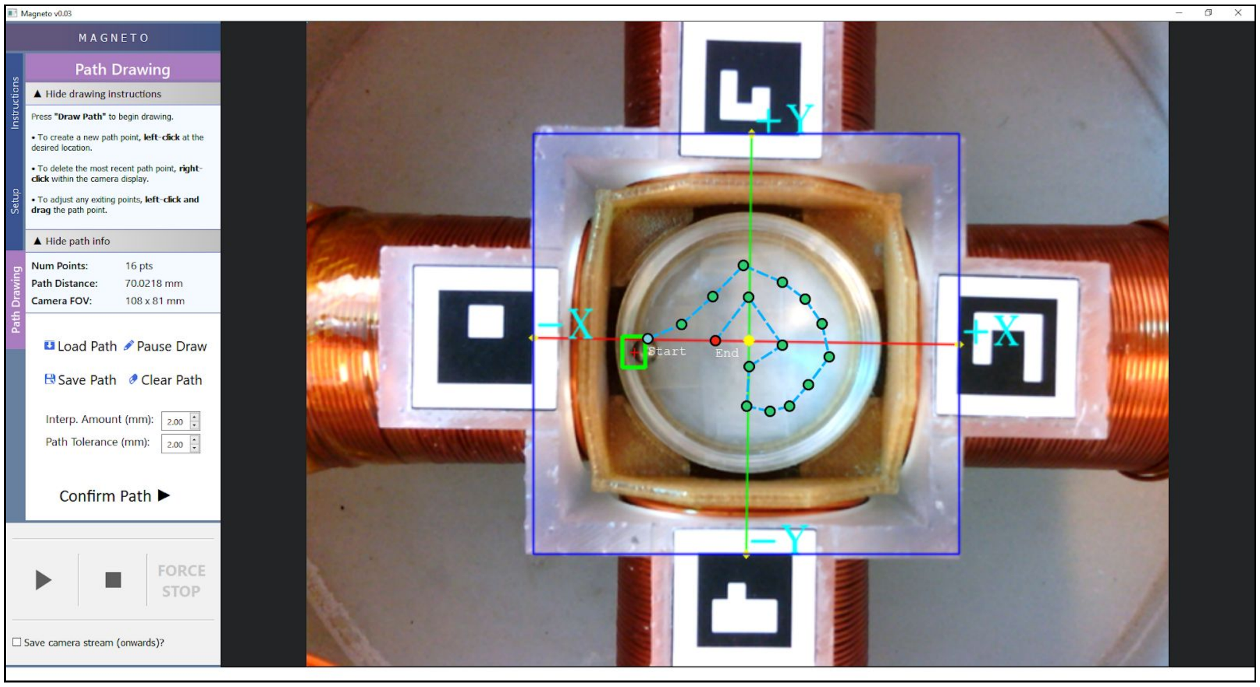


Figure 4. Old path drawing implementation uses image etching.

Path drawing is now implemented via the use of a graphics scene overlayed on top of the displayed image/stream of images. In the code, each path point is now a graphics object that can be dragged and dropped (no longer a part of the image itself), allowing for more robust and convenient editing.

A new feature is the ability to load and save paths. A drawn path can be saved to a file (using the custom ".path" extension) which stores the path's physical coordinates (in order to remove dependence on camera zoom). After coil localization is performed (see Image Segmentation Module Final Design & Description section) reverse transformations are applied to a loaded path, allowing it to be displayed correctly on the image regardless of hardware orientation under the camera. This feature greatly increases convenience and reproducibility of delivery operations.

The user is now able to edit two path-related fields: Interpolation Amount and Path Tolerance. Interpolation Amount is the amount of path discretization. For example, setting this field to "2mm" will generate path points in between the drawn markers such that no points are over 2 mm distance from each other. Path Tolerance is the acceptable distance of the particle from any path point such that it would be considered at the point. For example, setting this field to "1.5 mm" means that once the particle is within 1.5 mm of the current target path point, the current target path point will be updated to the next point along the path.



Press **"Draw Path"** to begin drawing.

- To create a new path point, **left-click** at the desired location.
- To delete the most recent path point, **right-click** within the camera display.
- To adjust any exiting points, **left-click and drag** the path point.

Num Points:	16 pts
Path Distance:	70.0218 mm
Camera FOV:	108 x 81 mm

Interp. Amount (mm): 2.00
Path Tolerance (mm): 2.00

☐ Save camera stream (onwards)?

Figure 5. New path drawing implementation uses graphics scene overlay. Notice the path-related fields on the left tab.

Operation Control and Automation

WMP specified 3 operation controls desired for their MCS.

1. **Start/Pause** - Allows the user to start/pause/resume the operation. During the paused state, the user is allowed to modify certain settings (e.g. particle detection parameters, hardware command frequency).

2. **Stop** - Halts the operation, properly ending hardware execution. A "Stop" command is sent automatically if the particle reaches the end of the delivery path (which indicates a successful operation).

3. **Force Stop** - Immediately halts the operation with improper closeout to hardware and the GUI. This is used only for emergency situations.

These features have been implemented within the Operation Control Panel. Currently in the code, "Stop" and "Force Stop" are implemented the same in that they both involve the termination of an operation timer (whose timeout would signal the next translation command) in order to halt hardware execution. For the purposes of the 4-coil magnet array system, this is more than adequate for immediate stoppage. WMP's future MCS will be capable of implementing "Force Stop" as a more abrupt killswitch to its various hardware components.

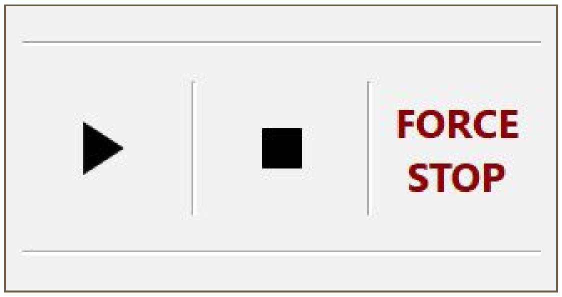


Figure 6. Operation Control Panel implements the 3 controls as iconed push buttons.

Operation automation is non-conflicting linear sequence of events. Within the high-level process listed below, there are many safeguards within the code that regulate the automation procedure.

1. The particle's current location within the image is acquired and transformed into physical coordinates.

2. The system compares the particle location to the target point of the delivery path (which may be updated if reached). If no errors are detected, the system give a go-ahead to compute the next translation.

3. The next translation command is computed by processing certain location inputs through matrix calculations (weights of Neural Network) and feedback elements (e.g. assessing if the previous translation resulted in undershooting).

4. The motor controller hardware receives the translation command and activates the solenoid coils to perform the translation.

5. Steps 1 - 4 are repeated at every timeout of the operation timer (typically set to 100s of milliseconds).

Data Collection

Collected data includes both numerical fields related to both traversal and hardware.

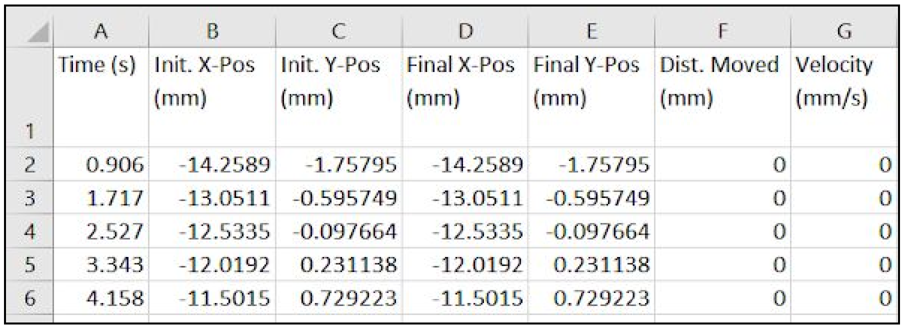
Traversal fields include:

1. Timestamp
2. Particle location pre-translation (x and y physical coordinates)
3. Particle location post-translation (x and y physical coordinates)
4. Particle distance moved (since the previous timestamp)
5. Particle velocity (since the previous timestamp)

Hardware fields include:

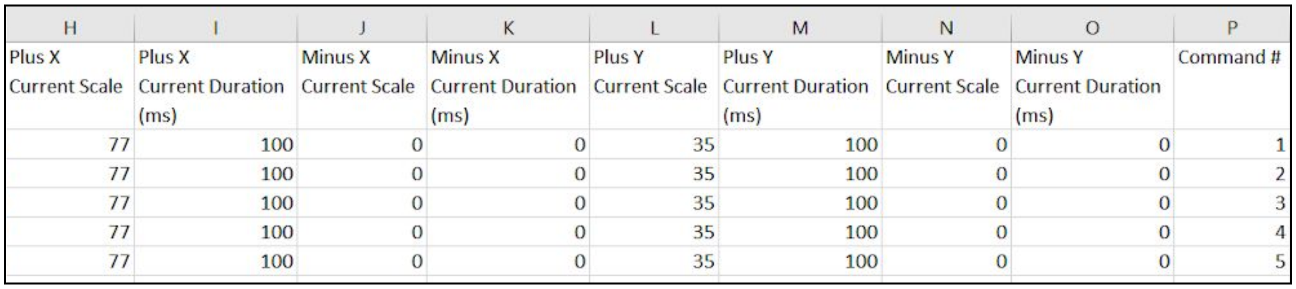
1. Outputted Current Scale (for each of the 4 coils)
2. Duration of Outputted Current Scale (for each of the 4 coils)
3. Total Number of Commands Executed (throughout entire delivery)

Collected data needs to be saved in a convenient, easily portable file format. The **".csv"** extension was chosen due to its accessibility and simplicity. CSV files can be opened and edited in a variety of text editors and programs such as Notepad and Microsoft Excel. The files can also be easily loaded into and parsed in MATLAB and can clearly distinguish between numeric and non-numeric data.



	A	B	C	D	E	F	G
1	Time (s)	Init. X-Pos (mm)	Init. Y-Pos (mm)	Final X-Pos (mm)	Final Y-Pos (mm)	Dist. Moved (mm)	Velocity (mm/s)
2	0.906	-14.2589	-1.75795	-14.2589	-1.75795	0	0
3	1.717	-13.0511	-0.595749	-13.0511	-0.595749	0	0
4	2.527	-12.5335	-0.097664	-12.5335	-0.097664	0	0
5	3.343	-12.0192	0.231138	-12.0192	0.231138	0	0
6	4.158	-11.5015	0.729223	-11.5015	0.729223	0	0

Figure 7. ".csv" file snippet of traversal data fields.



H	I	J	K	L	M	N	O	P
Plus X Current Scale	Plus X Current Duration (ms)	Minus X Current Scale	Minus X Current Duration (ms)	Plus Y Current Scale	Plus Y Current Duration (ms)	Minus Y Current Scale	Minus Y Current Duration (ms)	Command #
77	100	0	0	35	100	0	0	1
77	100	0	0	35	100	0	0	2
77	100	0	0	35	100	0	0	3
77	100	0	0	35	100	0	0	4
77	100	0	0	35	100	0	0	5

Figure 8. ".csv" file snippet of hardware data fields.

B. Image Segmentation Module Final Design & Description

1. Particle Localization Method

Objective: To consistently detect the the particle location within streamed optical images. The particle location must be detected with maximal time delay < 35 ms so as to not slow down camera frame acquisition. For system versatility, the segmentation method should be highly resistant to light pollution present within the image.

Particle Localization Schemes: Several localization methods (i.e. color thresholding, kalman filter, hough circles) were briefly evaluated against the objective above. More intensive assessment was conducted on a smaller pool of methods that exhibited strong initial promise. These methods are:

- Image Subtraction: A preoperative image in which the particle is absent (the “background” or “clean” image) is subtracted from later images in which the particle is present. The resulting different image is then segmented within the area of interest (petri-dish region) and filtered to attain particle location.

- Template Matching: A template image of the particle is used to find particle location by 2D convolution. The template slide is “slid” over the entirety of incoming images. A pixel comparison is made between the template image and each convolved region. Depending on the comparison method used, the particle location is extracted from either the region with the largest similarity or the region with the smallest similarity.

- Hough Circles: This is a feature extraction technique for circle detection within an image. This is done by applying hough gradient to find instances of objects, given parameters to detect a circle. In preprocessing, gaussian blur is applied to reduce noise, followed by applying canny edge detection to better define instance outlines. Once the image is preprocessed and the hough gradient is ran, and the algorithm works to vote upon the best representative circle.

	Criteria				
Weight	5	3	3	2	
Method	Computational Efficiency	Resistance to Light Pollution	Few Required Components	Supported Documentation	Weighted Score
Image Subtraction	4	5	5	5	60
Template Matching	3	5	2	5	46
Hough Circles	4	5	3	5	54

Table 1. Particle localization methods weighted decision matrix.
A method score of 5 optimally meets criteria while a score of 0 does not meet criteria.

Table 1 above is a weighted decision matrix comparing the three candidate particle localization schemes. Initial testing revealed template matching and hough circles localization methods to be slightly less computationally efficient and more inaccurate compared to image subtraction. The optimal localization method was found to be Image Subtraction. However, one cumbersome downside was that before each operation, the particle needed to be manually removed from view so that a new background image could be captured (as any previously taken clean images will have different hardware orientation, lightning conditions, etc.).

Image Subtraction Implementation: To eliminate the need for manually removing the particle from view, we implemented a more dynamic method of acquiring an adequate background image. At the same time that the user calibrates the coordinate system in the setup procedure, the particle is translated from coil to coil 8 times. 8 images are collected, one after each translation. The images are then averaged to mitigate the presence of the particle (which is at a different location in each image) and synthesize a sufficiently clean background image (Figure 7).

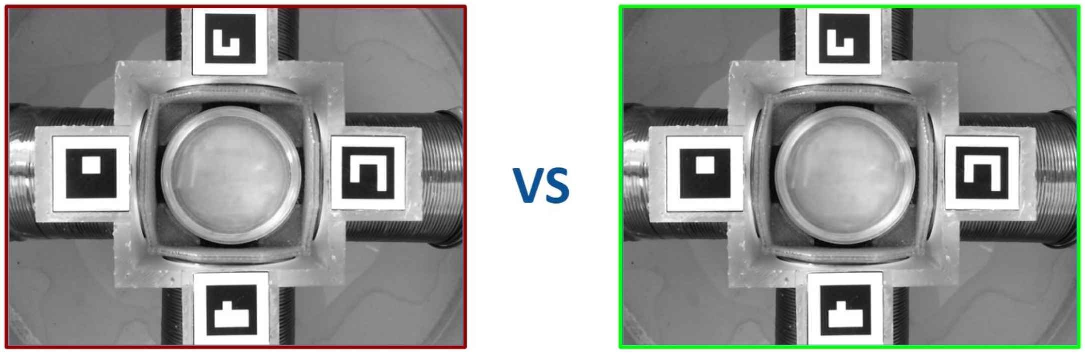


Figure 9. Background image without particle in view (left) vs. synthesized background image from image averaging (right).

2. Solenoid Coil Localization Method

Objective: To consistently map and calibrate a cartesian coordinate system based upon the solenoid coil locations. This is remove dependence on the hardware orientation under the camera. Each of the 4 coils defines an axis of this coordinate system. For a smooth user experience, the maximal time delay for calibrating the coordinate system per frame must be $<$ 190 ms. As with particle localization, this method must be highly resistant to light pollution.

Solenoid Coil Localization Schemes: Multiple coil localization methods were tested for adherence to the above objective. Because all 4 coils in the array are visually identical, each tested method involved distinct physical markers being placed on each coil. The methods evaluated were:

- Color Thresholding: A colored marker is placed on each coil. Knowing the marker color values in HSV space, the captured image is thresholded 4 times (once for each marker) to obtain their locations. Pairs of opposing markers define an axis. The intersection of these axes denote the origin of the coordinate system (0,0).

- ArUco Markers: A special fiducial marker (OpenCV's ArUco) is placed on each coil. Each marker is a square composed of a black border with a distinct inner binary pattern. The method applies an adaptive threshold to obtain black borders, contours the image for candidate markers, and then rejects candidates based on an internal filter. The binary pattern of each remaining contour is then analyzed and the four corners of a marker (defining its location) are returned. From here, the coordinate system is calibrated the same way as in Color Thresholding.

	Criteria				
Weight	5	3	3	2	
Method	Computational Efficiency	Resistance to Light Pollution	Consistency	Supported Documentation	Weighted Score
Color Thresholding	5	2	2	5	47
ArUco Markers	3	5	5	5	55

Table 2. Coil localization methods weighted decision matrix.
A method score of 5 optimally meets criteria while a score of 0 does not meet criteria.

Table 2 above is a weighted decision matrix comparing each coil localization method against factors critical for our application. While color thresholding proved to be slightly more computationally efficient, poor consistency and resistance to light pollution were major drawbacks. ArUco marker detection was found to best adhere to the objective above.

Markers are placed flat on a simple 3D-printed platform inserted over the coil array to ensure clear visibility by the camera. See below (Figure 10) for images of the actual print. A CAD render can be found attached to this document (see Appendix B). Note that the detected markers sit over each of the coils and are not flush with the petri-dish border. This adds ~5 mm when calculating the detected particle's distance from coil. At a maximum, the particle's distance from any coil is ~40 mm and is ~5 mm at a minimum.

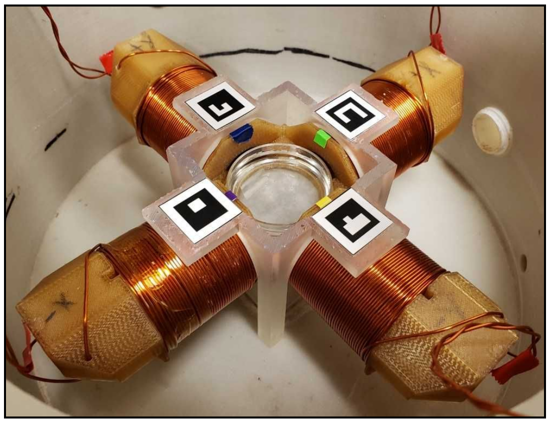
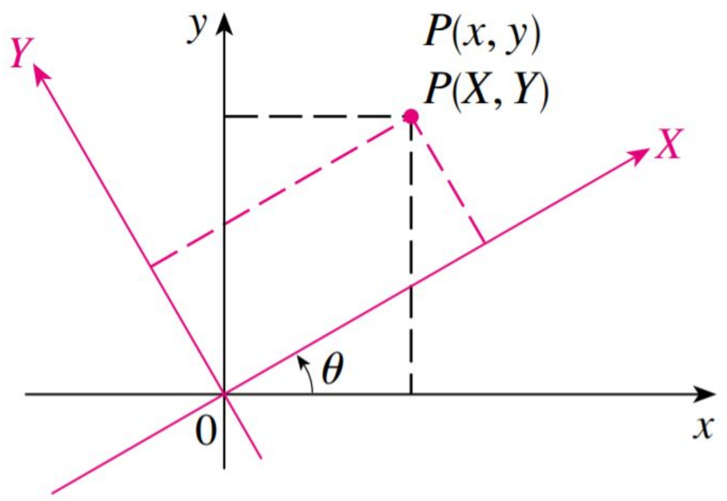


Figure 10. 3D-printed ArUco marker structure within coil array housing.

<u>ArUco Marker Implementation:</u> During initial calibration, each coil marker's detected location was across 20 images are stored and averaged to mitigate location noise/outliers. Camera field-of-view (FOV) was calculated by dividing the physical distance between two coils on the same axis (known beforehand) by the number of pixels between those coils:

$$Distance\ Per\ Pixel\ = \frac{Physical\ Distance\ Between\ Two\ Coil\ (mm)}{Pixel\ Distance\ Between\ Two\ Coils\ (pixels)}$$

Using the marker locations, the axes of the cartesian coordinate system are established (line intersection of +X and -X coil markers denotes X-axis; line intersection of +Y and -Y coil markers denotes Y-axis). The x-coordinate of the origin point is determined by equating the X and Y axis line segments in point-slope form. Plugging the x-coordinate back into either line equation results in the origin's y-coordinate. All locations in the image can be transformed to physical coordinates with respect to this coordinate system (Figure 11). Should the camera FOV or the hardware orientation change, the user simply reinitiates calibration process.



$$X = x\cos(\theta) + y\sin(\theta)$$
$$Y = -x\sin(\theta) + y\cos(\theta)$$

Figure 11. Mapping to a rotated coordinate system. In the above equations, inputs x and y are transformed into outputs X and Y. Picture and mathematics used courtesy of [3].

C. Particle Translation Modeling Design & Description

1. Particle Translation Protocol

Objective: Establish a methodology for accurately translating the magnetic particle through a user-defined path using a solenoid coil array. Given the current state of operation (i.e. current and desired particle locations) as an input, this methodology must return (for each solenoid) the current scale (unitless, motor controller command instruction) and current duration (in milliseconds) required to translate the particle.

Modelling Schemes: Three methods were considered for particle translation modelling. They were evaluated on their ability to handle multiple input parameters and their reproducibility for different scenarios.

- Surface Mapping: Given a dataset containing instances of particle translation, non-linear regression would be used to identify relationships between key features. This scheme would be highly specific to the object of interest as well as any other critical factors (i.e. liquid viscosity, hardware setup); however, the real strength of this approach lies in its ease of implementation and deployment. Previous use of this method in Fall 2018 proved to be inaccurate (possibly due to suboptimal data collection at the time) and required additional arbitrary modification within the code to output feasible translation commands.

- Mathematical Modeling: This involves formulating a model using the Biot-Savart Law for determining the strength and direction of a magnetic field created by a solenoid coil and using Stoke's Law to account for the drag force exerted on a spherical object moving through a viscous fluid. While this approach would lead to a more complete understanding of the forces that affect particle translation, it would be extremely difficult to account for the various physical factors in our system, such as the 4 coils in the array not being uniform (each hand-wrapped and unique from each other).

- Regression Neural Network: A "black-box" approach that would allow the model to account for multiple factors affecting particle translation. Given a dataset containing instances of particle translation, a neural network can be trained using key features (i.e. current location, desired location, distance from the coil to move towards, etc.) to predict the required current scale to translate the object of interest. One downside to this approach is that training a neural network model would require a tremendous amount of data that must be collected and processed. Additional development time would need to be spent on establishing data collection and data cleaning/preprocessing procedures.

After discussion with our advisors, a mathematical modeling approach was determined as infeasible due to the various physical factors affecting both the particle and 4-coil system. As a result, data-driven approaches such as a neural network or surface-mapping were deemed more preferable options.

In preprocessing the data for the each approach, Scikit-Learn's Mutual Information Regression algorithm was used select the most relevant features for accurate particle translation (Table 3). However, features selected by Mutual Information Regression seemed to only account for 1-dimensional particle manipulation.

For example, if the model is being built for the +X coil, Mutual Information Regression would suggest that we use the initial X-coordinate, the final X-coordinate, the distance from the +X coil, and the distance from the -X coil. Because none of the parameters are related to the Y-axis, these features may be incapable of accurate particle manipulation in 2D. Therefore, two neural network architectures were closely examined: one architecture would have four inputs (the features selected by Mutual Information Regression) and the other would use five inputs (based on empirical evaluation) that would theoretically account for 2-dimensional translation.

	Mutual Information Regression	**Empirical Evaluation**
Determined Features **(mm scale)**	- Initial Axis Coord (X or Y) - Final Axis Coord (X or Y) - Distance to Axis Coil - Distance from Opposing Axis Coil	- Initial X Coord - Initial Y Coord - Final X Coord - Final Y Coord - Distance to Axis Coil

Table 3: Features determined to be relevant by Mutual Information Regression (left) and Empirical Evaluation (right).

The development the neural network can be broken down into four distinct phases:

1. Data Collection
2. Preprocessing
3. Training and Evaluation
4. Deployment.

Data Collection Procedure: The first step in developing the model is to collect data. Due to the large amount of the data points that must be collected, manual data collection would have been infeasible, and therefore, an automated data collection procedure was implemented. Furthermore, to simplify the collection protocol, no more than one coil was activated simultaneously. Figure 12 below depicts the automated protocol. Once the particle is detected, the protocol checks the distance of the particle from any of the four coils. If the particle is within 20 mm of a coil, the algorithm will automatically select the opposing coil; otherwise, if the particle is not within 20 mm of a coil, the algorithm will randomly select one of the 4 coils. From there the selected coil will be activated with a random current scale value. A total of 30,000 instances over a period of roughly four weeks were collected using this protocol (11,000 data points were deemed as useable). See Figure 13 for visualization of collected particle translations.

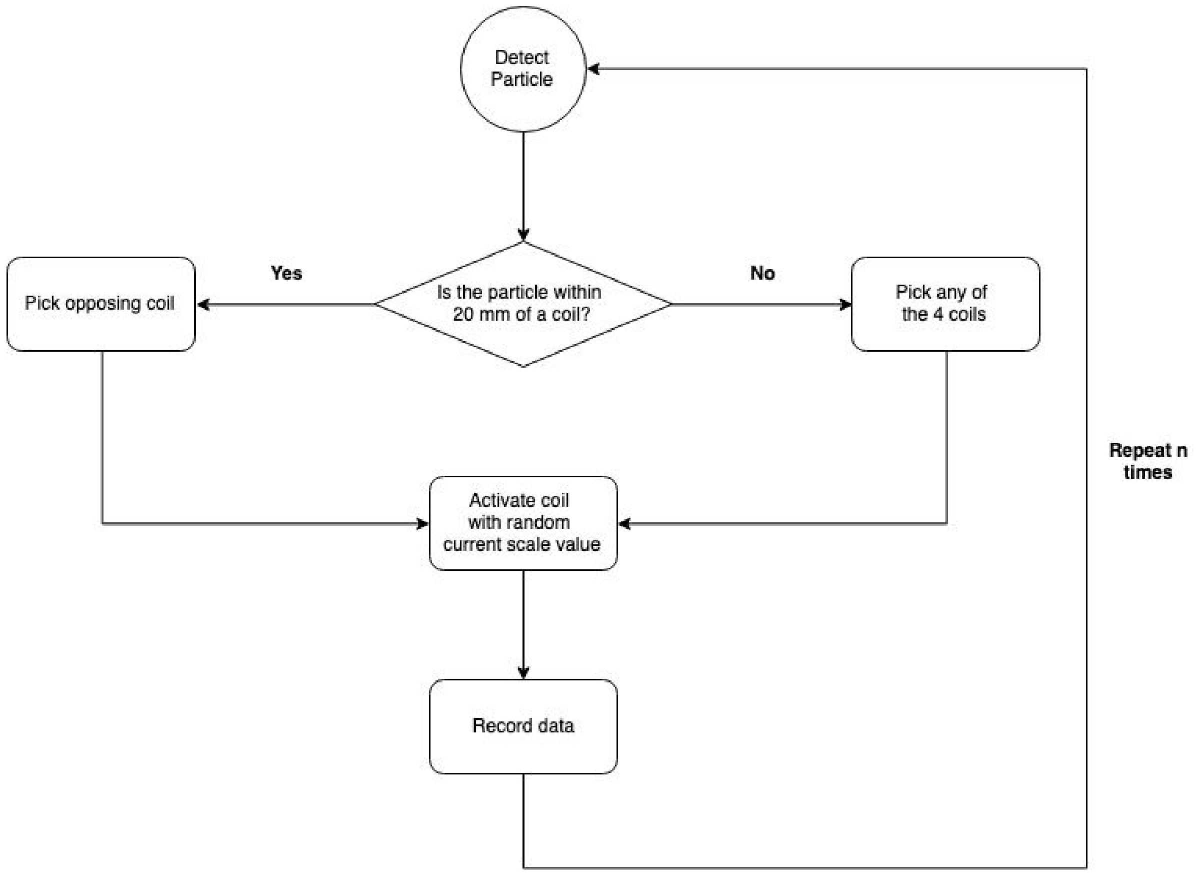


Figure 12. Standardized automated data collection protocol.
Flow begins at particle detection and repeats for a fixed N number of times.

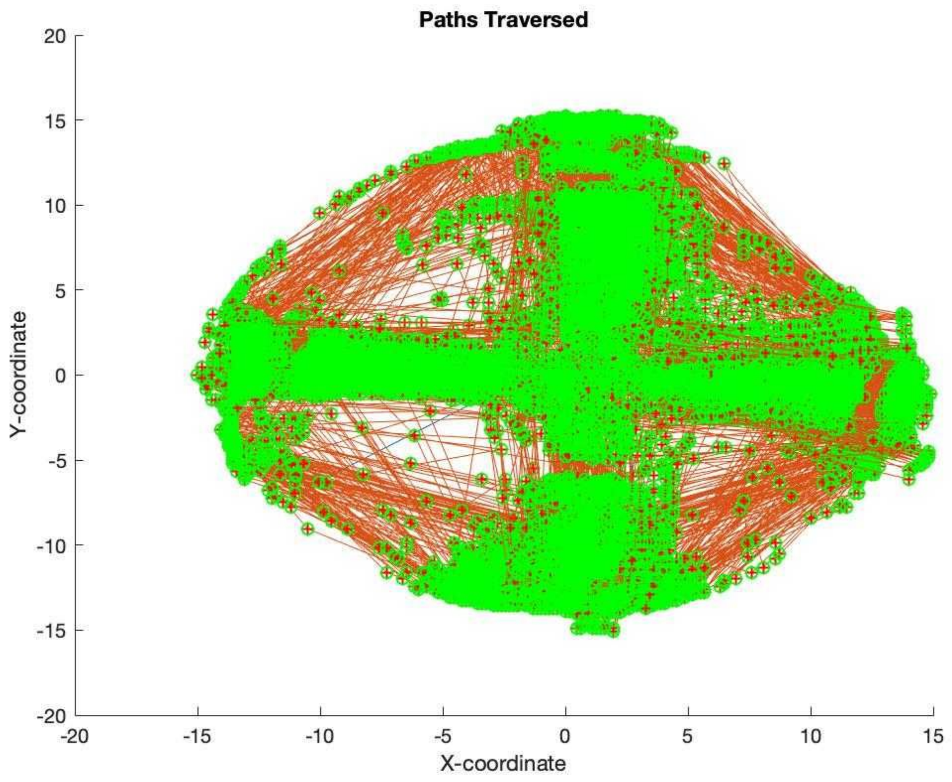
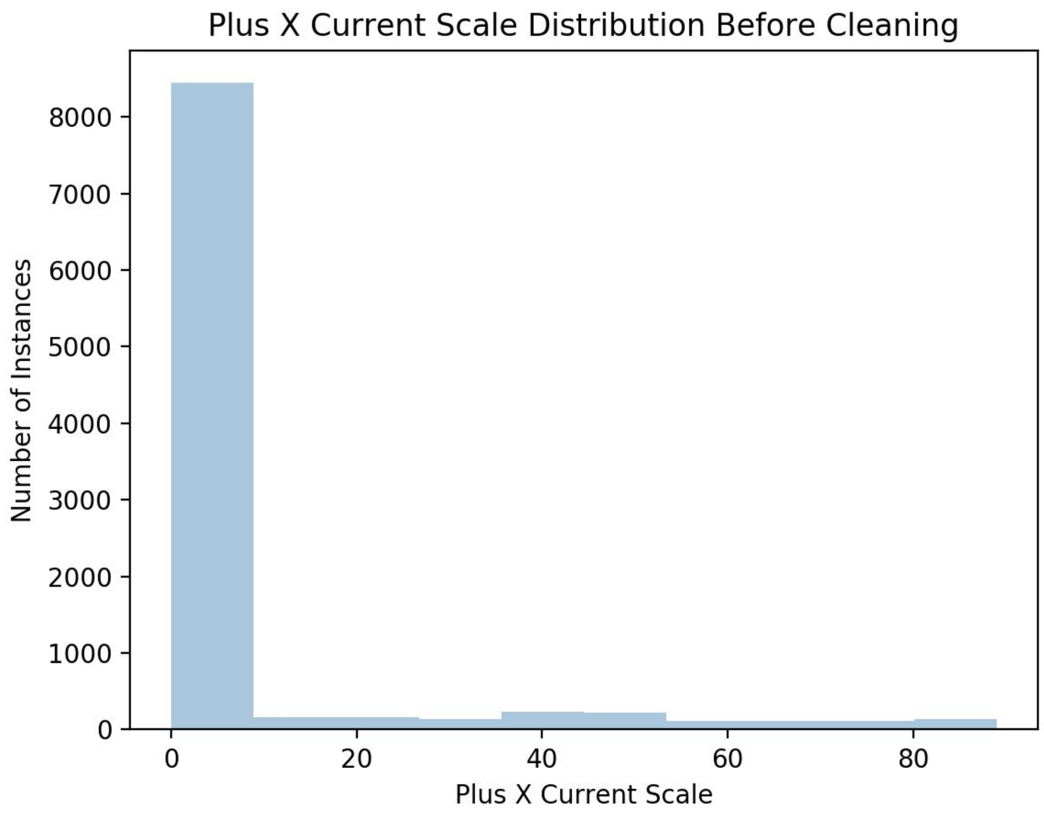


Figure 13. Starting points (green circles), endpoints (red "+"), and path (red lines) of the particle for all trials of the data collection

Due to hardware limitations, the static 4-coil array design made it very difficult to move the particle uniformly throughout the petri dish during the data collection process. As a result, the particle mostly stayed in a cross pattern with clusters around the coils and there are areas ("dead zones") in which there are very few data points collected as evidenced by the lack of green circles and red pluses (shown above in Figure 13).

<u>**Preprocessing:**</u> Before the data can be used to train the model, the data must be processed as follows:

1. Remove all values where current scale is equal to zero (i.e. if the data set is meant to be used to train the model for the +X coil, then all instances where the +X coil was not activated are removed). This is done because oftentimes the coil is not activated during data collection and therefore a value of zero is recorded. Therefore, using these instances when training would introduce noise into the model. As evidenced by the distribution plots most of the instances in the data set before cleaning had a current scale value of zero. As a result, when those instances were removed, only 15% of the dataset remained. In most situations, removing such a large portion of the data set is concerning; however, in our application those instances can be safely disregarded since the model will never be used when the current scale is zero.



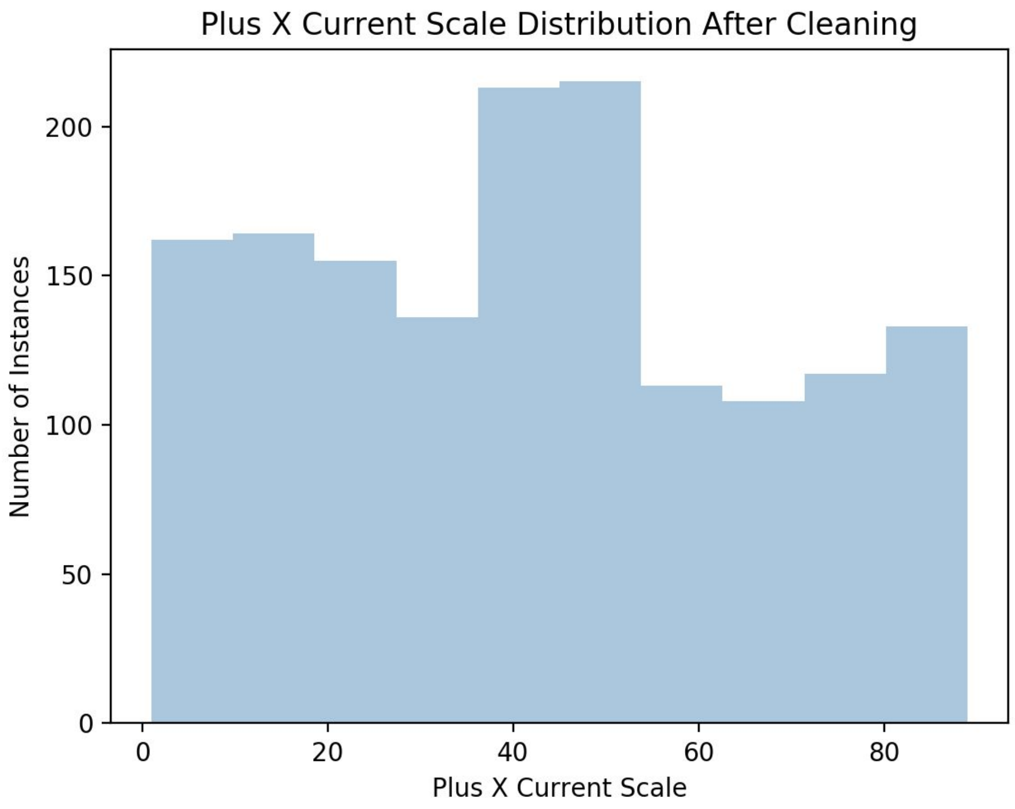


Figure 14. Data set before (top) and after (bottom) cleaning procedure in step 1 of preprocessing.

2. Once the dataset is cleaned, all model inputs were normalized to fall within a common scale. In our case, the values of initial X-coordinates, initial Y-coordinates, final X-coordinates, and final Y-coordinates are in the range of [-17.5, 17.5] mm, however the distances from the coils were in the range of [5, 40] mm. For our purposes, Z-Score normalization was implemented on all input features.

3. Partition the dataset up into the training set and the testing set. 80% of the data set was used for training and 20% was used for testing.

4. Select critical features from each instance to reduce dimensionality optimize performance. Scitkit-Learn's Mutual Information Regression feature selection was applied to determine the most relevant features to be used as inputs to the neural network. The feature selection process determined that the most relevant features were the initial X-coordinate (mm), initial Y-coordinate (mm), final X-coordinate (mm), final Y-coordinate (mm), and the distance from coil (mm). Distance from coil is defined as the euclidean distance between the current particle location and the coil of interest.

Architecture Comparison 5-Input vs. 4-Input Neural Network:

Model	**Features Selected By**	**Pros**	**Cons**
4-input	Mutual Information Regression	Less complex than 5-input model	Does not account for 2D travel
5-input	Empirical Evaluation	Can account for 2D travel	More complexity due to additional parameter May suffer from "curse of dimensionality"

Table 4. 5-Input vs. 4-Input neural network architecture comparison.

4-input: The features of the 4-input neural network is selected by Scikit-Learn's Mutual Information Regression. The advantage to using the 4-input neural network is its simplicity in comparison to the 5-input model. However, the disadvantage to using the 4-input neural network lies in its inability to account for 2-dimensional travel.

5-input: The features of the 5-input neural network were determined through empirical evaluation. Various parameters were loaded into the 5-input neural network to determine the best parameters to use during live testing. The disadvantage to the 5-input model is its complexity in comparison to the 4-input model. Additionally, it is subject to the "the curse of dimensionality", a phenomenon in which the data set has too many dimensions. This results in a sparse dataset. The advantage of the 5-input neural network is that it can account for 2-dimensional travel. This is an important feature because the model is used to traverse 2D paths.

Regression Neural Network Architecture: After careful testing and evaluation of both architectures, a 5-input neural network model was selected as the final translation method due to the increased propensity to account for 2D manipulation. Four neural networks (one for each coil) were trained to predict the required current scale value using the features selected in preprocessing step 4. Each network includes one input layer with five input nodes corresponding to initial X-coordinate (Xo), initial Y-coordinate (Yo), final X-coordinate (Xf), final Y-coordinate (Yf), and the distance from the coil (mm) as inputs. Furthermore, each network has 1 output representing outputs: X+ current scale, X- current scale, Y+ current scale, or Y- current scale. Each architecture has one hidden layer with 10 hidden neurons and a sigmoid activation function. The Levenberg-Marquardt training and optimization algorithm was implemented in MATLAB using the Deep Learning Toolbox. The network architecture can be found below (Figure 15).

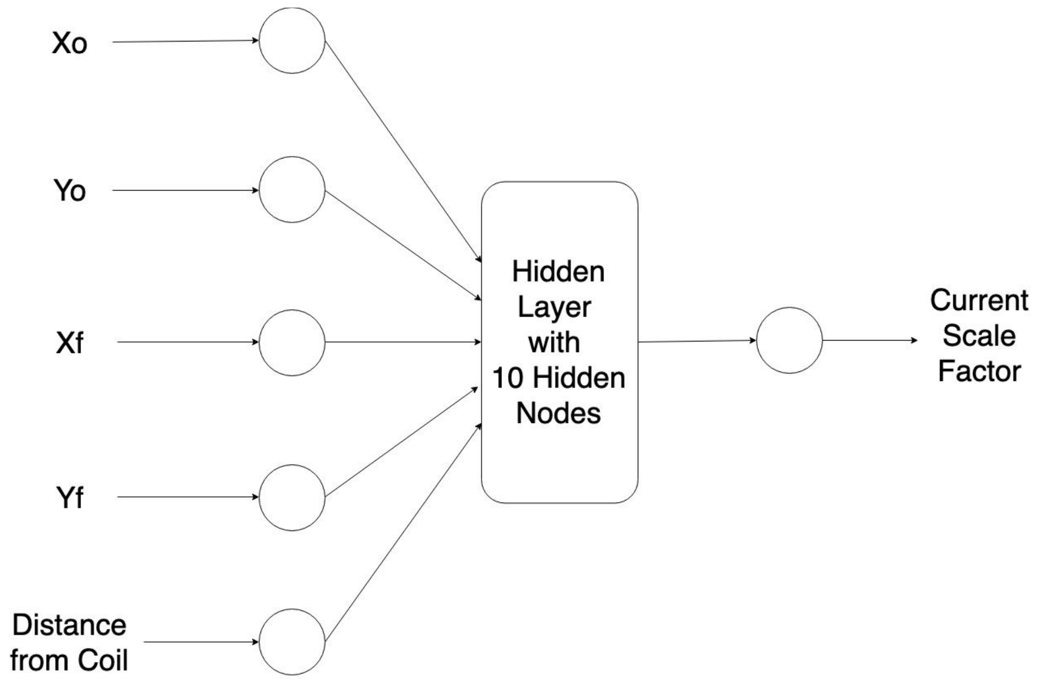


Figure 15. Neural network architecture: consisting of 5 inputs (left), one hidden layer (middle) and output (right).

Architecture Comparison 5-Input Neural Network vs. Surface Fitting:

Model	**Pros**	**Cons**
5-Input Neural Network	Can account for 2D translation Can account for a large number of parameters	Much more complex to implement "Black Box" Require a large amount of data to train model
Surface Fitting	Much easier to implement Produces an equation for the surface, therefore not a "Black Box" model Less computational power is needed to train and deploy the model	Cannot predict current scale values for when the ball magnet must travel more than 40 mm across the petri dish or when the ball magnet is more than 10 mm away from the coil. Can only account for two parameters Does not account for 2D translation

Table 5. 5-Input Neural Network vs. Surface Fitting translation model comparison.

5-Input Neural Network: Advantages of the neural network are: it can account for a large number of parameters and it allows for 2D translation. Larger data requirement and increased training complexity are two disadvantages to this approach.

Surface Fitting: A major advantage to a surface fitting model is the short development time required for each coil. However, this model's inability to produce a valid current scale value (current scale value must be between 0 and 127) with varying inputs was an early indicator to performance. 2D particle manipulation was also not feasible for the surface fitting model due to its inability to account for more than two parameters (distance from coil and euclidean distance to travel).

Surface Fitting Architecture: Four surface fitting models (one for each coil) were also implemented in MATLAB for comparison against the neural network approach. Each model uses the distance from the coil of interest and the distance the ball needs to travel as inputs and outputs the current scale value. The surface was created by using MATLAB's polyfit function. An example surface fitting model and can be found below (Figure 16).

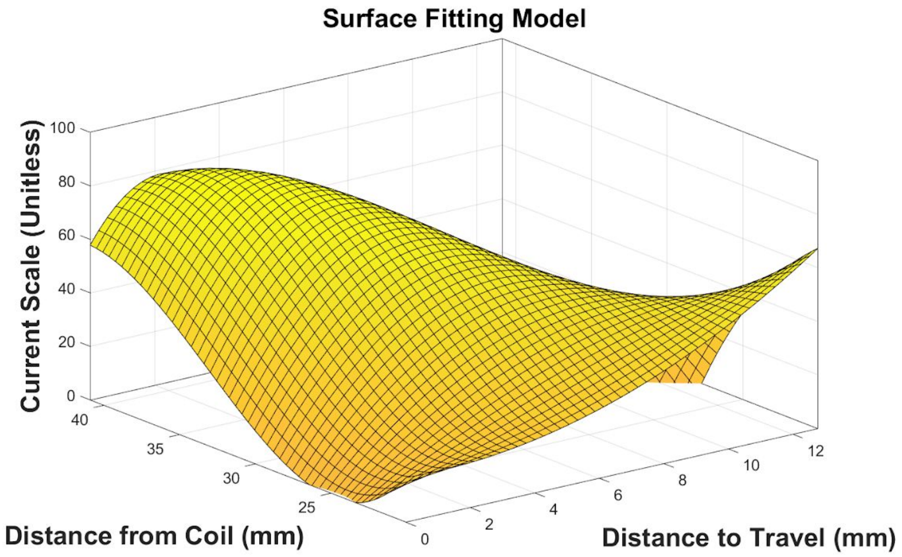


Figure 16. Example surface fitting model with inputs (bottom) and output (top left).

Deployment: Each trained neural network model's weights and biases were exported to our C++ code base in order to be used with the actual system. A matrix library in C++ (Eigen) was used to initialize the weight and bias matrices for each network. Each trained surface fitting model's resulting nonlinear fit equation was directly implemented in our C++ code base.

D. Hardware Final Design & Description.

1. Hardware-Software Communication

Objective: Establish seamless communication between graphical user interface and motor controllers. Communication between components must be bi-directional with sufficient motor control complexity amongst other factors.

Roboclaw Motorcontroller Control Schemes: All avenues of communication with the sponsor supplied motorcontrollers (Basic Micro Roboclaw 2x60A Motor Controllers) were evaluated with respect to the objective for hardware-software communication above. The possible methods of hardware control evaluated are described below:

- Radio Control: Motors can be controlled using a hobby RC radio and subsequent RC receiver.

- Analog: Motors can be controlled using a potentiometer or a filtered PWM signal.

- Simple Serial: Motors can be controlled using TTL level byte commands. Format is 8 bits, no parity bits and 1 stop bit. Interfacing can be done using a microcontroller or PC using a level shifting circuit. One-way communication, roboclaws can only receive data.

- Packet Serial: Buffered bidirectional simple serial mode. Allows for finer control of each motor controller. Roboclaw motorcontrollers can both receive and transmit data. Interfacing can only be done through USB or microcontroller unit (MCU).

	Criteria					
Weight	5	4	4	3	5	
Methods	Bidirectional Communication	Current Resolution (Dual channel mode)	Data transfer speed	Motor control complexity	Can be directly controlled using a PC	**Weighted Score**
Radio Control (RC)	0	4	5	3	3	60
Analog	0	5	5	3	2	59
Simple Serial	0	2	2	4	5	53
Packet Serial	5	3	2	4	5	82

Table 6. Roboclaw motor controller control schemes weighted decision matrix.
A method score of 5 optimally meets criteria while a score of 0 does not meet criteria.

Table 6 above is a weighted decision matrix comparing all avenues of motorcontroller control schemes. The method that complies with the hardware-software communication objective and is the optimal choice for this application is packet serial control.

Packet Serial Control Schemes: Given the optimal motor controller control scheme, various implementations of the packet serial control method were evaluated for safety, efficiency, and relative ease of implementation amongst other criteria. A critical feature of each method is that it must support simultaneous control of each motor controller. The implementations evaluated were:

- USB Control: Both roboclaws are individually connected to PC-GUI via USB. A provided C# library can be used to send instructions to each roboclaw independently. Simultaneous control with variable time is accomplished by implementing multi-threading within the GUI.

- Arduino Control (Two-workers): Three arduinos are used in this process. One arduino is connected to PC-GUI using USB to microUSB cable and acts as the "master" arduino which receives instructions from the PC-GUI. Two "worker" arduinos are each connected to individual motor controllers. The master arduino receives instructions from PC and delegates commands to the appropriate roboclaw. Simultaneous control is accomplished by isolating the required busy wait of each roboclaw to separate arduinos.

- Arduino Control (One-worker): Two arduinos are used in this process. One "master" arduino is connected to PC-GUI via USB and is also connected to a single roboclaw. Another arduino is then connected to the master and to the remaining roboclaw. Instructions are sent from PC-GUI to the master arduino. Master arduino receives instructions and sends commands to worker arduino. Both master and worker arduino send instructions to respective roboclaws for simultaneous execution. Similarly to the two-worker scheme, simultaneous control is accomplished by isolating the required busy wait of each roboclaw to distinct arduinos.

- Raspberry Pi Control: One raspberry pi is used in this process. The PC-GUI and raspberry pi communicate using a shared web server. The raspberry pi receives instructions from the PC-GUI and then communicates with roboclaws via serial ports for command execution. Simultaneous control is accomplished by implementing multi-threading within the raspberry pi.

	Criteria							
Weight	2	3	5	5	4			
Method	Ease of Implementation (relative)	Offline Environment	Safety	Efficiency	Upgradability	Weighted Score	Components List	Method Cost
USB Control	4	5	5	5	3	85	- 2 USB 2.0 to Micro USB cables @ $5.39/unit	$10.78
Arduino Control (Two-workers)	4	5	2	2	4	59	- 3 Arduino MEGA boards @ $9.99/unit - 1 USB 2.0 to Micro USB cable @ $5.39/unit	$35.36
Arduino Control (One-worker)	4	5	3	3	4	69	- 2 Arduino MEGA boards @ $9.99/unit - 1 USB 2.0 to Micro USB cable @ $5.39/unit	$25.37
Raspberry Pi Control	2	2	3	3	2	48	- 1 Raspberry Pi 3 Model B @ $34.49/unit	$34.49

Table 7. Packet serial control schemes weighted decision matrix.
A method score of 5 optimally meets criteria while a score of 0 does not meet criteria.

Table 7 on the previous page is a weighted decision matrix comparing all implementations of packet serial control. Alongside each implementation is the complete components list and subsequent method cost (component prices are accurate as of 05/10/19). The packet serial control method found to comply with the hardware-software communication objective as well as meeting all control criteria (with the lowest method cost) was USB Control. A wiring diagram (Figure 17) depicting all hardware connections for USB Control can be found below.

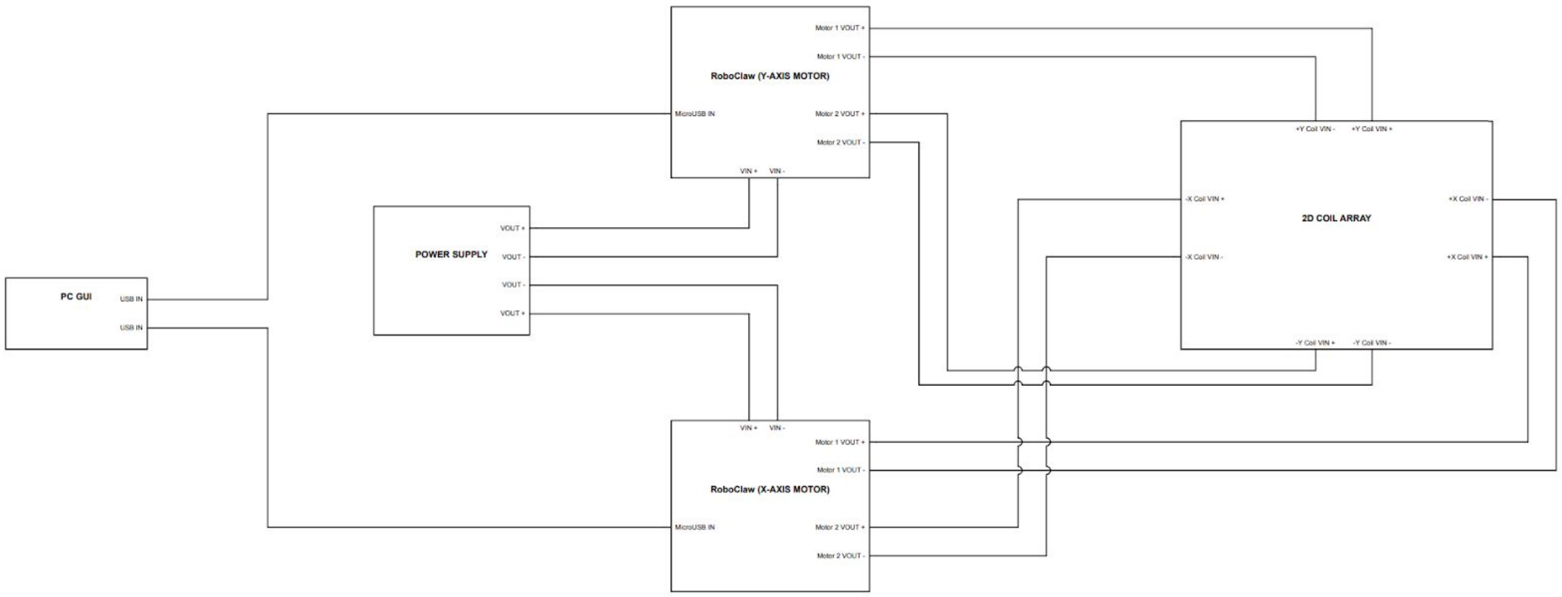


Figure 17. USB Control Wiring Diagram

3. Testing Methodology and Analysis

This section describes the final testing and evaluation (T&E) performed on all project components, including obtained results and their analysis and discussion.

A. GUI Testing

Objective: Attain and assess measurements for software speed and functionality.

1. Testing Description:

A. Functionality

This was evaluated both informally and formally. Informally, several checks were conducted by running all sections of the code repeatedly over the entire software development life cycle to evaluate output and perform debugging. This continuous development prevented the threat of bug accumulation as the project timeline advanced. Formally, towards the end of the project timeline, a checklist of GUI widgets and components was used by the team as a whole to objectively evaluate the final GUI capabilities and deficiencies and produce a Task Completion Rate.

B. Speed

Elapsed timers with nanosecond precision were placed strategically within the code to measure the lag times for several major GUI functions. These functions include:

- Startup
- Camera Connection
- Motor Controller Connection
- Displaying Settings Window
- Applying Changes Made In Settings Window
- Starting/Pausing System Operation
- Stopping Further Hardware Execution

2. Results (Relevant graphs/tables):

A. Functionality - The checklist covered 37 UI components. All 37 components were found to be fulfill their intended purpose, resulting in a 100% Task Completion Rate for the component scope (see Appendix H).

B. Speed - Table 9 below displays the recorded lag times. Measurements are shown in milliseconds (but still with nanosecond precision) for better time comprehension of the software's responsiveness.

Lag Time Measurements of Significant Functions		
Function	**Avg Lag Time (ms)**	**Standard Deviation (ms)**
Startup	1287.1280 ms	16.5940 ms
Camera Connection	987.3840 ms	182.7573 ms
Motor Controller Connection	10.8091 ms	0.5863 ms
Opening Settings Subwindow	52.3760 ms	2.1065 ms
Applying Changes Made In Settings Window	0.1628 ms	0.1003 ms
Starting/Pausing System Operation	1.9845 ms	0.5634 ms
Stopping Further Hardware Execution	0.0932 ms	0.0808 ms

Table 9. Lag times for significant GUI functions. The times for the most intensive functions (Startup and Camera Connection) were ~1000 ms.

3. Analysis and Discussion:

A. Functionality - Because the checklist evaluation was conducted very close to the end of the project timeline (when GUI functionality was finalized), the 100% Task Completion Rate was expected.

B. Speed - The most intensive operations (Startup and Camera Connection) took ~1000 ms. All other recorded times fell well below that (< 60 ms). Some standard deviation values were large in comparison to the corresponding lag time (such as for Stopping Further Hardware Execution), meaning that the spread was very high. However, due to how short those lag times are (completely unnoticeable by the user), this was deemed negligible.These obtained lag times support the high responsiveness of the software.

4. GUI Testing Conclusion:

Lag times were acquired as a measure of GUI speed. Due to the brevity of these times (often under 100 ms), the GUI was deemed as responsive. The functionality checklist utilized a simple, objective tool to assess GUI functionality. Improvements to this evaluation could be made for future software iterations (outside of the project) by involving non-users and expanding upon the checklist scope.

B. Image Segmentation Module Testing.

Objective: Attain a measure of consistent particle and coil marker detection over a fixed time period under different lighting conditions. All testing was performed using image subtraction with a synthesized background image for particle localization and ArUco markers for coil localization.

1. Testing Description:

A. Characterized particle localization deviation:

Recorded the detected particle location in each frame over a testing period (1.0 minute) under different lighting conditions. The particle was not moved throughout the duration of the testing period and the lighting condition was held constant. For preliminary testing parameters, three extreme lighting environments were used: dark room (6 Lux), ambient lighting (80 Lux), and directly under a light source (220 Lux). An example image of each lighting condition can be found below (Figure 18). For each testing set, a total of 50 frames were acquired and segmented to attain particle location. The resulting locations were then analyzed for standard particle deviation amongst other statistics.

B. Characterized coil localization deviation:

Recorded each detected coil location in all incoming frames over a testing period under different lighting conditions. The coil array was not moved throughout the duration of the testing period and the lighting condition was held constant. Again, as preliminary testing parameters, three extreme lighting conditions were used: dark room, ambient lighting, and directly under a light source. An example image of each lighting condition can be found below (Figure 18.). For each testing set, a total of 100 frames were acquired and segmented to attain each coil marker's location. The resulting locations were then analyzed for average deviation amongst other statistics.

2. Results (Relevant graphs/tables):

A. Particle Localization Deviation For Each Preliminary Lighting Environment:

	Light Environment		
	220 Lux (Direct Light)	80 Lux (Ambient Lighting)	6 Lux (Dark Room)
Standard Particle Deviation (mm)	0.1337	0.0521	0.3432

Table 10. Standard particle deviation for each tested lighting environment

B. Coil Localization Deviation For Each Preliminary Lighting Environment:

	Coil Marker Standard Deviations (mm)				
Light Environment	+X Coil	-X Coil	+Y Coil	-Y Coil	Average Deviation (mm)
220 Lux (Direct Light)	0.0196	0	0	0	0.0049
80 Lux (Ambient Lighting)	0	0	0	0	0
6 Lux (Dark Room)	0	0.0351	0.0523	0.0191	0.0266

Table 11. Standard deviation of each coil's detected ArUco marker under different lighting environments.

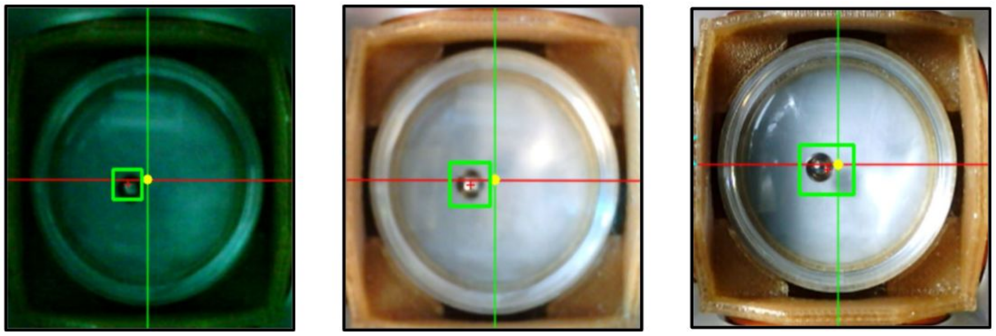


Figure 18. Detected particle location in different lighting conditions tested.
From left to right, the conditions tested are: dark room, ambient lighting, and under a direct light source.

3. Analysis and Discussion:

A. **Regarding Table 10:** There was no significant particle deviation for both the 220 Lux and 80 Lux lighting environments. For the dark lighting environment (6 Lux), the particle location could only be detected by lowering the filter threshold.

B. **Regarding Table 11**: ArUco markers were detected in each frame for all environments. Largest observed standard deviation across all tested lighting environments was 0.0523 mm (Dark Room, +Y Coil). There was no significant detected marker deviation for the 80 Lux lighting environment.

4. Image Segmentation Testing Conclusion:

Two software tests were performed under different lighting conditions to attain a measure of consistency in detecting critical features within images streamed from a USB camera. The first test measured the standard deviation of detected particle location. There was no significant deviation in particle location within the 220 and 80 Lux environments. The dark room environment required a lower filter threshold to locate the particle and resulted in the highest location deviation of 0.3432 mm. Our project application does not involve operating in low lighting conditions, so we do not see this as a limitation.

The second test measured standard deviation of each detected coil marker location. Detection of coil markers in the 6 Lux lighting environment performed the worst (out of the three environments tested) with an average deviation of 0.0266 mm across all four coil markers. Markers were detected with no significant deviation in the other two light environments (0.0049 mm and 0 mm respectively for 220 and 80 Lux environments).

C. Particle Translation Modeling Testing

1. Testing Description:

A. Comparing neural network and surface fitting regression results:

Using the collected data, several neural network architecture designs were tested with varying feature inputs, number of hidden layers, number of hidden neurons, and activation functions. The number of hidden layers was varied between 1 - 3 and the number of hidden neurons was tested at 10, 15, 20, 30, and 60. It was determined that 1 hidden layer with 10 hidden neuron was the best option for deployment. More complex architectures did not provide any significant increase in performance (measured by R^2 correlation coefficient). Both the neural network and the surface-fitting model were trained on the same training set and the evaluated using the same testing set.

B. Neural Network Architecture Traversal Comparison:

The paths traversed by the particle using a 4-input neural network model approach was compared to the path traversed using the 5-input neural network model. Both tests utilized an identical desired path and starting conditions (initial particle location, total path interpolation amount, path tolerance) and were performed in ambient lighting conditions.

C. Comparing neural network and surface fitting translation results:

The paths traversed by the particle using a neural network model approach was compared to the path traversed using the surface fitting model. Both tests utilized an identical desired path and starting conditions (initial particle location, total path interpolation amount, path tolerance) and were performed in ambient lighting conditions.

D. Neural network average traversed path:

The standard path deviation was calculated for N = 10 particle translations using a neural network modeling approach on the same desired path with identical starting conditions (initial particle location, total path interpolation amount, path tolerance) and were performed in ambient lighting conditions. All paths were averaged and a standard deviation was computed to provide a measure of consistent path translation.

2. Results (Relevant graphs/tables):

A. Comparing neural network and surface fitting regression results:

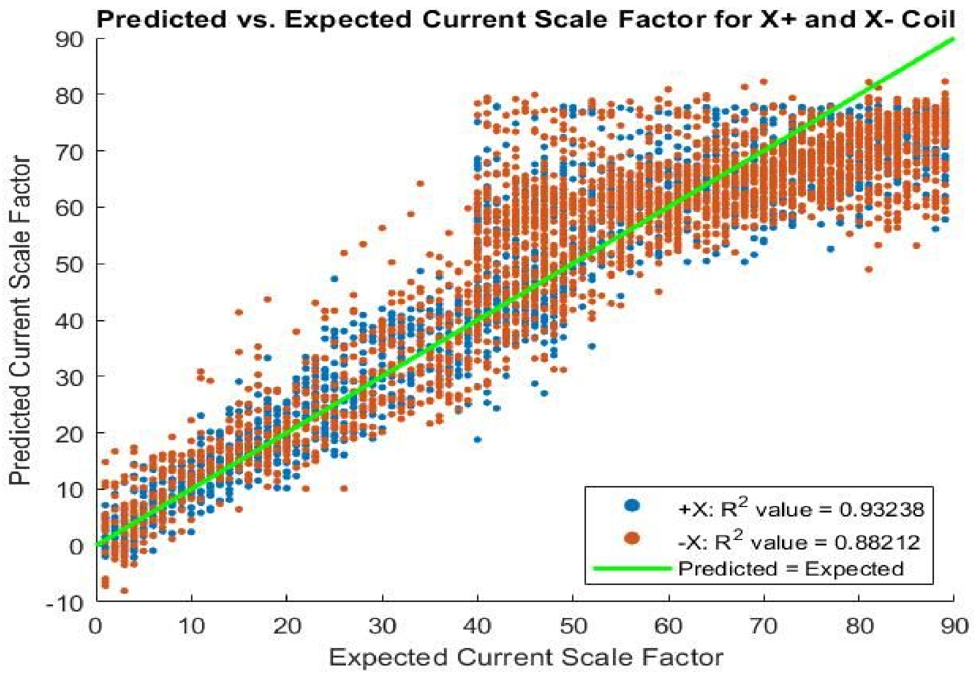


Figure 19. Predicted vs. Expected current scale with linear correlation coefficient (R^2) for 5-input neural network

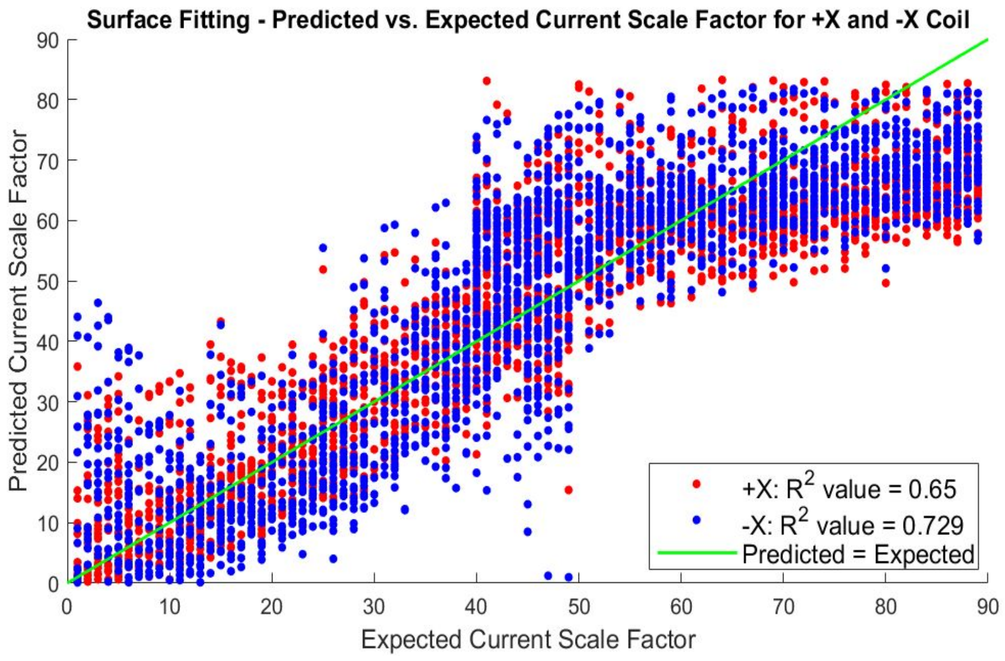


Figure 20. Predicted vs. Expected current scale with linear correlation coefficient (R^2) for surface fitting model.

	R^2 Correlation Coefficient	
Trained Coil	Surface Fitting	Neural Network
+X	0.65	0.931
-X	0.729	0.886
+Y	0.735	0.887
-Y	0.827	0.934
AVERAGE	0.73525	0.9095

Table 12. Linear correlation coefficient (R^2) comparison for surface fitting and neural network models across all four coils.

B. Comparing neural network and surface fitting translation results:

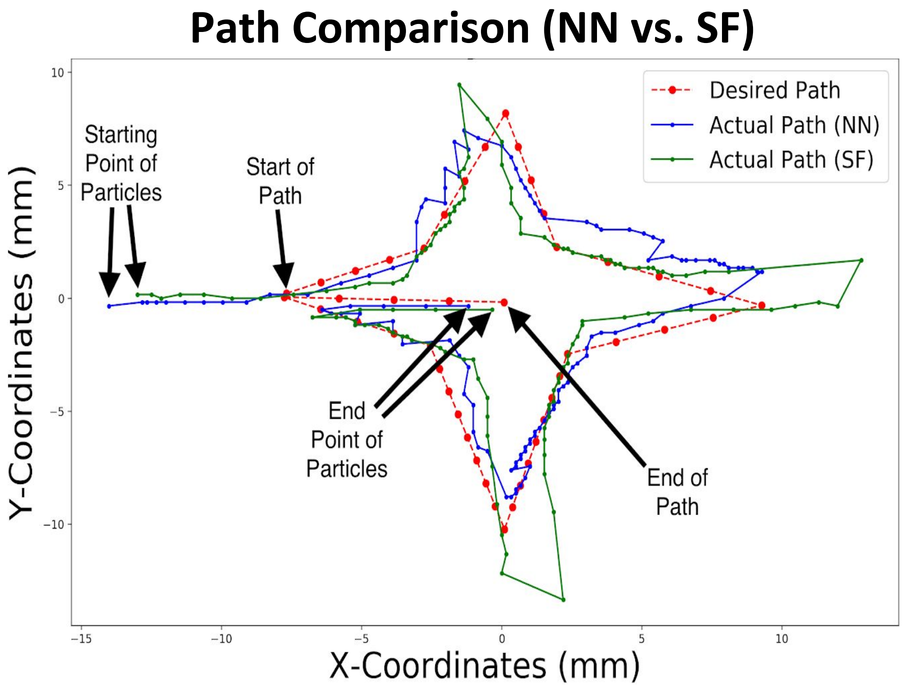


Figure 21. Path comparison between neural network (blue) and surface fitting (green) translation models across the same desired path (red).

C. Neural Network Architecture Traversal Comparison:

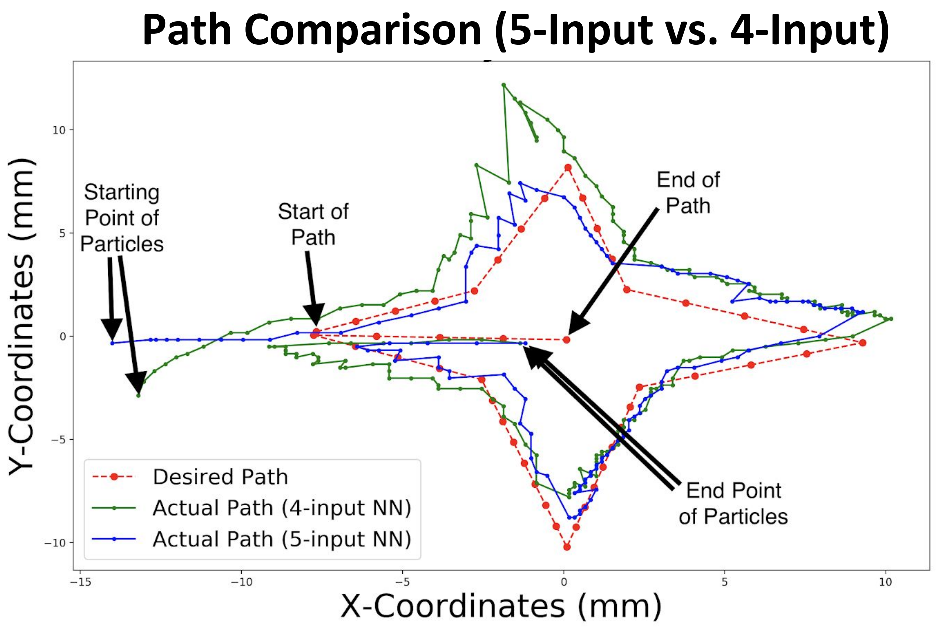


Figure 22. Path comparison between 5-Input neural network (blue) and 4-Input neural network (green) across the same desired path (red).

D. Neural network average traversed path:

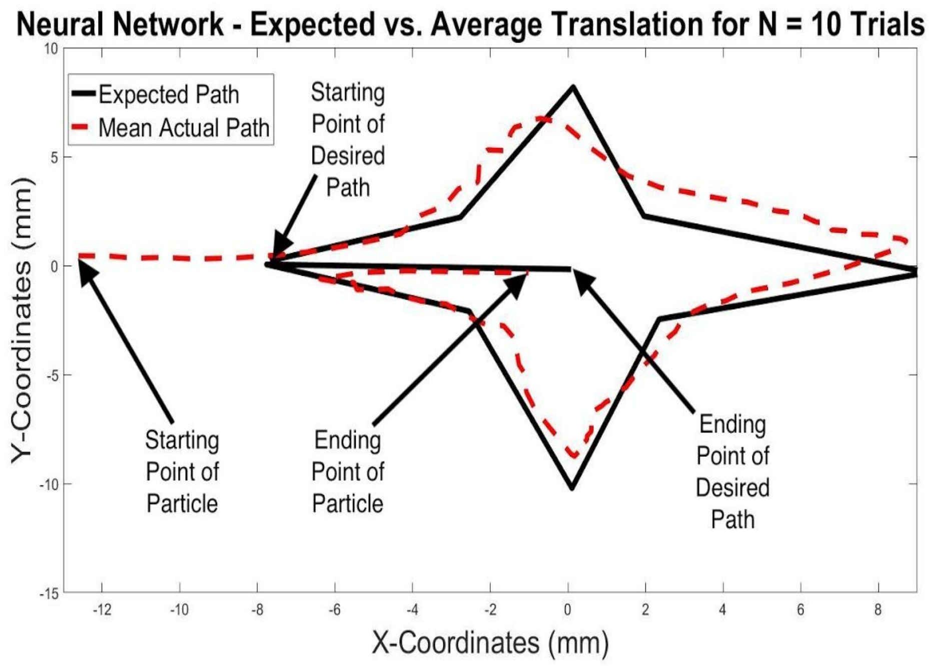


Figure 23. Average path traversed (red dashed line) using neural network model for N = 10 trials.

3. Analysis and Discussion:

A. **Regarding Figure 19, Figure 20, and Table 12:** As evident by Table 12, the linear regression coefficient (R^2) for neural network (0.88 - 0.99) was much higher across each trained coil compared to the R^2 of the surface fitting model (0.65 - 0.729). According to Figure 20, the predicted current scale value was considerably larger or smaller than ground truth. These two factors were early indicators that a neural network approach was more applicable to our design. Further testing (after deployment) was required before a final translation model was selected.

B. **Regarding Figure 21:** Both the neural network and surface fitting translation models were tested using the same starting particle location and desired path. Both models showed similar translation towards the center of the operating area however the surface fitting model would always cause the particle to overshoot closer to the coils compared to the neural network which performed consistently throughout the desired path. Coupled with the previous analysis, these results proved a neural network approach to be the superior particle translation model.

C. **Regarding Figure 22:** Once a neural network was determined to be the superior option to the surface fitting model, further architectural comparison was required before final deployment. Upon testing the 4-input model and the 5-input model, the 4-input model performed almost exactly the same as the 5-input model along every point in the path except for when the particle was close (around 10 mm) to the +Y coil. Whenever the particle was close to the +Y coil, the predicted current scale value produced by the 4-input neural network would consistently cause the particle to overshoot. Therefore, it is determined that the 5-input neural network is the best architecture to use for a particle translation model.

D. **Regarding Figure 23:** A total of 10 neural network actual particle translation paths were collected using identical desired path and starting conditions (starting particle location) across each trial. The location of the particle along the path was exported and evaluated for standard deviation and the average path. Across all trials, an average 0.85 mm standard deviation in particle location was observed.

4. Particle Translation Model Conclusion

To determine the optimal neural network design, several different network architectures were tested with varying number of hidden layers and hidden neurons. The optimal architecture was found to be 1 hidden layers with 10 hidden neurons resulting in a correlation coefficient of 0.92. The neural network translation model was found to perform more consistently across an identical desired path compared to surface fitting. The average standard deviation observed for N = 10 path translations in ambient lighting using a neural network approach was 0.85 mm. This measure of consistency was considered acceptable given the standard deviation of detecting the particle in ambient lighting (0.0521 mm) and other factors such as solution dynamics. In conclusion, a 5-input neural network was implemented as the final particle translation approach.

D. Hardware Testing

Objective: Attain a measure of consistency between hardware-software communication over a long testing period. All testing was performed with the final motor controller control method (Packet Serial, USB Control).

1. Testing Description:

A. Characterized total time for complete hardware execution:

Recorded the time for a complete hardware command execution (with millisecond precision) throughout the testing period. Time was obtained via use of strategically place elapsed timers in the code and debugging statements. Tested various applied current durations (time duration that current is being sent to hardware per software command) in multiple sessions to detect potential trends between the current duration and lag time. The current duration was tested between 75 - 200 ms in increments of 25 ms.

For each current duration value, 500 total software commands were sent with a brief 1s pause between commands (8.333 minutes per test session). Time period for complete hardware command execution was recorded for each command. One testing set was performed with only one motor controller being activated. The other testing set was was performed with activating two motor controllers simultaneously (with the same current duration range).

B. Characterized time difference between simultaneous activation of our 2 motor controllers:

Recorded the time delay (with millisecond precision) between the completion of two simultaneous motor controller commands. Like the previous test, the applied current duration was varied to detect any trends between command completion and current duration. The current duration was tested between 75 - 200 ms in increments of 25 ms.

For each current duration value, 500 total commands with a brief 1 s pause between commands were sent activating both motor controllers (also 8.333 minutes per test session). Time delay between the completion of each motor controller command was recorded for each command during the test sessions.

2. Results (Relevant graphs/tables):

A. Total hardware command execution time for one-coil (top) and two-coils (bottom) being activated:

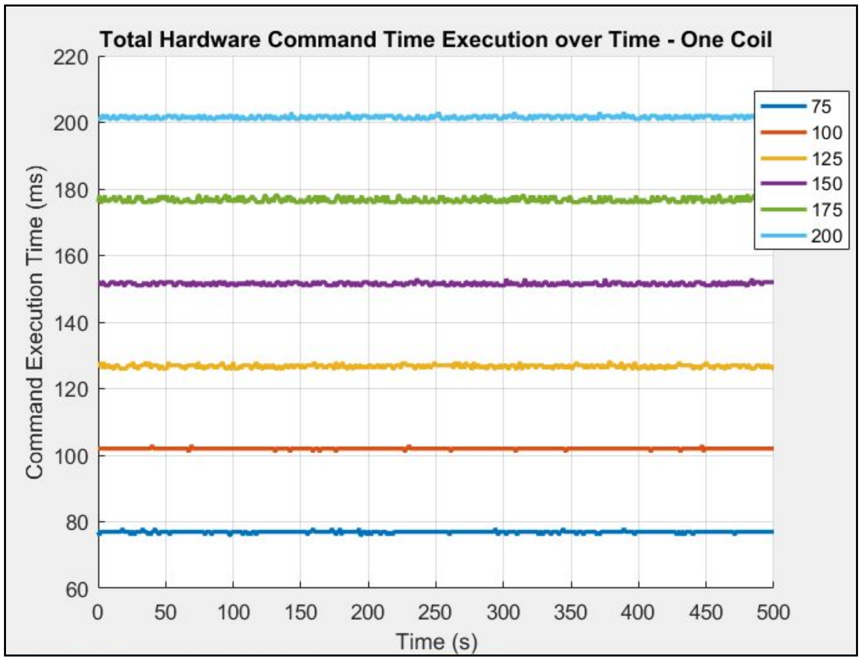


Figure 24. Hardware command execution time for one coil activation. Legend depicts total current duration used for each line.

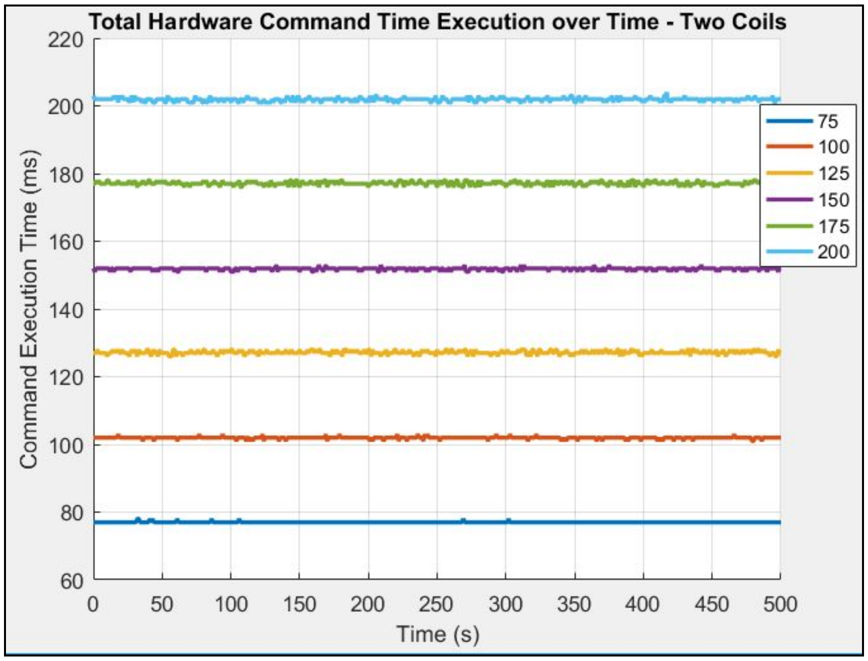


Figure 25. Hardware command execution time for two coil activation. Legend depicts total current duration used for each line.

B. Time difference between simultaneous motor controller command completion:

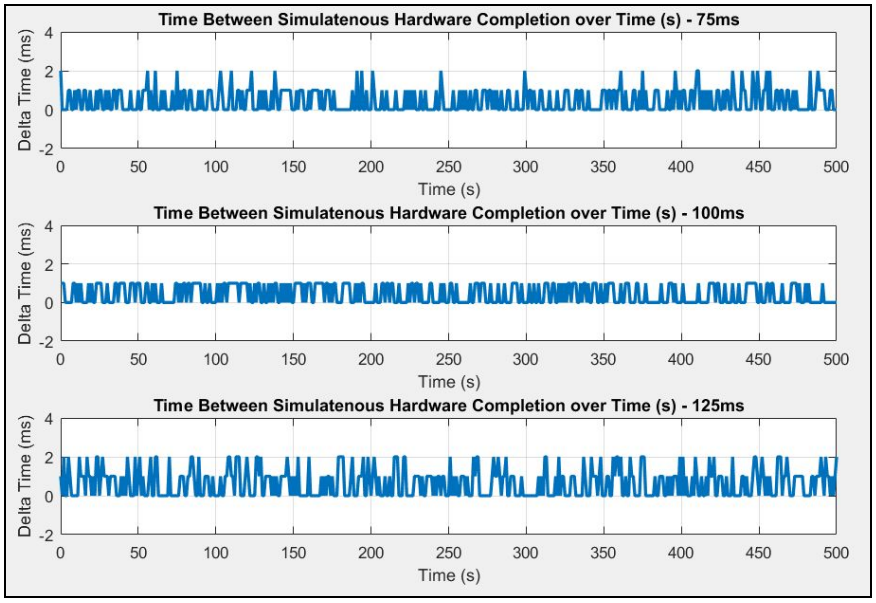


Figure 26. Time difference between simultaneous motor controller command completion for 75ms, 100ms, 125ms total current duration.

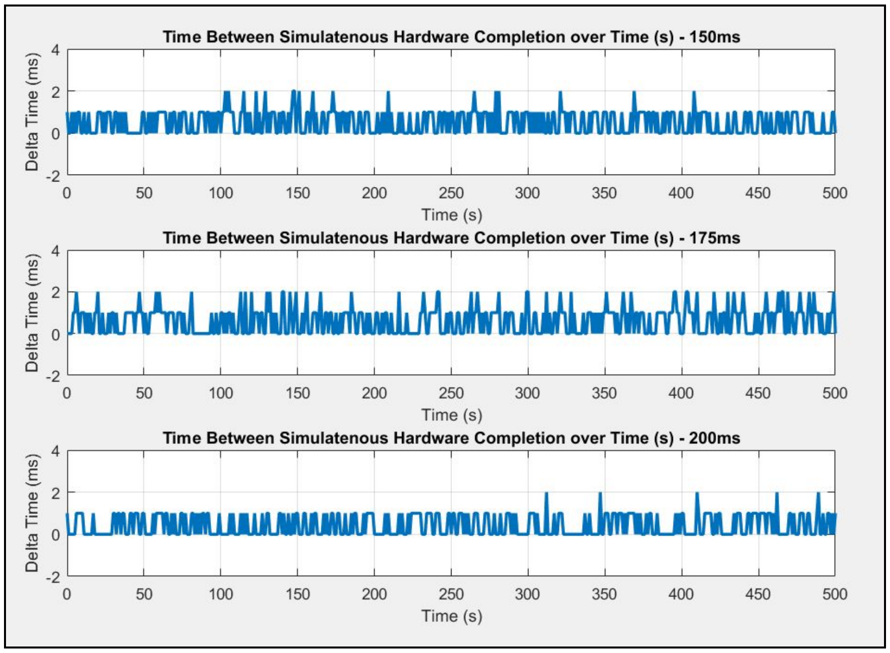


Figure 27. Time difference between simultaneous motor controller command completion for 150ms, 175ms, 200ms total current duration.

3. Analysis and Discussion:

A. Regarding Figures 24 and 25: Total hardware execution time for one-coil (top) and two-coils (bottom) being activated:

A general trend observed between both one-coil and two-coil activations is increasing the total current duration time increased the execution time standard deviation over the duration of the testing period. See table 9 below for results across both one and two coil activation tests. Across all current durations, the average standard deviation was 0.4786 ms and 0.4238 ms for one and two coil activations respectively.

	Recorded Execution Time Standard Deviation (ms)	
Desired Execution Time (ms)	**1-Coil Activation**	**2-Coil Activation**
75	0.3215	0.1330
100	0.1835	0.3189
125	0.5601	0.5583
150	0.5201	0.4059
175	0.7664	0.5735
200	0.5216	0.5531

Table 13. Recorded execution time standard deviation (ms) for all desired times (ms) tested for one and two coil activations.

B. Regarding Figures 26 and 27: Time difference between simultaneous motor controller command completion:

Similarly to the previous test, a general trend observed is increasing the total current duration time increased the time difference standard deviation over the testing period. See table 10 below for full results. Across all current durations, an average 0.5940 ms standard deviation was observed.

Current Duration (ms)	Time Delay (ms)
75	0.5940
100	0.4949
125	0.7378
150	0.5640
175	0.6605
200	0.5940

Table 14. Measured time delay between simultaneous motor controller activations across different current durations.

4. Hardware Testing Conclusion:

Two hardware tests were performed to attain a measure of consistency between hardware-software communication. The first test performed attempted to characterize the total hardware command execution time with varying current durations. Across all current durations tested, 0.4786 ms and 0.4238 ms average standard deviations were observed for the one-coil and two-coil tests respectively. While there was an increased standard deviation observed when increasing the current duration, the average standard deviation across both test sets is still relatively low. Looking ahead, we do not expect this value to affect project performance.

The second test attempted to characterize the time difference between simultaneous motor controller command completions. Ideally, given the same current duration, both motor controllers will complete instructions at the same time. Across all current durations, an average 0.5940 ms standard deviation was observed and a 0.5166 ms average time difference between simultaneous command completions. While this time difference can be further mitigated by applying a short delay (~0.3 ms) before one of the two motor controller command being sent, we do not expect an average time difference of 0.5166 ms to significantly affect project performance.

4. Societal, Ethical, Economic and Global Context Benefits

Clinically feasible non-invasive targeted drug delivery/therapy can allow for efficient uptake of drugs/payloads to target sites within the body while mitigating any adverse effects to surrounding tissues. Magnetic particle delivery has shown promise as the control mechanism for targeted drug delivery but suffers from obstacles such as magnetic attenuation and physiological obstacles to particle transport. The realization of WMP's future MRI-guided MCS serves to address these problems, but its operation must be automated via safe and effective control procedures implemented in software. Automated operation would also to reduce the workload and human error from surgeons.

Project Magneto serves as the precursor to this software control interface and lays the foundation for automated control. The enhancements made to the control of WMP's 4 coil MCS help to aid in any future pilot experiments working towards their final product. Furthermore, project Magneto acts as a testing platform for WMP to prototype particle manipulation methods in a consistent manner. Any ethical concern of the software system limiting human judgement is mitigated by an executive user control panel which includes an emergency button stopping all particle translation. Increasing safety, consistency, and automation. Magneto is a software that aids in the growth of both automated and augmented robotics.

5. Summary

The Fall 2018 prototype of Project Magneto was functional but not serviceable. While a particle could be delivered along a user-defined path, translation was not consistent and was unreliable. The software handled operation data poorly, was limited to what it could do, and was highly computationally intensive.

In Spring 2019, several enhancements have been made to the project components. The GUI design received a design and code overhaul, boosting the user experience, functionality, and computational performance. Using fiducial markers, image segmentation can automatically calibrate the hardware orientation and physical field-of-view while being highly resistant to image noise. A newly implemented particle detection protocol removes the cumbersome need for manual capture of a background image. An improved hardware communication method simplified hardware connection to the gui and allowed for simultaneous control of multiple hardware components. A neural network approach for particle translation proved to be highly consistent and reliable in manipulating the particle. Final testing and evaluation showed exceptional results in all major project components.

We conclude that project Magneto has successfully accomplished its objectives and requirements. The next step for our project is to turnover of all hardware and software systems to Weinberg Medical Physics. A software package including source code, executables, and guiding documentation will be included in the transition. If required, necessary modifications will be made to ensure smooth integration of our system to the future experiments by WMP. We are proud to have taken part in the realization of WMP's solution for clinical magnetic particle delivery.

6. References

[1] B. Shapiro, S. Kulkarni, A. Nacev, S. Muro, P. Y. Stepanov, and I. N. Weinberg, “Open Challenges in Magnetic Drug Targeting,” Wiley Interdiscip Rev Nanomed Nanobiotechnol, vol. 7, no. 3, pp. 446–457, May 2015.

[2] Krizhevsky, A., Sutskever, I. & Hinton, G. E. “ImageNet Classification with Deep Convolutional Neural Networks,” *Communications of the ACM*, vol. 60, no. 6, pp. 3-4, May 2017.

[3] J. Stewart, “Single Variable Calculus: Early Transcendentals 8th edition - James Stewart - Google Books.” pp. 412-414, January 2016.

7. Appendices

A. Team Charter

1. Safety Considerations

The only significant safety hazards for our project involve the motor controller and solenoid coil hardware. The motor controllers and DC power supply will be kept away from any liquids. The solenoid coils will checked frequently during usage for overheating so as to prevent any damage to the system. All testing will allow follow any safety protocol specified by WMP.

2. Provided Equipment

WMP has provided the 4-coil MCS as a testing environment for our software system. The provided system includes: the four-coil array, water housing, two motor controllers, and a power supply. All equipment provided by WMP will be returned at the end of the project timeline.

3. Work Location(s)

- **Weinberg Medical Physics** - 12156 Parklawn Dr, Rockville, MD 20852
- **George Mason University Peterson Hall** - Fairfax, VA 22040

4. Best Effort Basis

To fulfill the project requirements and objectives, we have performed the technical and administrative work required for our project to the best of our ability throughout both the Fall 2018 and Spring 2019 semesters.

5. Handling of Restricted Data

Any data provided by WMP that is designated as confidential will be restricted to viewing and usage by members of our senior design team and, if allowed by WMP, our GMU advisors. This data can include, but is not limited to:

- Code scripts
- Equipment pictures
- Patents (and applications)
- Future company plans or experiments

B. CAD Schematics

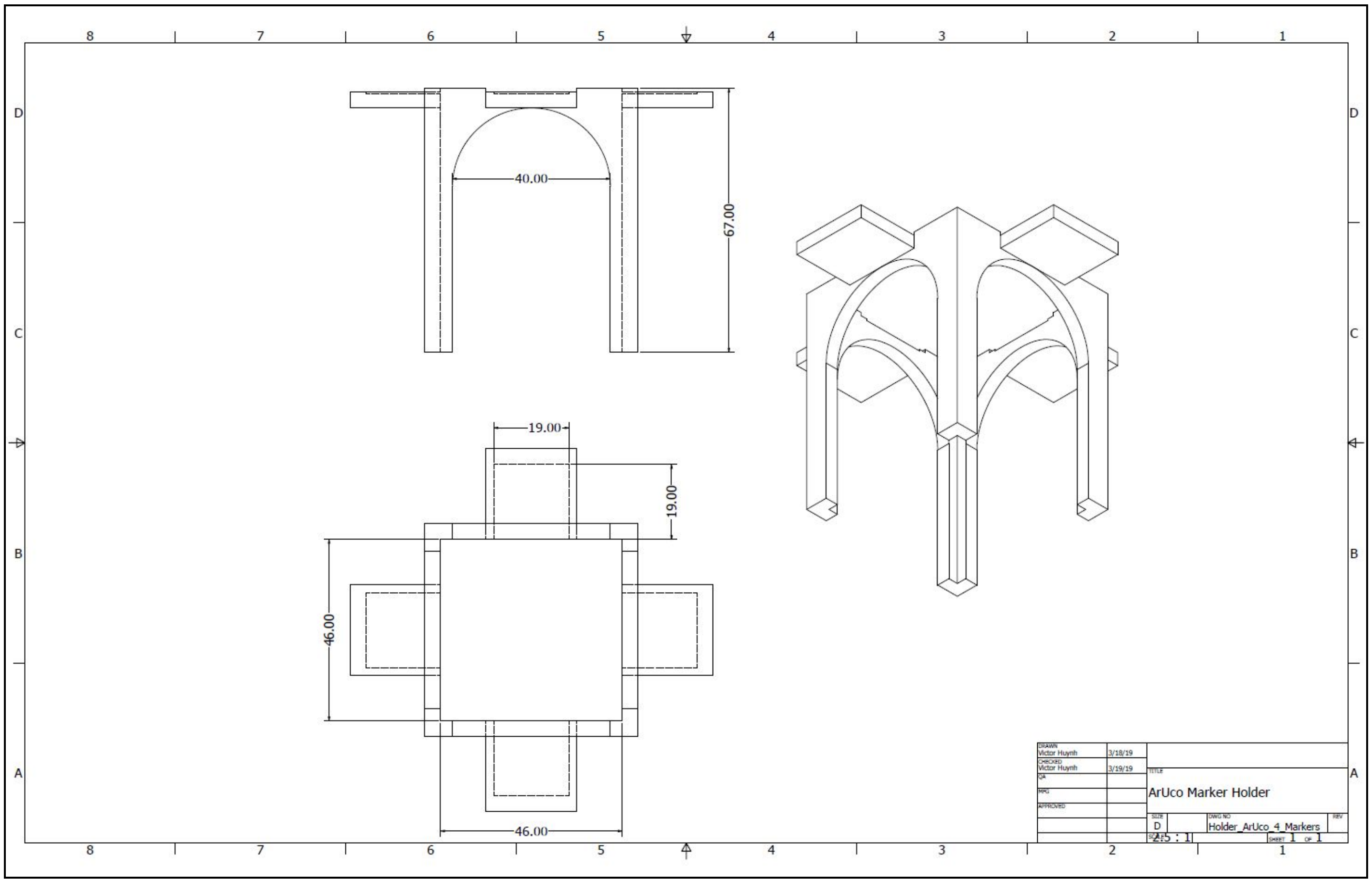


Figure 24. CAD design of 3D-printed ArUco holding platform.

C. Budget

Item	Quantity	Cost
Qt Development Environment	4	$0
Microsoft Visual Studio 2017 Community	4	$0
OpenCV	4	$0
Github Version Control	4	$0
27V 1.593kW Power Supply *	1	$360
ArUco Markers	4	$0.50
Roboclaw 2x60A Motor Controllers *	2	$400
Ipevo Ziggi USB Camera *	1	$148
4-Coil Magnetic Solenoid Array *	1	N/A
Micro USB to USB 2.0	2	$10.78
Total Operational Cost		**$919.28**

Table 11. Amount of money spent on individual components of the project.

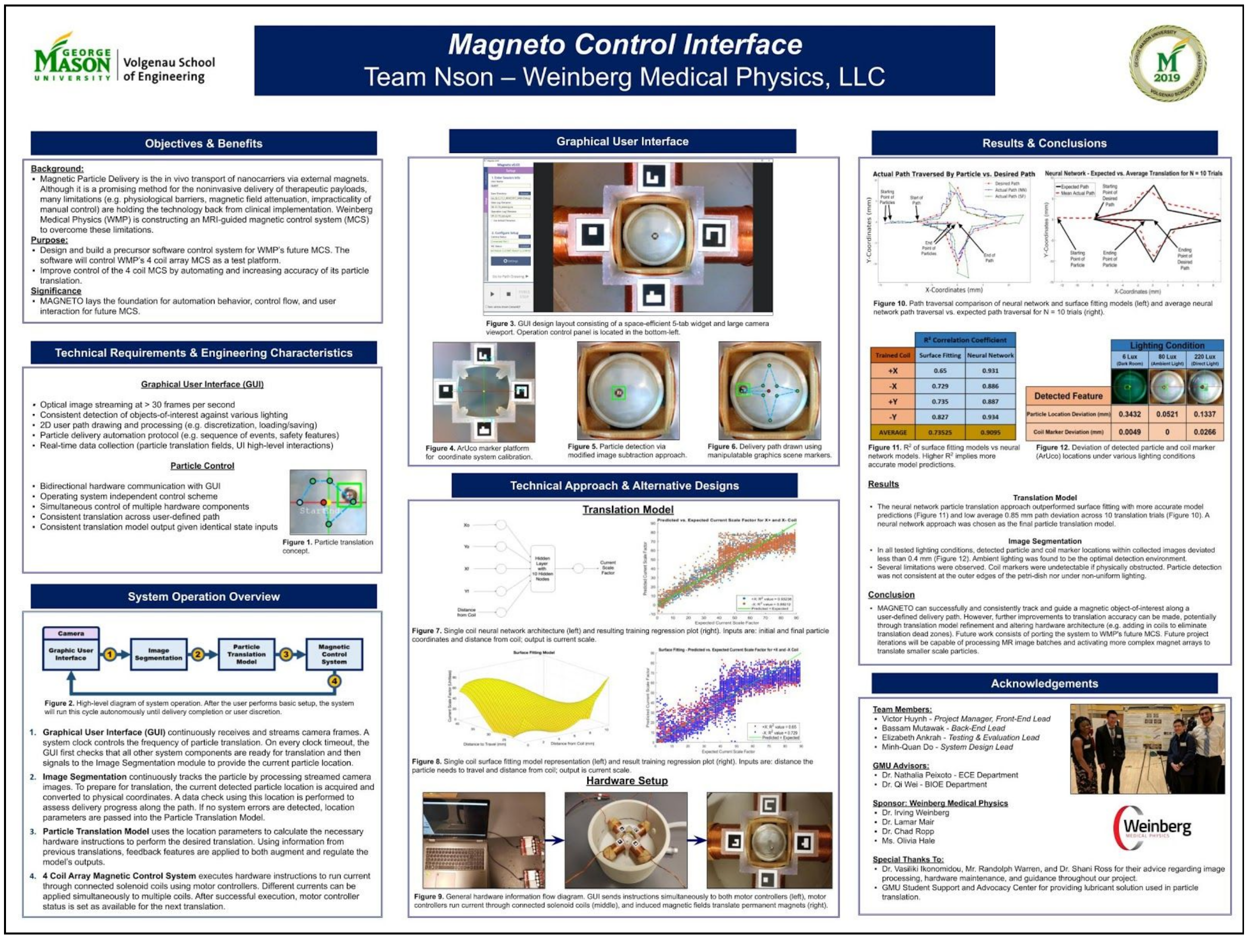
* **Supplied by Weinberg Medical Physics, LLC**

D. Project Schedule (Full-Scope)

Below is an overview of our Spring 2019 project schedule (containing both our accomplishments and in-progress tasks). These include both technical and administrative (for BENG 493) tasks.

Month Due	Task (w/ Status)
February	● BENG 493 Activity Report #1 (complete) ● BENG 493 Activity Report #2 (complete) ● BENG 493 Homework #1 - Prototype T&E (complete) ● Plan GUI design overhaul (complete) ● Plan GUI data collection overhaul (complete) ● Plan Image Segmentation FOV automation (complete) ● Create initial GUI tests checklist (complete) ● Conduct T&E on Fall 2018 GUI prototype (complete) ● Revise and reorganize GUI base code (complete)
March	● BENG 493 Homework #2 - Prototype T&E Results (complete) ● BENG 493 In-Progress Presentation and Report (complete) ● GUI design implementation in code (complete) ● GUI user path drawing via QGraphicsScene (complete) ● GUI image streaming via OpenGL (complete) ● Create base GUI settings window (complete) ● Create separate Data Collection GUI (complete) ● Create and run operation control test cases (complete) ● Implement and test color detection for coil localization (complete) ● Implement and test ArUco markers for coil localization (complete) ● Design and 3D-print platform for placing ArUco markers (complete) ● Implement 2-way direct USB motor controller-to-computer communication (complete) ● Finalize GUI Path Drawing Supplemental Features (complete) ● Plan Image Segmentation automated particle detection (complete) ● Revise neural network architecture for particle translation modelling (complete) ● Revise data collection protocol (collection scheme and code) (complete)
April	● Implement system error detection (complete) ● Refine GUI data log fields and format (complete) ● Implement automatic motor controller connection with GUI software (complete) ● Implement and test finalized particle detection method (complete) ● Test and finalize neural network model for particle translation (simulated vs. actual results) (complete) ● Revise data collection protocol (collection scheme and code) (complete) ● Transition finalized neural network into project's C++ codebase (complete) ● Continually monitor and optimize GUI computational performance (complete) ● Deploying GUI software from Debug to Release versions (complete) ● Capstone Day presentation (complete) ● Major course assignments (complete)

E. Capstone Day Poster



GEORGE MASON UNIVERSITY
Volgenau School of Engineering
GEORGE MASON UNIVERSITY
M 2019
VOLGENAU SCHOOL OF ENGINEERING

Magneto Control Interface

Team Nson – Weinberg Medical Physics, LLC

Objectives & Benefits

Background:

- Magnetic Particle Delivery is the in vivo transport of nanocarriers via external magnets. Although it is a promising method for the noninvasive delivery of therapeutic payloads, many limitations (e.g. physiological barriers, magnetic field attenuation, impracticality of manual control) are holding the technology back from clinical implementation. Weinberg Medical Physics (WMP) is constructing an MRI-guided magnetic control system (MCS) to overcome these limitations.

Purpose:

- Design and build a precursor software control system for WMP's future MCS. The software will control WMP's 4 coil array MCS as a test platform.
- Improve control of the 4 coil MCS by automating and increasing accuracy of its particle translation.

Significance

- MAGNETO lays the foundation for automation behavior, control flow, and user interaction for future MCS.

Technical Requirements & Engineering Characteristics

Graphical User Interface (GUI)

- Optical image streaming at > 30 frames per second
- Consistent detection of objects-of-interest against various lighting
- 2D user path drawing and processing (e.g. discretization, loading/saving)
- Particle delivery automation protocol (e.g. sequence of events, safety features)
- Real-time data collection (particle translation fields, UI high-level interactions)

Particle Control

- Bidirectional hardware communication with GUI
- Operating system independent control scheme
- Simultaneous control of multiple hardware components
- Consistent translation across user-defined path
- Consistent translation model output given identical state inputs

Figure 1. Particle translation concept.

System Operation Overview

Figure 2. High-level diagram of system operation. After the user performs basic setup, the system will run this cycle autonomously until delivery completion or user discretion.

1. **Graphical User Interface (GUI)** continuously receives and streams camera frames. A system clock controls the frequency of particle translation. On every clock timeout, the GUI first checks that all other system components are ready for translation and then signals to the Image Segmentation module to provide the current particle location.
2. **Image Segmentation** continuously tracks the particle by processing streamed camera images. To prepare for translation, the current detected particle location is acquired and converted to physical coordinates. A data check using this location is performed to assess delivery progress along the path. If no system errors are detected, location parameters are passed into the Particle Translation Model.
3. **Particle Translation Model** uses the location parameters to calculate the necessary hardware instructions to perform the desired translation. Using information from previous translations, feedback features are applied to both augment and regulate the model's outputs.
4. **4 Coil Array Magnetic Control System** executes hardware instructions to run current through connected solenoid coils using motor controllers. Different currents can be applied simultaneously to multiple coils. After successful execution, motor controller status is set as available for the next translation.

Graphical User Interface

Figure 3. GUI design layout consisting of a space-efficient 5-tab widget and large camera viewport. Operation control panel is located in the bottom-left.

Figure 4. ArUco marker platform for coordinate system calibration.

Figure 5. Particle detection via modified image subtraction approach.

Figure 6. Delivery path drawn using manipulatable graphics scene markers.

Technical Approach & Alternative Designs

Translation Model

Xo
Yo
Xf
Yf
Distance from Coil
Hidden Layer with 10 Hidden Nodes
Current Scale Factor
Predicted vs. Expected Current Scale Factor for X+ and X- Coil
Predicted Current Scale Factor
Expected Current Scale Factor
+X: R² value = 0.93236
-X: R² value = 0.88212
Predicted = Expected

Figure 7. Single coil neural network architecture (left) and resulting training regression plot (right). Inputs are: initial and final particle coordinates and distance from coil; output is current scale.

Surface Fitting Model
Current Scale Factor (Unitless)
Distance to Travel (mm)
Distance from Coil (mm)
Surface Fitting - Predicted vs. Expected Current Scale Factor for +X and -X Coil
Predicted Current Scale Factor
Expected Current Scale Factor
+X: R² value = 0.65
-X: R² value = 0.729
Predicted = Expected

Figure 8. Single coil surface fitting model representation (left) and result training regression plot (right). Inputs are: distance the particle needs to travel and distance from coil; output is current scale.

Hardware Setup

Figure 9. General hardware information flow diagram. GUI sends instructions simultaneously to both motor controllers (left), motor controllers run current through connected solenoid coils (middle), and induced magnetic fields translate permanent magnets (right).

Results & Conclusions

Actual Path Traversed By Particle vs. Desired Path
Desired Path
Actual Path (NN)
Actual Path (SF)
Starting Point of Particles
Start of Path
End Point of Particles
End of Path
Y-Coordinates (mm)
X-Coordinates (mm)
Neural Network - Expected vs. Average Translation for N = 10 Trials
Expected Path
Mean Actual Path
Starting Point of Desired Path
Starting Point of Particle
Ending Point of Particle
Ending Point of Desired Path
Y-Coordinates (mm)
X-Coordinates (mm)

Figure 10. Path traversal comparison of neural network and surface fitting models (left) and average neural network path traversal vs. expected path traversal for N = 10 trials (right).

	R^2 Correlation Coefficient	
Trained Coil	Surface Fitting	Neural Network
+X	0.65	0.931
-X	0.729	0.886
+Y	0.735	0.887
-Y	0.827	0.934
AVERAGE	0.73525	0.9095

Figure 11. R^2 of surface fitting models vs neural network models. Higher R^2 implies more accurate model predictions.

	Lighting Condition		
	6 Lux (Dark Room)	80 Lux (Ambient Light)	220 Lux (Direct Light)
Detected Feature			
Particle Location Deviation (mm)	0.3432	0.0521	0.1337
Coil Marker Deviation (mm)	0.0049	0	0.0266

Figure 12. Deviation of detected particle and coil marker (ArUco) locations under various lighting conditions

Results

Translation Model

- The neural network particle translation approach outperformed surface fitting with more accurate model predictions (Figure 11) and low average 0.85 mm path deviation across 10 translation trials (Figure 10). A neural network approach was chosen as the final particle translation model.

Image Segmentation

- In all tested lighting conditions, detected particle and coil marker locations within collected images deviated less than 0.4 mm (Figure 12). Ambient lighting was found to be the optimal detection environment.
- Several limitations were observed. Coil markers were undetectable if physically obstructed. Particle detection was not consistent at the outer edges of the petri-dish nor under non-uniform lighting.

Conclusion

- MAGNETO can successfully and consistently track and guide a magnetic object-of-interest along a user-defined delivery path. However, further improvements to translation accuracy can be made, potentially through translation model refinement and altering hardware architecture (e.g. adding in coils to eliminate translation dead zones). Future work consists of porting the system to WMP's future MCS. Future project iterations will be capable of processing MR image batches and activating more complex magnet arrays to translate smaller scale particles.

Acknowledgements

Team Members:

- Victor Huynh - *Project Manager, Front-End Lead*
- Bassam Mutawak - *Back-End Lead*
- Elizabeth Ankrah - *Testing & Evaluation Lead*
- Minh-Quan Do - *System Design Lead*

GMU Advisors:

- Dr. Nathalia Peixoto - ECE Department
- Dr. Qi Wei - BIOE Department

Sponsor: Weinberg Medical Physics

- Dr. Irving Weinberg
- Dr. Lamar Mair
- Dr. Chad Ropp
- Ms. Olivia Hale

Weinberg MEDICAL PHYSICS

Special Thanks To:

- Dr. Vasiliki Ikonomidou, Mr. Randolph Warren, and Dr. Shani Ross for their advice regarding image processing, hardware maintenance, and guidance throughout our project.
- GMU Student Support and Advocacy Center for providing lubricant solution used in particle translation.

Note: A separate PDF attachment is provided along with this report.

F. Man Hours and Report Breakdown

	Victor	Bassam	Minh	Elizabeth	Team
Fall 2018	**367**	**297.75**	**281**	**188**	**1133.75**
Spring 2019	**317.75**	**273.5**	**176.75**	**148.5**	**916.5**
Total	**549.75**	**431.75**	**369.75**	**274.6**	**2050.50**

Table 10. Team member final man-hours across Fall 2018 and Spring 2019 (table format).

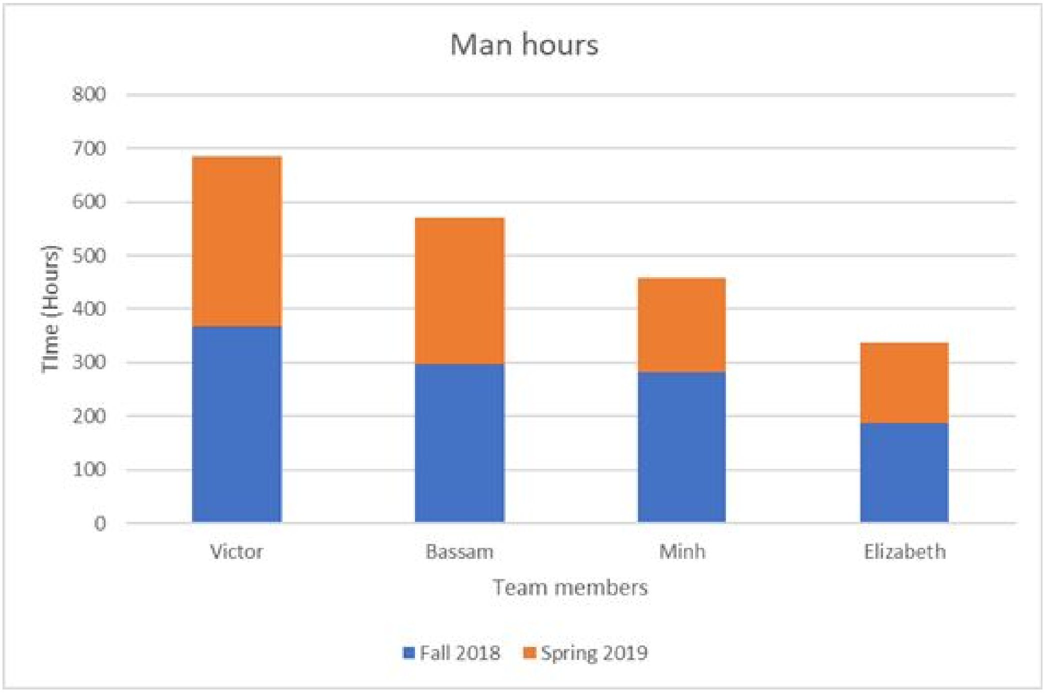


Figure 25. Team member final man-hours across Fall 2018 and Spring 2019 (bar chart format).

G. Project Resources

This appendix serves to a brief guide to others who wish to implement the resources used for conducting this project.

1. Qt

Qt (pronounced "cute") is a free and open-sourced widget toolkit for creating GUI. It is compatible with multiple platforms such as Windows, Linux, and macOS. This was used for the creation of our GUI.

How to download: Qt can be downloaded from the Qt website (https://www.qt.io/). There is a free and commercial version that can be downloaded. For the scope of this project, the free version is sufficient. In order to properly install this software,

Prerequisites: In order to properly use Qt, programmers must be:
- Proficient in C++ (see Learning C++ later in the appendix)
- Basic knowledge on how to use Microsoft Visual Studios
- Link OpenCV to the project opened in QtCreator (Qt's own IDE)
- Link Microsoft Visual Studios compiler to project opened in QtCreator

Links for learning: Below are helpful links for learning Qt

- ProgrammingKnowledge's beginner series teaching widget tools, windows, etc. https://www.youtube.com/playlist?list=PLS1QulWo1RIZiBcTr5urECberTITj7gjA
- Rbaleksandar series on integrating OpenCV and using signals and slots (Part 1 – Part 4) https://www.youtube.com/watch?v=vKIEzqmeajQ
- VoidRealms's video on Qt Debugger https://www.youtube.com/watch?v=B7UsWtyhXh4
- VoidRealms's guide on QThread (for concurrent processing) (Part 1 – Part 6) https://www.youtube.com/watch?v=JaGqGhRW5Ks

2. MATLAB Deep learning toolbox

The Deep Learning Toolbox ™ (formerly Neural Network Toolbox ™) is a toolbox provided (through purchase) with in the MATLAB software. It provides a platform for designing and implementing the Neural Network used for particle translation. The weights achieved from the model is the most important feature.

How to download: MATLAB and its toolbox may be purchased and downloaded directly from the website (https://www.mathworks.com).

Prerequisites:
- Proficiency in MATLAB
- Understanding of NN

How to create Neural Network in Matlab: The Neural Fitting app is a GUI that allows users to build their NN. Programmers can learn how to properly use it through: https://www.mathworks.com/help/deeplearning/gs/fit-data-with-a-neural-network.html. In addition the the GUI programmers may code it. The MATLAB mathworks platform should be used to answer any specific questions or address any problems.

Acquiring NN Weights:
https://www.mathworks.com/help/deeplearning/ref/getwb.html?s_tid=doc_ta

3. ArUco

ArUco is an open sourced library that is used through OpenCV. It is used for detecting squared fiducial markers ("ArUco markers") in images.

Prerequisites:
- Proficiency in C++
- Download and set up OpenCV and OpenCV - contrib
- Download Microsoft Visual Studio

Links for learning:
- Opencv documentation on how to use Aruco markers
  https://docs.opencv.org/3.4.0/d5/dae/tutorial_aruco_detection.html
- Original documentation by creators is on an open access google doc
  https://docs.google.com/document/d/1QU9KoBtjSM2kF6ITOjQ76xqL7H0TEtXriJX5kwi9Kgc/edit
- Rafael Muñoz Salinas (the creators) video tutorial on ArUco
  https://www.youtube.com/playlist?list=PL7EOs-8ZXfMb2qRog9wOa3Ar-EyvRYdrp

4. OpenCV 3.4.5

OpenCV (Open Source Computer Vision Library) is an open sources library for computer vision and machine learning softwares. Multiple libraries are used in order to create the image segmentation features needed to track, and identify objects for our system.

How to download: OpenCV can be downloaded from the GitHub repository (https://github.com/opencv/opencv). Please follow the documentation on the github to download and install the base OpenCV library. To download the required contrib libraries, please follow this youtube link (https://www.youtube.com/watch?v=MMDABTypnZg).

Documentation: Can be found on the Opencv doc website (https://docs.opencv.org/) be sure to specify the version you wish to use. This contains information about all libraries and their functions.

Links for integrating OpenCV with various platforms:
- Stackoverflow response for setting up OpenCV with QTcreator. Look at second answer
  https://stackoverflow.com/questions/15881913/how-to-link-opencv-in-qtcreator-and-use-qt-library
- Video on how to integrate OpenCV with Microsoft Visual Studios:
  https://www.youtube.com/watch?v=oJ6fh-XLjtg

5. Eigen 3.3.7

Eigen library is a C++ matrix library used to deploy our trained neural network into the C++ codebase.

How to download: The full eigen library can be downloaded and installed using the following link (https://eigen.tuxfamily.org/dox/GettingStarted.html).

Documentation: Comprehensive documentation on all available functions and packages can be found here (http://eigen.tuxfamily.org/dox/).

6. C++ Programming Language

C++ is a object-oriented programming language with low-level language capabilities. The majority of the project was coded in C++. This is a prerequisite for all other softwares except Matlab.

How to download: Can be accessed through Microsoft Visual Studios (MVS). See the MVS section below.

Prerequisite: Microsoft Visual Studios

Learning: There are multiple online course and resources available to learn C++ online. Lynda.com and udemy are some website used to learn C++.

7. Microsoft Visual Studios (MVS)

MVS is an integrated development environment. This was used to prototype and test back-end code before deployment into the QtCreator platform to create our GUI.

How to download: The community version was downloaded from the website (https://visualstudio.microsoft.com/downloads/). Be sure to download C++ development atmosphere.

Links for learning:
- Steps provided by Microsoft on how to install MVS (https://docs.microsoft.com/en-us/cpp/build/vscpp-step-0-installation?view=vs-2019)

8. Roboclaw

Roboclaw 2x60A motor controllers are directly connected to the GUI and send current through our coils. These hardware components are controlled through a C++ library of functions.

How to download: Several libraries to control the roboclaw can be found on the product web page (https://www.basicmicro.com/RoboClaw-2x60A-Motor-Controller_p_8.html). It's important to note that a C++ implementation of this library (which our project uses) is not available on the website. In general, our C++ implementation is based on the Arduino roboclaw library (see link product page link). The actual implementation can be found in the software package attached to this report.

Prerequisite: ROS Serial Library (http://wiki.ros.org/serial).

H. GUI Functionality Checklist

See next page.

GUI Functionality Checklist - evaluated by Team NSON

This spreadsheet is used for objective evaluation of GUI functionality
(e.g. Do the UI components and widgets fulfill their intended purpose?).

Section	Widget/Component	Purpose	Functional (Y = 1/N = 0)
Main Window - General	Camera Viewport	Displays images streamed from the connected camera. No lag nor screen tear artifacts should be present	1
	Exit Dialog	When the user presses the exit button on the window, this prompts him/her to confirm if he/she actually wants to exit the software.	1
Main Window - Setup Tab	User Name Line Edit	Allows the user to enter his/her name which will update the data log filenames appropriately.	1
	Browse Save Directory Button	Allows the user to select the directory in which to export collected data.	1
	Data Log Filename Line Edits	Allows the user to modify the name of the output data files.	1
	Use Default Filenames Checkbox	Resets the data log filenames to a conventional default format	1
	Connect Camera Button	Connects the camera to the software and begins streaming to the viewport	1
	Connect Motor Controllers Button	Connects the roboclaw motor controller hardware to the software	1
	Settings Window Button	Opens the settings window. Enabled only after both the camera and motor controllers have been connected.	1
	Go to Path Drawing Button	Enables and opens the path drawing tab. Enabled only after all necessary settings operations have bene performed.	1
Settings Window	Calibrate System Button	Executes the calibration of system coordinates and field-of-view; output should show the bounding box, detected axes, and particle movement during the calibration period	1
	Filter Threshold Slider	Modifies the pixel threshold data field for image subtraction; updates the value in the corresponding spinbox	1
	Filter Threshold Spinbox	Modifies the pixel threshold data field for image subtraction; updates the position of the corresponding slider	1
	Minimum Particle Size Slider	Modifies the minimum particle size data field for image subtraction; updates the value in the corresponding spinbox	1
	Minimum Particle Size Spinbox	Modifies the minimum particle size data field for image subtraction; updates the position of the corresponding slider	1
	Maximum Particle Size Slider	Modifies the maximum particle size data field for image subtraction; updates the value in the corresponding spinbox	1
	Maximum Particle Size Spinbox	Modifies the maximum particle size data field for image subtraction; updates the position of the corresponding slider	1
	Display System Axes Checkbox	Toggles whether to display the calibrated system axes on streamed images; purely cosmetic option	1
	Command Frequency Spinbox	Modifies the frequency at which to output translation commands to the motor controllers; can be edited during paused operation	1

	Preview Button	Preview image subtraction results using the set Filter Threshold, Mininum Particle Size, and Maximum Particle Size parameter values; detection changes should be instant noticeable on the viewport	1
	Apply Settings Button	Confirms all setting values within the Settings Window; updates data fields across the rest of the GUI as appropriate	1
	Cancel Settings Changes Button	Cancels any changes to values within the Settings Window and exits the window	1
Main Window - Path Drawing Tab	Show Drawing Instructions Drop-Down	Toggles display of bulleted path drawing instructions; pushes other tab components down appropriately	1
	Show Path Info Drop-Down	Toggles display of path-related fields; pushes other tab components down appropriately	1
	Load Path Button	Opens file explorer dialog and prompts user to select a ".path" file; if a current path is present, confirms with user if he/she wishes to override that path with the loaded path	1
	Save Path Button	Prompts user if he/she wishes to save the current path and opens a file explorer dialog to select the save directory	1
	Draw/Pause Draw Path Button	Enables/unables the user to click, move, and remove path point markers over the camera viewport	1
	Clear Path Button	Clears the current path after confirming with user if he/she no longer needs the current path	1
	Interpolation Amount Spinbox	Modifies the interpolation amount data field (level of path discretization for actual delivery)	1
	Path Tolerance Spinbox	Modifies the acceptable delivery deviation data field	1
	Confirm Path Button	Processes path point locations and path-related values through the GUI; if no errors detected, enables usage of the Operation Control Panel	1
Main Window - Operation Control Panel	Start/Pause/Resume Operation Button	Starts/pauses/resumes the delivery operation; while delivery is ongoing, users should have limited freedom navigatign other UI components; in paused state, user	1
	Stop Operation Button	Stops the delivery operation regardles of current detected particle position	1
	Force Stop Operation Button	Stops the delivery operation regardles of current detected particle position (for this project, same as Stop Operation functionality)	1
	Save Camera Stream Checkbox	Writes streamed camera images into an output video file starting from when the checkbox is checked to when it is unchecked (or when delivery is finished)	1
Main Window - Instructions Tab	Instructions List	Displays instructions for conducting a delivery operation	1

Report Breakdown (Team Contribution)

Victor Huynh

- Executive Summary
- Contributors
- Problem Definition
- Requirements (GUI)
- Metrics (GUI)
- Final Design (GUI)
- Testing Methodology (GUI)
- Societal, Ethical, Economic and Global Context Benefits
- Summary
- References
- Team Charter
- Project Budget
- CAD Schematics
- Project Schedule
- Project Man-Hours
- Project Resources

Bassam Mutawak

- Requirements (Hardware Control)
- Metrics (Hardware Control)
- Final Design (Hardware Control)
- Final Design (Image Segmentation)
- Testing Methodology (Hardware Control)
- Testing Methodology (Image Segmentation)
- Societal, Ethical, Economic and Global Context Benefits
- Summary
- References
- Team Charter
- Project Budget
- CAD Schematics
- Project Schedule
- Project Resources

Minh-Quan Do

- Final Design (Particle Translation Model)
- Testing Methodology (Particle Translation Model)
- References
- Project Resources

Elizabeth Ankrah

- Contributors
- Metrics (GUI)

- Final Design (Image Segmentation)
- Testing Methodology (GUI)
- Societal, Ethical, Economic and Global Context Benefits
- Project Man-Hours
- Project Resources