Carnegie Mellon University

Robotics Capstone Project

System Development Review

Friction Force Explorers:

Don Zheng
Neil Jassal
Yichu Jin
Rachel Holladay

supervised by
Dr. Cameron Riviere

Version 1.0
March 7, 2017

Contents

List of Figures

1 Build Progress

This section details system development and progress made since the last milestone presentation. Progress is split into two major sections: electromechanical and software. Electromechanical updates detail chassis build progress, as well as setup of the electronics to drive the motors for locomotion and using the writing implement. Software updates describe progress towards subsystem completion.

1.1 Electromechanical Updates

As shown below in Fig.1.1, we have built a physical robot prototype that incorporates chassis, painting mechanism, and locomotion system. Chassis is made of laser patterned acrylic. It is designed to be as compact as possible because smaller robots are less likely to collide with each other during drawing operations. This prototype proves that the chassis' current cutout sizes have no clearance issue with moving components.

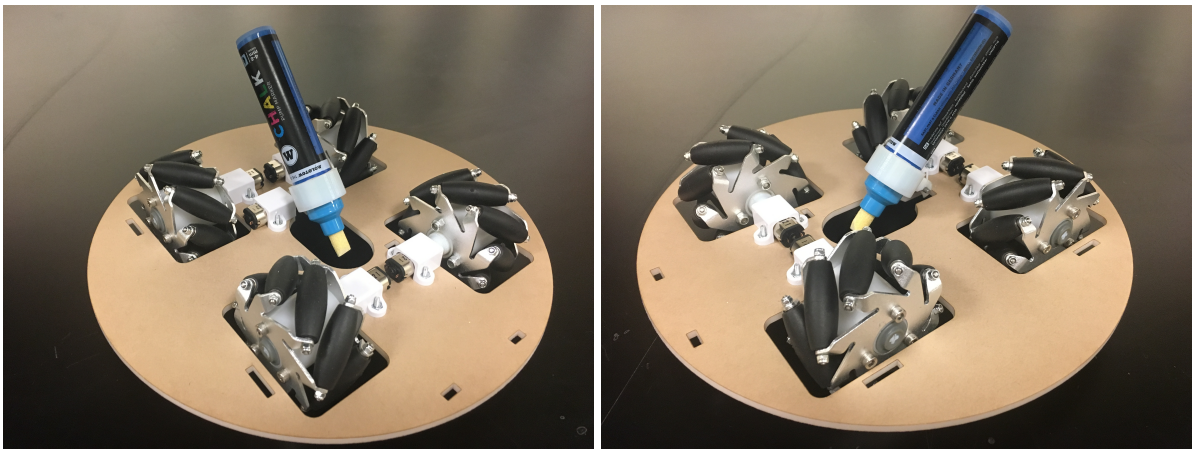


Figure 1: Prototype Overview, from left view (left) and right view (right)

Painting mechanism composes a 3D printed chalk holder and a micro gear motor which is shown in Fig.1.1. The driving motor is mounted to the chassis via an off-the-shelf motor case. When designing the chalk holder, four internal ribs are added inside the holder to securely hold the chalk marker in place while allowing users to easily switch out the marker. A thin cap is added on the bottom of the chalk holder to prevent the chalk marker from sliding out while drawing. One flaw of this design is that the holder's D-shaft cutout is a little undersized. Therefore, the chalk holder broke while pressing the motor shaft through. This problem will be addressed in the next iteration.

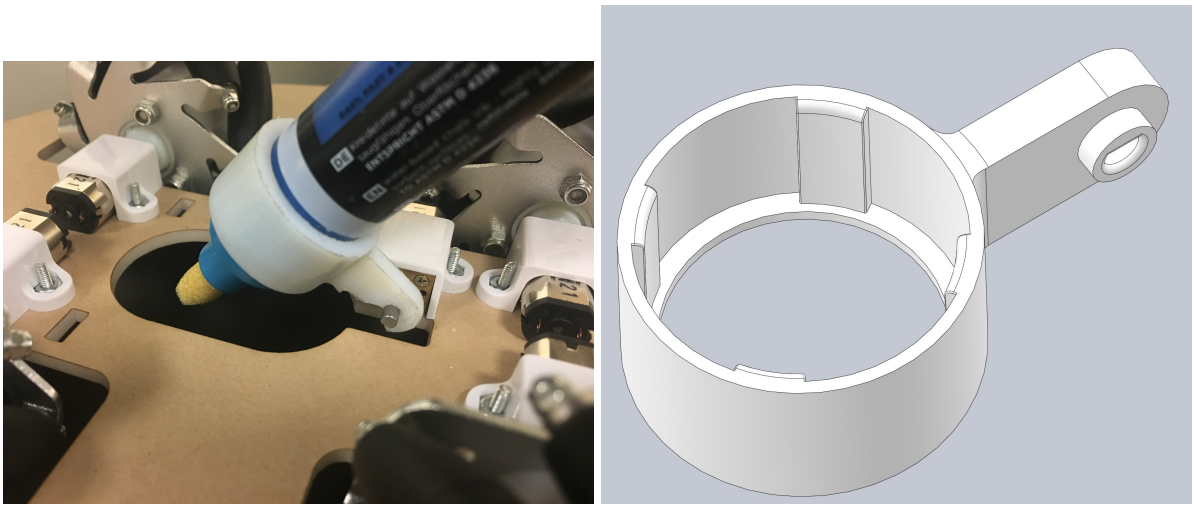


Figure 2: Painting Mechanism (left), Chalk Holder CAD (right)

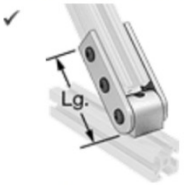
Fig.1.1 shows the locomotion system. Four Mecanum wheels are oriented in a "X" shape to minimize motor workload. These wheels are connected to driving motors through 3D printed wheel adaptors. These adaptors contain two segments: standard Lego technic axle and D-shaft housing. Like the chalk holder, the D-shaft cutout is a little undersized. Therefore, we had to press fit the motors in.

Figure 3: Locomotion System (left), Locomotion System Components (center), Wheel Adaptor CAD (right)

Besides mechanical update, motor controller code was also completed. However, we did not get enough time to wire all electronics to this prototype and test the code. This would be the next step of system development.

Since we have enough left-over budget, we plan to use 80/20 aluminum frames, instead of wood, to construct the camera jig. The jig will be built using components listed in Fig.1.1. We are in the process of testing camera's optimal height, and will then incorporate that information to the camera jig CAD design.

Pivots for Single Rails



Inline and Inline/Perpendicular

Pivots provide smooth, consistent motion at the junction between two rails.

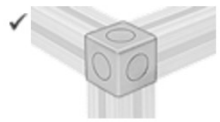
For Rail Ht.	Lg.	Color	Material	Mounting Fasteners Included		Each
Inline						
1"	3"	Silver	Anodized Aluminum	Yes	47065T191	$16.33
1 1/2"	4 1/2"	Silver	Anodized Aluminum	Yes	47065T13	18.43
Inline/Perpendicular						
20mm	2 3/8"	Silver	Anodized Aluminum	Yes	5537T219	17.73
30mm	3 5/8"	Silver	Anodized Aluminum	Yes	5537T865	20.17
40mm	4 3/4"	Silver	Anodized Aluminum	Yes	5537T221	21.07
45mm	5 1/2"	Silver	Anodized Aluminum	Yes	5537T222	21.70

Single Standard Rails—Aluminum

Hollow rails are lighter and more economical than solid rails.

Rail Ht.	Rail Wd.	Rail Construction	T-Slot Wd.	Center Hole Dia.		Lengths 1 ft.	2 ft.	3 ft.	4 ft.	5 ft.	6 ft.	8 ft.	10 ft.
Silver Anodized													
1 1/2"	1 1/2"	Hollow	0.32"	0.26"	47065T102	$7.76	$15.04	$21.78	$26.90	$33.33	$39.31	$51.36	$61.94
30mm	30mm	Hollow	8mm	7mm	5537T97	5.84	9.47	13.09	16.73	20.36	23.98	31.24	38.49
45mm	45mm	Hollow	10mm	10mm	5537T103	8.44	14.03	19.98	25.80	30.50	35.96	48.22	59.02

Corner Brackets for Single Rails



Three-Way Outside Corner

Outside corner and **three-way outside corner brackets** require tapped holes to connect rails.

For Rail Ht.	Color	Material	Lg.	Mounting Fasteners Included		Each
Three-Way Outside Corner						
1"	Silver	Anodized Aluminum	1"	Yes	47065T244	$9.86
1 1/2"	Silver	Anodized Aluminum	1 1/2"	Yes	47065T245	10.78

Figure 4: 80/20 Parts

1.2 Software Update

1.2.1 Communication

The goal of the communication subsystem is to abstract out networking operations such that other subsystems can maintain modularity. Currently, this subsystem is mostly complete. The system is able to easily establish, maintain, and close TCP connections between the separate robots, given their IP addresses. It also contains code to generate a singular protobuf data containing all data necessary to send robots, and pass it through the TCP connection. Onboard communication code is able to continually receive these TCP protobuf messages, and parse them accordingly.

In order to further isolate communication code from the actual subsystems, other subsystems fill in a data struct containing relevant information to send to the robots. For example, the localization subsytem will enter information into a localization data struct, which is passed to the communication subsystem at runtime. The communication subsystem will then parse relevant localization data into the protbuf message to send across the network. These data structs have the additional use of allowing for convenient transfer of data between other subsystems as well.

1.2.2 Locomotion

The locomotion subsystem has had some major changes. Previously, we planned to run locomotion offboard, where it would generate motor powers to send to the robot system. However, analysis showed that offboard motor processing incurs a higher latency than an accurate control could easily use. Decreased latency allows the motor PID controller to provide more accurate stabilization, to better enable the robot to follow a path. As a result, the locomotion subsystem is being moved to an onboard robot system. The offboard system will send the robot's current position and orientation, as well as a target position and orientation. The robot will compute the locomotion commands necessary to reach the target, and run the position and velocity controller accordingly. The positional localization and target data is able to be sent via protobuf message to the onboard system. The encoder and localization motor control is written, and will be tested with sample data once chassis construction is completed.

1.2.3 Localization

The localization subsystem is mostly unchanged, and currently in progress. Integration of the AprilTags C++ library is in progress, which requires setting up the C++ environment, and passing functions to the Python subsystem via Boost.

1.2.4 Scheduling, Distribution and Planning (SDP)

In order to our SDP module, we first had to add a few basic UI elements. We laid out a file format for specifying the lines to be drawn and wrote the UI functionality to parse the data in.

Given the data the next step is to distribute the work between the two robots, offline. We will later describe a first pass distribution algorithm along with the UI developed to visualize its results. We will conduct further testing to see if a more advanced algorithm is needed. Luckily, this can be done in parallel with other developments since we have fixed the input and output of the system, allowing us to swap in different distribution tactics. The output of the distributor is a set of vectors that specify the plan for each robot. These vectors will then be handed off to the locomotion module, described above, that will follow each of them in sequence. Therefore, this gives us two next steps: to integrate the planning with the locomotion and to develop a collision avoidance strategy.

To handle collisions, we will start off with a naive strategy. We define a robot's *boundary* as a fixed radius circle around robot, where the radius exceeds that of the robot to provide cushion. As each robot moves, it will check if the other robot's boundary intersects with its own boundary. If this condition is true, one robot (Bad) will stop execution, allowing the other robot (Blue) to pass until the condition is false. While we believe this method will always prevent collisions, it may not be the most efficient. Therefore we will implement this and test accordingly to check performance.

For our distributor, we developed a very greedy method. We start Blue the robot at one corner of the drawing area and Bad the robot at the other corner. Our goal is greedily balance their cost, where cost corresponds to the length of the line drawn so far. We initialize both robots with cost zero. From there, we loop over the line count. We pick the robot with the lower cost, defaulting to one in the case of equality. Whichever robot has the lower cost, we pick the line with the closest starting point to the

robot's current position. We then calculate the cost as the distance to drive to the line plus the distance to drive to draw out that line. Having updated the robot's position, we continue.

To illustrate the output of our planner we developed a visualization, shown below in Fig.1.2.4. The red path represents Bad the robot and the blue path represents Blue the robot. Solid lines corresponding to drawing lines and hence making a mark on the pavement while dotted lines correspond to purely transit. In Fig.1.2.4, Blue the robot starts from the top right corner and transits to draw one line and then return home. In contrast, Bad the robot draws two intersecting, nearby lines.

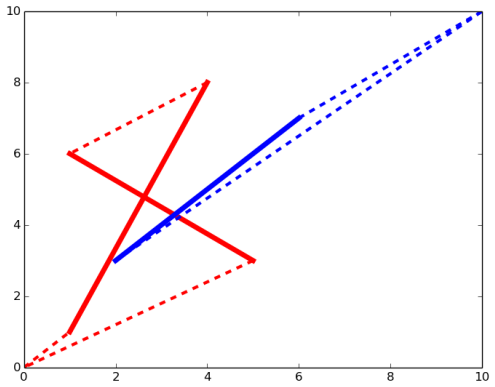


Figure 5: Planning Output

In some cases, such as in Fig.1.2.4, we see a nice breakdown of work as each robot is responsible for one box. The occurs because at the end of each step, the next closest line is directly adjacent, making the transit distance zero. In Fig.1.2.4 we see that the greedy approach segments the lines less cleanly and raises the possibility of collision. We note that at every intersection of two robots there is a possibility of collision, but not necessarily if the robots traverse those areas at different times. Additionally the robots could collide outside of intersections since we are not dealing with point robots. As mentioned above, we will continue to explore collision avoidance.

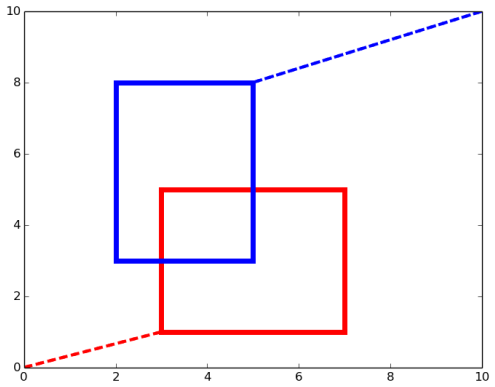


Figure 6: Planning Output

2 Schedule

Fig.2 shows an updated Gantt chart of our system development progress. Sections colored in gray represent completion status of various tasks. Overall, we are on schedule, with few changes. The UI subsystem has been moved forward to the end of spring break, as our team has time to work on it then. In addition, the UI system has been developed in conjunction with the planning system, and as such is already partially completed.

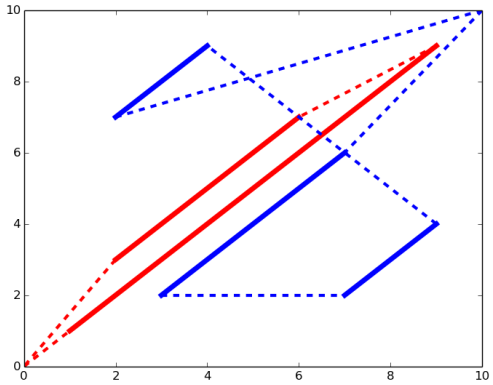


Figure 7: Planning Output

Electromechanical progress remains on schedule, with slight delays and adjustments due to parts taking longer than anticipated to deliver. The chassis is completed, and is currently in testing for potential issues such as vibration causing unstable writing motions. The writing implement is mostly completed, but due to a broken part is currently being refabricated. The camera rig is slightly behind, and we have decided to alter the material used to design it. Instead of wood, we switched to using 80/20, which are designed for simple construction and adjustment.

Software progress remains on schedule. The scheduling subsystem is currently operational, with some integration left. The communication subsystem has been completed, localization is being finished, and locomotion is in complete, but being updated to account for design changes.

	Week Number	1	2	3	4	5	6	7	8	9	10	11	12	13
WBS	Task	1/30	2/6	2/13	2/20	**2/27**	3/6	3/20	3/27	4/3	4/10	4/16	4/24	5/1
1.1	Electromechanical													
1.1.1	Finalize CAD Design													
1.1.2	Obtain Parts													
1.1.3.1	Chassis Fabrication													
1.1.3.2	Writing Implement Fabrication													
1.1.4.1	Camera Rig Assembly													
1.1.4.2	Robot Agent Assembly													
1.1.5	Mechanical Testing													
1.2	Software Implementation													
1.2.1	Softare Architecture Design													
1.2.2.1	Localization Subsystem Development													
1.2.2.2	Locomotion Subsystem Development													
1.2.2.3	UI Subsystem Development													
1.2.2.4	Communication Subsystem Development													
1.2.2.5	SDP Subsystem Development													
1.2.3	Software Subsystem Testing													
1.3	Integration													
1.3.1	Software Subsystem Integration													
1.3.2	Full System Integration													
1.3.3	Requirements Testing													
	Demo Preparation													

Figure 8: Prototype Overview, from left view (left) and right view (right)

3 Risk Management