RoboCup Rescue 2017 Team Description Paper MIT Robotics Team

Drew Beller, David Mayo, Joey Muller, Michelle Tan, Reo Baird, and Lukas Beyer

Info

Team Name:	MIT Robotics Team
Team Institution:	Massachusetts Institute of Technology
Team President:	Drew Beller
Team Technical Lead:	David Mayo
Team Faculty Advisor:	Russ Tedrake
Team URL:	https://roboteam.mit.edu

RoboCup Rescue 2017 TDP collection: http://wiki.robocup.org/Robot_League

***Abstract*—This paper encloses details about the MIT Robotics teams planned submission into the RoboCup Rescue 2017 Challenge. The team focused on building a robust system that autonomously and seamlessly aids the operator in control of the robot and also sends back important map and vision data needed in a first responder situation. The team took advantage of high-end sensors and robotic equipment to see the limits of what is possible in robotic disaster relief today.**

***Index Terms*—RoboCup Rescue, Team Description Paper, MIT Robotics Team, Disaster Relief, Robot.**

I. Introduction

THE goal of the MIT Robotics Team is to create a robust search and rescue platform that utilizes semi-autonomous functionality to aid the operator. We seek to take advantage of high end sensors and robotic equipment to push the boundary of what is capable by traditional teleop control. We believe this hybrid human-computer semi-autonomous system will improve performance of search and rescue robots in both their speed in completing tasks and the variety of tasks the robot is able to complete.

II. System Description

A. Hardware

Refer to Table IV in the Appendix.

1) Drive System and Chassis: We are using the HD2 Chassis from Super Droid Robotics. The HD2 Chassis is a robust belt driven chassis capable of supporting the weight of our arm and sensors over a variety of difficult terrains. To account for even more difficult terrains, the team will be adding flippers on the front of the chassis to aid in the traversal of obstacles taller than the chassis.

The MIT Robotics Team works out of the MIT Edgerton Center, e-mail: roboteam-exec@mit.edu.

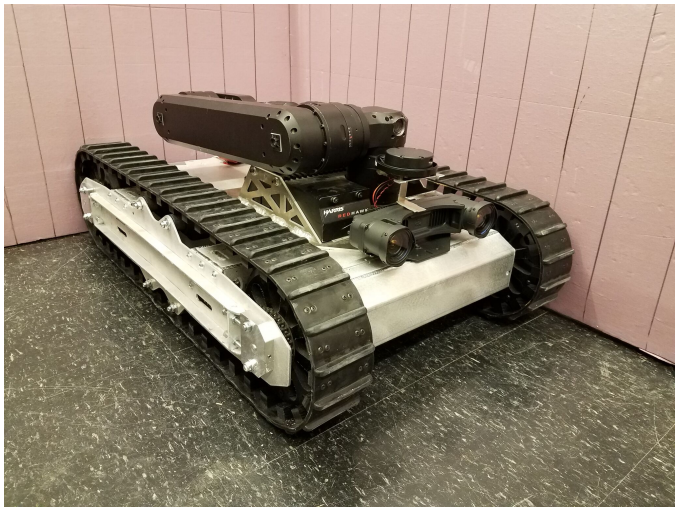

Fig. 1. Robot in expected final assembly configuration. The arm is folded back for storage and navigation.

2) Arm and Manipulator: We are using the Harris RedHawk MPR robotic arm see Table II for more details. We are integrating the arm control software with the MoveIt motion planning and inverse kinematics software package. The six degrees of freedom the manipulator provides and the custom hardware we are developing for the end effector will be helpful in completing manipulation tasks, as it contains multiple cameras and sensors.

For the end effector,we have developed our own custom gripper for the arm and we have already developed a custom PCB to interface with the arms internal wiring. The RedHawk MPR provides Ethernet on its tool interface connector. Our custom end effector electronics contain a single-board computer running Linux, which will make it possible to interface with sensors and actuators on the end effector. Additionally, the custom hardware contains multiple switch-mode voltage regulators and provides power for the embedded computer (5V and 3.3V), servos and other actuators (12 to 15V configurable) and high power LEDs (using a constant current driver). To communicate with sensors, we have included an ARM Cortex M4 microcontroller and the necessary circuitry for RS232, RS422 and RS485 serial interfaces. Mechanically, our end effector electronics are mounted in an almost fully enclosed cylindrical shell that attaches to the arms tool end. The only exposed components are a 40 pin header connector, a CO2 gas sensor, an analog video input connector, and a thermal camera interface.

The most useful connections are broken out to the conve-

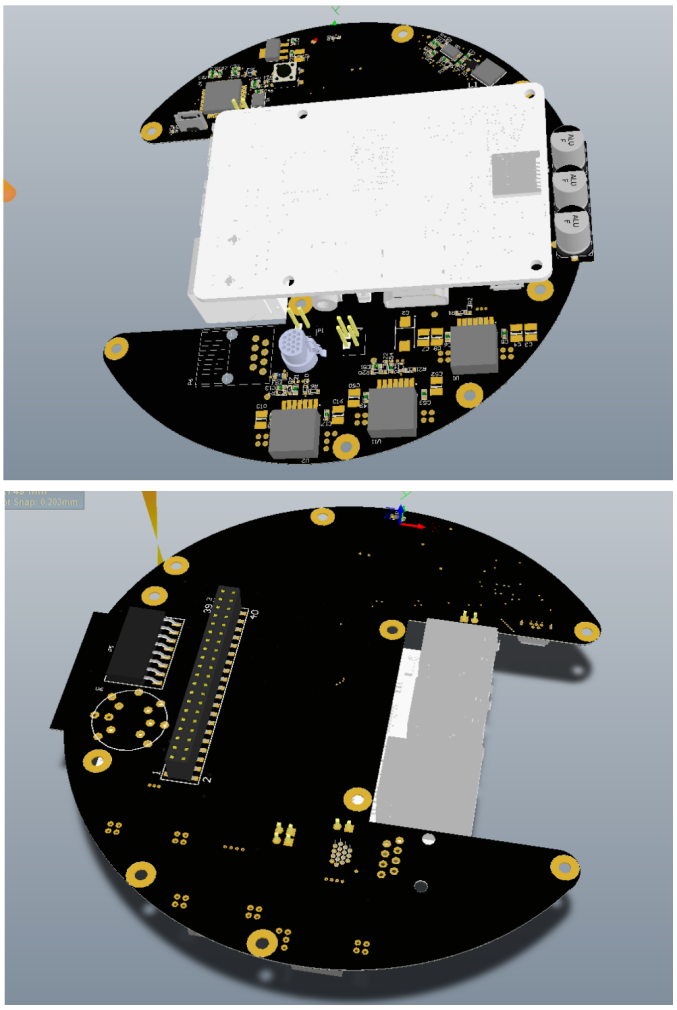

Fig. 2. CAD model of the end effector electronics (top: shows underside of electronics that will be facing the arms tool end; bottom: shows the top side of electronics, which will be facing the gripper. 40 pin header, CO2 and thermal connectors visible)

nient 40 pin header on the tool end of our custom electronics. The variety of interfaces and power rails provided on this connector will enable the development of multiple end effector designs, and provides the expandability required for adding and experimenting with different sensor configurations.

See Figure 2.

3) Computer: The robot uses a full desktop computer running Ubuntu 16.04 with ROS Kinetic. The computer is completely configurable and is set up with an external solid state hard drive, a quad core i7 Intel processor and 16 gb of RAM. Currently the computer has no on-board graphics card though we plan to add one to improve performance in handling sensor data.

4) Sensors: Standard webcams will be used to send real-time video feedback to the operator. A Neato XV-11 lidar and a Carnegie Robotics MultiSense S21 stereo camera will be used for mapping, localization, and path planning. A MG811 CO2 sensor will also be mounted on the custom PCB for our end-effector. The end effector will also include a FLIR Lepton thermal imaging camera.

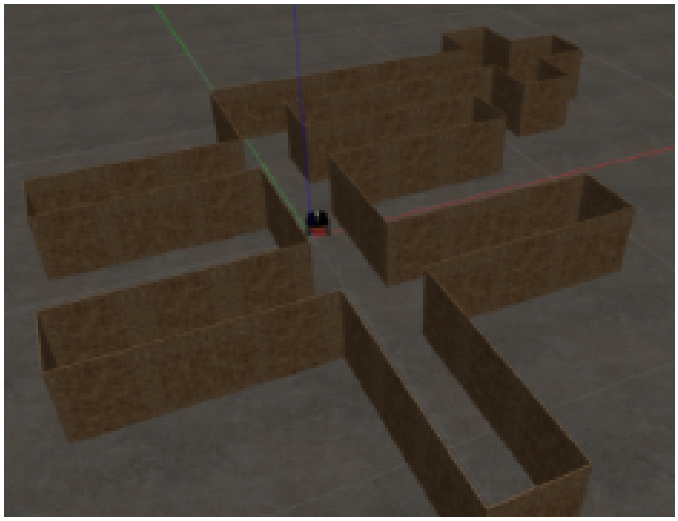

Fig. 3. Pioneer2dx robot in one of the NIST standard test arena scenarios

B. Software

Refer to Table V in the Appendix.

1) Low-Level Control: Low-level control of the robot systems will be managed via multi-threaded ROS topics. Communications to and from the drive train will be handled over serial RS-232. Communications to and from the arm and stereo camera will be handled via Ethernet. The lidar, IR camera, and webcams will be connected directly to the motherboard via USB.

Communication to the robot from the operator's station will be handled by a combination of UDP (User Datagram Protocol) and TCP (Transmission Control Protocol) calls over a single channel wi-fi connection. UDP will be used for teleop control and direct video feedback, while TCP will be used for less time sensitive data like map updates.

2) SLAM: Some of the most essential tasks for rescue robots are to be able to both dynamically map terrain for rescue responders and also to keep track of its location in order to explore an area effectively. We are using Hector SLAM with a 2D Neato laser scanner in order to simultaneously localize our position and create a map. This creates a probabilistic occupancy grid that shows what areas are empty and occupied. In order to test out the mapping without the robot, we used gazebo with the Pioneer2dx robot as shown in Figure 3.

3) Vision: The vision sensors on the robot include standard RGB cameras on the arm base and manipulator, a stereo camera on the front of the robot, and a FLIR thermal camera on the manipulator. Currently, we are able to detect QR codes using image data from our robots RGB cameras and process it using OpenCV.

We are also able to detect objects without artificial markers using deep convolutional neural networks. We have created a dataset of images of objects that a search and rescue robot may need to be able to identify autonomously such as door handles, fire extinguishers, valves, etc. We then used a technique called transfer learning, which involved removing the final layer from a convolutional neural network that had already been trained on the ImageNet dataset (specifically the Google Inception v3),

rewiring it to classify images into the categories we choose for our search and rescue image dataset, and then training the network on our dataset. This method allows us to leverage the power of the feature representations learned from training a deep neural network on a large image dataset, while applying them to classifying our new smaller dataset after only a small amount of retraining. In practice, this technique yields a high degree of accuracy.

In the future, we plan to expand our dataset to be able to detect a larger number of obstacles. We also plan to implement algorithms to utilize 3D point clouds created by our stereo camera to detect obstacles and identify open spaces. For the RoboCup Rescue manipulation tasks we plan to use deep neural networks to detect objects that we need to grasp with our arm. To detect the specific orientation and position of markers on objects, we will be using more feature-specific algorithms.

4) Path Planning: The global map and point cloud data from the stereo camera will be used to plan the best path around obstacles. Obstacle avoidance will occur autonomously as the operator simply drives in the desired direction. When the flippers are added onto the drive train, the path planning will utilize them to navigate over obstacles, by setting their optimal angle for traversing over what is ahead of the robot. Development of this feature has already begun in simulation.

C. Communication

Communication between the robot and operator station occurs via a single channel wireless network. The rover will broadcast its own secure network that the operator can then connect to. Feedback and control data is sent as ROS messages between the operator station the robot. The video feed is also transferred as uncompressed image frames continuously updated at five Hz. We are currently exploring further techniques for data compression, reducing latency, and generally improving connectivity.

D. Human-Robot Interface

The robot may be driven by one driver via the control interface. The interface is implemented as a rqt node, with multiple plug-ins and provides a video feed from the on-board cameras, a click-and-drag map, a ROS message monitor, and a terminal. The GUI can be seen in Figure 4.

Control will be semi-autonomous. The operator will control the direct the robot moves, but the robot will autonomously path plan for obstacle navigation. Manipulation will utilize inverse kinematics via MoveIT so that the operator can graphically set overall position goals relative to the robot in space that the arm will then perform (such as grasp an item, or place an item), rather than direct control of individual joints.

Overall, minimal training is required to control the robot. Ideally, the operator should have practice with the system by maneuvering over test obstacles and manipulating the arm to feel comfortable with it.

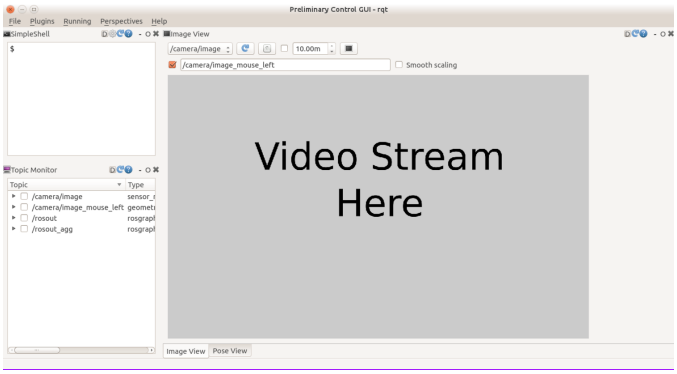

Fig. 4. Operator GUI

III. Application

A. Set-up and Break-Down

The robot is a self-contained system. Any laptop/desktop with the designated control software installed may be used as the operation station.

1) Set-up:

- Remove screws on top, and top case with arm.
- Install charged batteries in chassis and arm.
- Close case, put back screws.
- Reconnect Ethernet to arm which comes out of near back of robot.
- Close main power switch. Internal computer will be set to automatically turn on and launch our system in ROS to allow for control from the operator station.
- Connect drive and arm control joysticks to operator station. Power on operator station.
- On the operator station, connect to the mission control network; the robot computer will automatically connect to this wirelessly.
- Launch the control interface on the operator station which will communicate over the mission control network to the robot.

2) Break-down:

- Use the control interface to reset the arm to the folded position.
- Stop all nodes launched and stop roscore on the robot computer.
- Robot may be safely powered down with main power switch.
- Stop the control interface software.
- Shutdown the operation station.
- Remove and charge batteries in chassis and arm.

B. Experiments

1) Simulation: Testing in simulation has been very useful for us. For SLAM Mapping, we had the robot drive around in sample robocup rescue worlds from a simulation of NIST standard test methods. We were able to create the map shown in Figure 5.

The 2D map is helpful for simple areas, but to account for 3d features, we plan on using an RGBD camera provide point

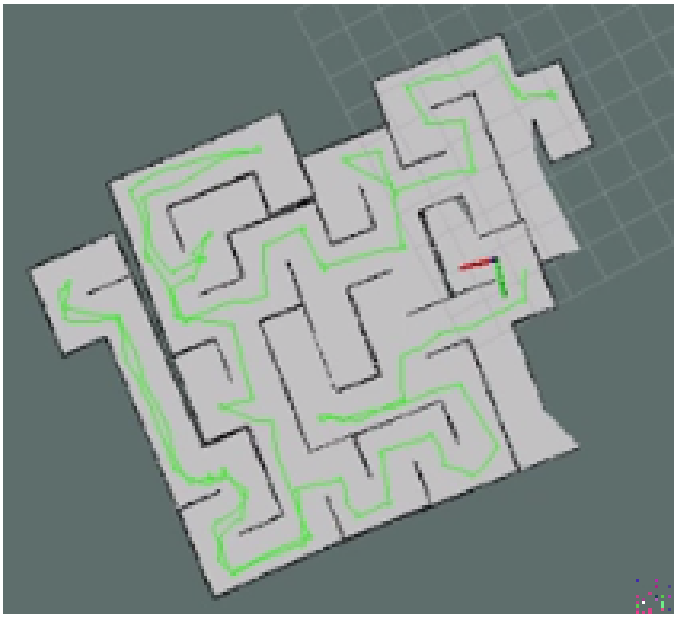

Fig. 5. Map created with Hector SLAM

cloud data. Knowledge of the 3d features allows for more efficient path predictions and for more successful obstacle traversal. We plan on formatting this into a 3d occupancy grid using the Octomap package.

We have also been able to use simulation testing for the development of path planning algorithms.

2) Old Platforms: We were able to take advantage robots developed for past competitions to test parts of our software and sensor stack before our drive train and chassis were finished being assembled. This allowed us to identify the differences between the simulation of our sensors and their real actual performance. Taking advantage of our old platforms has ultimately lead to faster software development and more refined sensor data filtering and handling.

3) Physical Obstacles: To test the performance of our hardware in conjunction with our software we have begun to physically construct many of the RoboCup Rescue obstacles. The specifications and instructions were taken from the 2011 NIST RoboCup Rescue Arena Assembly Guide [1] and the results can be seen on YouTube here.

C. Application in the Field

By engaging in this project, we have produced a durable robot currently capable of navigating uneven terrain and performing complex arm manipulations with relative ease. With further improvements, our system will be useful in a search mission exploring unknown territories, relaying sensory data to investigators. Additionally, the individual design of components being developed, such as the arm end-effector, pose utility on their own, each helping responders to understand and react to their environment independent of our specific robot. We still require development to ensure our robotics system is mechanically and electrically robust and that the interface will convey meaningful feedback to the human operators. We are looking forward to future field tests to ensure this.

IV. Conclusion

The team is making good progress with our development. Our system is fully operable under teleop control and development to add more autonomous functionality is well under way. Mechanically, everything has been designed and built except our manipulator. All of the manipulator parts and the PCB arrived this week and we expect assembly to be done shortly.

While the team has not competed in the the RoboCup Rescue Challenge before, we have a lot of experience competing in similar competitions such as the NASA RASCAL RoboOps Challenge, the NASA SRC Centennial Challenge, and the UAE Robots for Good competition. These experiences have taught us what it means to be competition ready, and the team is prepared to prevent on-site problems through thorough debugging. We also have experience traveling for competitions and are aware of the extra preparation needed to ship a robot and bring along any tools or extra parts we may need.

Appendix A
Team Members and Their Contributions

• Drew Beller	Drive train lead
• David Mayo	Vision lead
• Reo Baird	Software lead
• Michelle Tan	Mapping lead
• Rita Ainane	Electrical lead
• Jasmine Palmer	Drive train lead
• Nadya Balabanska	Vision lead
• Joey Muller	Control GUI lead
• Lukas Leo Beyer	Arm lead
• Wei Wu	Treasurer
• Tatsuya Daniel	Fund-raising/Business lead
• Sandra Walter	End effector
• Selena Feng	Arm IK
• Akhilan Boopathy	Arm IK
• Eyob Woldeghebriel	Mapping
• Rodrigo Tocalini	Control GUI
• Chris Madrigal	Control GUI
• Nate Foss	Control GUI
• Timothy Truong	Mapping
• Matthew Horton	Mapping
• Quang Le	Mapping
• Dillon Zhang	Mapping
• Angela Leong	Mapping
• Kara Luo	Vision
• Vlad Seremet	Vision
• Franklin Zhang	Mechanical
• Gabriel Li	Drive train
• Jennifer Lauv	Drive train
• Jeremy Sogo	Mechanical
• Johnny Fung	Drive train
• Kevin Zheng	Mechanical
• Mario Contreras	Drive train
• Marissa Steinmetz	Mechanical
• Raymond Tse	Drive train
• Sandra Walter	Manipulator design
• Xavier Alexander Zapien	Drive train

TABLE I
DRIVE SYSTEM

Attribute	Value
Name	MITBot
Locomotion	tracked
System Weight	35 kg
Typical operation size	0.5 x 0.8 x 0.4 m
Startup time (off to full operation)	5 min
Power consumption (idle/ typical/ max)	120 / 270 / 520 W
Battery endurance (idle/ normal/ heavy load)	260 / 120 / 60 min
Maximum speed (flat/ outdoor/ rubble pile)	4 / 4 / 2 m/s
Payload (typical, maximum)	22/ 45 kg
Support: set of bat. chargers total weight	2 kg
Support: set of bat. chargers power	500W (100-240V AC)
Support: Charge time batteries (80%/ 100%)	90 / 120 min
Cost	2500 USD

TABLE II
MANIPULATION SYSTEM

Attribute	Value
Locomotion	tracked
System Weight	19.5kg
Typical operation size	0.66 x 0.96 x 0.24 m
Startup time (off to full operation)	1 min
Battery endurance (normal)	120 min
Payload (typical, maximum)	20/ 45 kg
Maximum operation height	1.3 m
Payload at full extend	9.1kg
Support: set of bat. chargers total weight	2 kg
Cost	35000 USD

APPENDIX B
CAD DRAWINGS

- End Effector PCB — Figure 2

APPENDIX C
LISTS

A. Systems Lists

- Drive System — Table I
- Manipulation System — Table II
- Operator Station — Table III

B. Hardware Components List

See Table IV.

C. Software List

See Table V.

TABLE III
OPERATOR STATION

Attribute	Value
Name	Mission Control
System Weight	1 kg
Weight including transportation case	1.5 kg
Transportation size	.3 x .2 x .01 m
Typical operation size	.3 x .2 x .2 m
Unpack and assembly time	1 min
Startup time (off to full operation)	1 min
Power consumption (idle/ typical/ max)	60 / 80 / 90 W
Battery endurance (idle/ normal/ heavy load)	10 / 5 / 4 h
Cost	1000 USD

TABLE IV
HARDWARE COMPONENTS LIST

Part	Brand & Model	Unit Price (USD)
Drive motors	IG52-04 with Encoder	155
Drive gears	N/A	
Drive encoder	N/A	
Motor drivers	Roboteq MDC 2460	395
Batteries	K2 25.6V LiFePO4	360
Computing Unit	GIGABYTE LGA1151 Intel Z170 ATX	150
WiFi Adapter	Ubiquiti Networks Router (ER-X)	50
Lidar	Neato Lidar VX-11	120
Stereo Camera	Carnegie Robotics MultiSense S21	?
Cameras	Logitech C930e	85
Infrared Camera	FLiR Dev Kit	260
CO_2 Sensor	MG811 CO_2 Sensor	35
Battery Chargers	Smart Charger for 25.6 LiFePO4	50
6-axis Robot Arm	Harris Redhawk MPR	35000
Drive Train	Super Droid HD2 (Chassis and Drive Train only)	2500
Operator Laptop	Any Laptop	?

TABLE V
SOFTWARE LIST

Name	Version	License	Usage
Tensorflow	1.0	open	Vision
Ubuntu	16.04.02	open	System
ROS	kinetic	BSD	System
OpenCV	2.4.8	BSD	Vision
Hector SLAM [2]	0.3.5	BSD	2D SLAM
octomap	1.8.1	BSD	Probabilistic 3D Mapping
Hector NIST Arenas Gazebo	?	BSD	Debugging
Image Transport	1.11.12	BSD	Video Streaming
CV Bridge	1.12.4	BSD	Communication
Gazebo	8.0.0	BSD	Modeling
rqt	0.1.2	BSD	GUI
Pioneer SDK	P2-DX	open	Modeling

ACKNOWLEDGMENT

The authors would like to thank the MIT Edgerton Center for providing the support and space for making our submission possible. We would also like to thank our sponsors 5D Robotics, Northrop Grumen, the MIT Innovation Initiative, and Dassault Systems.

REFERENCES

[1] A. Jacoff and S. Tadokoro, "RoboCup Rescue Arena Assembly Guide," , 2011.

[2] S. Kohlbrecher, J. Meyer, O. von Stryk, and U. Klingauf, "A flexible and scalable slam system with full 3d motion estimation," in *Proc. IEEE International Symposium on Safety, Security and Rescue Robotics (SSRR)*. IEEE, November 2011.