RoboUtes Final Paper for NIA RASC-AL Robo-Ops Competition

Dan Andersen, Troy Arbuckle, Elias Bagley, Abhijit Boppana, Parker Conroy, Kyle Crandall, John Daniels, Jonathan Davies, Brandon Gibson, Eric Keeney, Manikantan Nambi, Serei Panh, Graham Pedersen, John Reed, Nicklaus Traeden, Eric Vega, Dustin Webb, Russell Weber

University of Utah

Advisor: Mark Minor PhD

RoboUtes Final Report for NIA RASC-AL Competition

Troy Arbuckle, Parker Conroy, Jon Davies, Nicklaus Traeden, Dustin Webb

The University of Utah, 201 South Presidents Circle, Salt Lake City Utah, 84112

The RoboUtes are; Dan Andersen, Troy Arbuckle, Elias Bagley, Abhijit Boppana, Parker Conroy, Kyle Crandall, John Daniels, Jonathan Davies, Brandon Gibson, Eric Keeney, Manikantan Nambi, Serei Panh, Graham Pedersen, John Reed, Nicklaus Traeden, Eric Vega, Dustin Webb, Russell Weber
With Professors; Mark Minor PhD

Abstract. The RoboUtes are overcoming the challenges of the RASC-AL Robo-Ops Competition by fielding a sophisticated rover. This rover utilizes cutting edge chassis design, modern construction materials, and industry leading manufacturing techniques. The suspension of the rover is built of a flexible spine that allows the front and rear wheels to move independently apart from each other in two axis. This passive suspension allows for adaptability to the environment keeping each of the wheel engaged at all time. A four degree of freedom arm is used to access rocks located in the workspace in front of the rover. Software system employs a distributed architecture to simplify control, handle telemetry acquisition, and manage the constraints of the wireless connection. Furthermore computation is carried out by energy efficient embedded computing systems.
The RoboUtes team has engaged the community by promoting the excitement of space exploration and how science and engineering are dynamic and interesting fields. This is being achieved through an engaging social media experience and online presence. Through outreach, members of the public have been able to touch parts of the rover and talk to engineers about science and robotics, peaking their own understanding of the field. RoboUtes members are directly influencing members of the community through volunteer work, mentoring programs, and live, hands-on demonstrations of the rover.

Keywords: RoboUtes, Robotics, Rover, Teleoperation, University of Utah, RASC-AL, NASA, Education

INTRODUCTION

The RoboUtes set out to build a sophisticated mobile robotics platform to accomplish the design criteria of the RASC-AL Robo-Ops Competition. The team entered into this competition to experience a challenge that utilizes their collective engineering skills and knowledge. The competition presents a unique experience to operate on NASA's facilities and work with cutting edge technology. The RoboUtes team is an interdisciplinary organization. Members were assigned to tasks based on experience level and interest. The team set out to create the simplest solution to the challenge possible. The defining feature of the RoboUtes rover is the compliant spine system developed by Dr. Mark Minor. This passive system is a simplification of traditional suspension systems and hardware components. Simplification of the rover's hardware systems moves complexity into the software systems. Additional desired features move more complexity into the software.

The team faced many challenges and setbacks throughout the design process including monetary issues, design issues, product lead time, organizational structure, conflicting work and school scheduling, team member availability, and equipment access. These setbacks lead to a number of design changes that altered many of the originally proposed features. The largest setback was the delay of funding after receiving the grant. Since the RoboUtes is a new student club financial accounts had not been set up with the university, as such a large amount of construction time was lost. The team could not buy expensive parts until a charge account was set up, which did not happen until the beginning of April. Team members used personal funds to purchase materials when necessary. However, this was not always possible for large budget items and many deadlines could not be met due to these limitations. The inter-team deliverables were pushed into the same time frame that final projects in engineering classes were due. This high stress environment caused multiple members to quit the team. Despite these obstacles the RoboUtes were able to contribute towards many events and attract a lot of public attention along with creating a rover that meets the challenges of the Robo-Ops competition.

MECHANICAL SYSTEMS

Chassis

The chassis of the rover consists of a front and back module connected by a compliant spine made out of stainless spring steel. The compliant spine system acts as a passive suspension system, allowing the front and back modules to articulate independently as the rover traverses the varying terrain. This system minimizes the weight and hardware complexity of more traditional suspension systems. Strain gauges along each side of the spine allow the user to know the orientation of the rover at all times. The chassis of the RoboUtes rover is based on research conducted by Dr. Mark Minor at the University of Utah. The foundation of the chassis is constructed out of quarter inch polycarbonate with a minimal number of aluminum supports. Additional thinner pieces of polycarbonate are used for protective shielding. The manipulator arm is located at the front of the rover, near the front nose camera. The majority of the electronics and computing systems are located on the rear module along with a deployable boom camera.

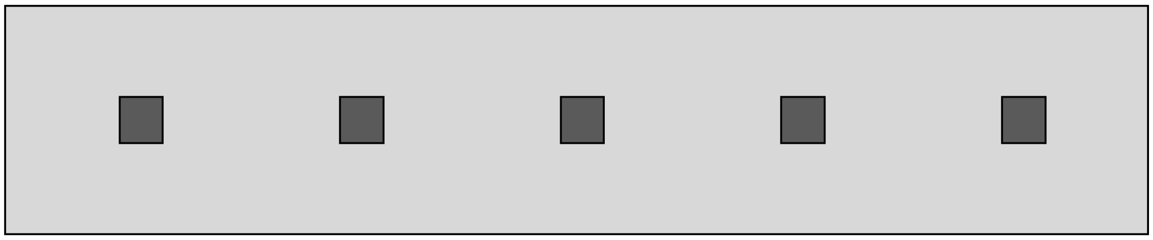
Figure 1. Five strain gauges are attached to both sides of the spring steel spine. The strain gauges detect the bending in the spine which is used to calculate the curvature of the spine. The spine curvature is modified by the drive motors to achieve the desired trajectory.

Drive System

The drive system uses four independently driven motors attached by chain to sixteen inch bike wheels. These wheels provide the required ten centimeters of clearance. The axle of the wheels is rigidly attached to the chassis of the rover and bearings are located inside the wheels. Motors from Anaheim Automation were implemented to provide the desired amount of speed to traverse the field quickly and provide a significant safety factor on torque to move up the hill and over obstacles.

The leg system originally proposed used a worm gear drive to raise and lower the rover's height. This was found to be implausible considering the money constraints of the competition. Also, any leg motion system that had enough torque and was non back-drivable proved to be either too expensive or too heavy. The leg system was abandoned, so to compensate for lower clearance the team chose to go with larger wheels.

Four Degree of Freedom Manipulator Arm

The rover was equipped with a four degree of freedom manipulator arm and gripper. The arm is mounted on the front of the chassis so that it can collect rocks as the rover approaches them, and deposit the rocks into a collection bin which is also located in the front of rover. The arm's work space will be directly in front of the rover where it is visible to all vision system. The links of the robotic arm are constructed of light weight aluminum channel. To enable quick development, hobby servo motors were used to actuate the arm. To reduce the torque load on the motors, the shoulder and elbow motors were mounted on the turn table at the base of the arm. The elbow is then actuated using a belt and pulley drive system running though the aluminum channel. Similarly, the servo motors used to actuate the wrist and gripper were mounted on the elbow. The wrist is actuated using a belt and pulley system, while the linear motion to actuate the gripper is being provided by a cable connected to a lever arm actuated by a servo motor. The gripper is designed to pick up rocks of various sizes and shapes. The design of the gripper was simplified from the proposed design and a camera will be mounted on the palm of the gripper as an optional vision source for the operator. This will enable the operator to pick up rocks more efficiently. Also, a range finder on the palm of the gripper will be used to provide haptic feedback to the operator, and to act as a fail-safe mechanism to prevent the gripper from crashing into the ground.

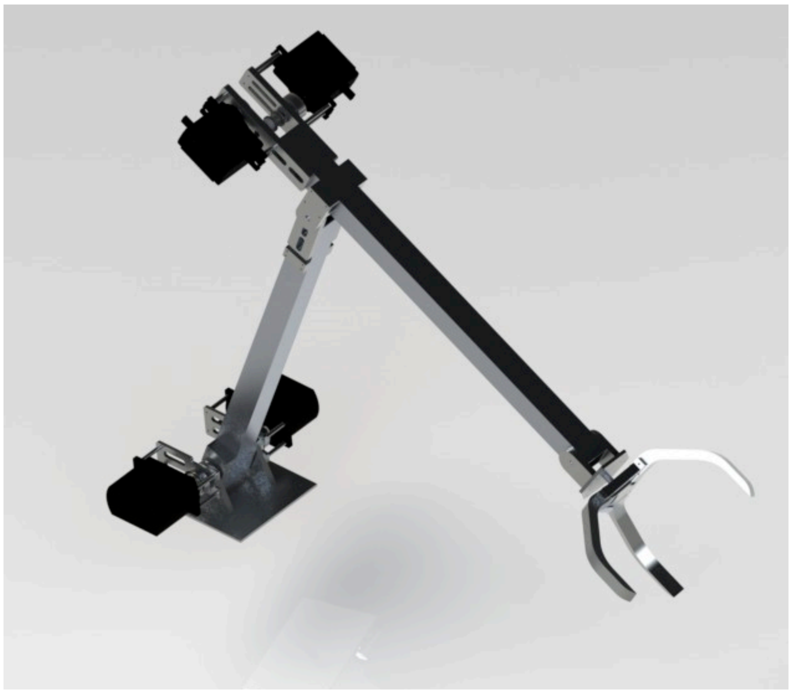

Figure 2. Rendered image of the manipulator arm

Boom

The rover is equipped with a boom that elevates a camera above the chassis which increases the camera's field of view while providing both pan and tilt capabilities for image acquisition. The boom is constructed of 25 mm aluminum square tubing and is 70 cm tall with the camera mounted on top. The proposed carbon fiber proved to not cost effective. The boom can be folded down across the top of the chassis for compact storage. A servo motor is used to raise and lower the boom as needed. To avoid the need to continuously power the servo motor, the boom will be locked in place when it is in the upright position.

Power

The power system is comprised of eight lithium iron phosphate batteries (LiFePO4), each are rated at a voltage of 3.2V and 20Ah which provide a constant current of 100 A at 3.2V or as the motors require 24V thus 13 amps are supplied. Having batteries that are modular allows for placement of batteries in various positions on the chassis for weight distribution. Devices such as control and data collection require relatively minimal power compared to the drive system motors. Other than the motors, the batteries are the heaviest component on the rover. The bus used control the power distribution is designed to handle the 24V that the motors require while providing lower voltages for control and sensor devices.

CONTROL SYSTEMS

The control system on the RoboUtes rover is composed of three core components, the Command Center User Interface (CCUI) client, the network communication layer managed predominantly via Player [1], and on board control and data acquisition. This segmentation has allowed for parallel development of many pieces and permitted RoboUtes members to contribute back to various open source projects and in turn the robotics research community at large.

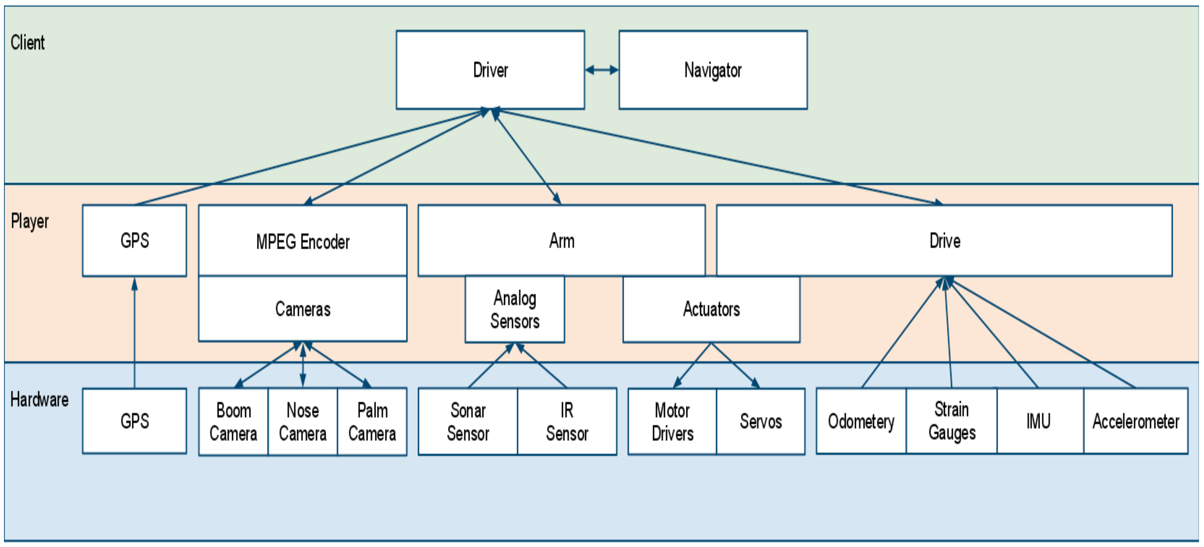

Figure 3. Control diagram for RoboUtes rover

Command Center

CCUI allows the operator to control the rover and view the video feeds and other telemetry. It is written in Java for cross platform compatibility, and uses Javaclient for Player/Stage to communicate with the rover. In order to make operating the rover as easy as possible, our Command center interfaces with an Xbox 360 controller for driving, and a Novint Falcon for 3D arm manipulation. In addition, the system provides features that allow members of our team other than the operator to assist in analysis of photos from the boom cam and mapping out a path for the operator to follow.

The expectation is that multiple users involved in the control of the rover will want to have access to much of the data from the rover, without having to constantly peer over the shoulder of the operator. To facilitate this, CCUI is designed to be run in multiple instances on a small, local area subnet. In particular, there are three roles that users can have when using the CCUI: operator, navigator, and viewer. The Command Center program can be run in any one of these three instances. In general, the architecture of the system is designed so that there is one operator, one navigator, and multiple viewers.

Support for these various roles is provided by CCUI, which also provides as much information as feasible to all parties involved in the operation of the rover. This is achieved through the use of 4 separate windows: The Operator Window, the Telemetry Window, the Map window, and the Image Viewing window. Each of these windows serves a particular purpose in support of the contest goals, but in general it is expected that most users will only be concerned with one or two of these windows, and will ignore the others.

The operator of the rover is primarily concerned with the Operator Window, which is not displayed to anyone else using the Command Center. This window provides a video feed view from one of the cameras on the rover, and a GPS overlay of the field as well. The operator uses the Xbox controller to steer the rover, pan and tilt the currently active camera, switch cameras, or take high resolution still photos. He can also switch into Arm Mode, where the rover will stay locked in place and the operator switches to using the Novint Falcon to control the movement of the arm in order to pick up the rocks. In doing so the video feed will automatically switch to the Palm Camera, and the arm will automatically un-dock to a position in front of the rover so that the operator can easily guide it to pick up rocks. Once the rock is picked up, the operator can simply press a button and the arm will go back to its position above the bucket and place the rock there.

Figure 4. Novint Falcon™ Haptic interface. Image copyright Novint Technologies.

The navigator primarily uses the Map and Image Viewing Windows. They can use the Map Viewing Window to see where on the field the operator is, and also to mark locations and paths for the operator to follow. Any locations and paths drawn by the navigator are immediately viewable by the operator in the Operator Window and all Viewers in their Map Windows. The Navigator can also view any high resolution photos taken by the operator, and uses these to determine where on the field rocks are located to assist in determining a path for the operator to follow.

The viewers may use the Image Viewing and Telemetry windows. They assist the navigator in determining where rocks might be, and can also monitor telemetry from the rover to determine its health. Viewers cannot draw on the shared map and must communicate any desire to do so to the Navigator. This allows the Navigator to maintain a coherent navigating plan for the operator, instead of having multiple users cluttering the map by marking it up excessively.

Network Communication

A great deal of effort has been put into addressing the latency and bandwidth constraints of the communication channel. Examples include limiting video transmission to only one camera at any given time, encoding the data, and building in functionality that permits the rover to place its end effector in key locations upon request.

Command Center

Since there are multiple instances of the Command Center running for a single rover, significant networking communication is required. In addition, we wish to minimize bandwidth use by the rover and avoid overloading the operator's Command Center instance with requests to relay data from the rover. To this end, we have designed a system that ensures that the operator's instance of the command center is the only instance that is requesting data from the rover. Furthermore any data it receives is transmitted at most once. In particular, there are three types of information that get relayed: telemetry data, video data, and high resolution image data. In addition, we also need to communicate map information between the Navigator and everyone else.

Telemetry data is simply broadcast in a User Datagram Protocol (UDP) packet to all users on the same subnet as the operator. The design assumption here is that the Command Center for the rover is going to be in a single location and the users of the Command Center software will have control over the subnet that they are using. Also, we assume that it is not critical if any Command Center instance misses a single update, since the data will change rapidly anyway. Any Command Center instances on the subnet will listen for the broadcast packets and update their data accordingly.

Video Data is relayed to the NASA server by the operator's Command Center instance. It is not relayed to any other Command Center instances, because it is not critical for most other users to see the video stream from the operator at any particular moment, and others in the command center can simply view the official video feed.

High resolution Images are sent to a File Transfer Protocol (FTP) server where they can be polled and downloaded by any Command Center instance.

Communication between the navigators and others is relayed via Transmission Control Protocol (TCP) through server specifically meant to handle map information. The use of TCP ensures that all Command Center instances have correct and up to date map information. The server was made separately process from the Command Center program to avoid the complexities of mixing server and UI code while providing more flexibility with regards to the location and configuration of the server. For example, the server can now be hosted on a well-known static IP.

Rover Interaction

Communication between the Command Center and the on-board processing components is handled via the Player Project. As stated on the Player Project website:

"The Player Project creates Free Software that enables research in robot and sensor systems. The Player robot server is probably the most widely used robot control interface in the world. Its simulation backends, Stage and Gazebo, are also very widely used."

Using Player has allowed us to take advantage of the large amount existing work, to ensure a more stable system, provided us with a simple architecture to build upon for our unique needs, given us an opportunity to contribute to the robotics research community, and permitted us to begin developing the control code before the rover was complete.

A number of components needed to be added to the Player server to facilitate the necessary functionality. These included adding support for a number of devices which consist of:

- the Lynxmotion SSC-32
- the Phidgets GPS
- the Phidgets 3/3/3 IMU

It was also necessary to add support for controlling multiple cameras and to add support for compressing video streams. Talk about the fact that Player has this functionality but it's not setup well to handle low bandwidth networking environments.

Another driver will manage operation of the arm. The arm driver takes a 3D vector representing a position in the work space and calculates the inverse kinematics for the arm to determine how to move the servos. One unique feature of the arm is that the hand is always pointed down. This ensures the palm camera is always facing the work space. Furthermore, it allows the use of IR and sonar sensors in the palm to determine the distance of the hand to the ground. This information is used to slow the movement of the arm as it approaches the ground reducing the likelihood of damage to the hand as a result of impact with the ground. The data is sent back to the Command Center to provide feedback to the user.

Finally a driver was built to handle driving the rover. This was necessary to simplify the operator's task. Otherwise the operator would be responsible for operating each of the four wheels individually. This part of the program takes a trajectory vector from the operator, as well data from a set of strain gauges, the wheel encoders, an IMU on the rover's front module, and an accelerometer on the rover's rear module to determine what accelerations need to be applied to each wheel at any given moment in time.

Because the user interface is written in Java and Player project does not support Java, we used Javaclient, an unsupported client to for the Player server. This client needed to be upgraded in a number of ways. First, it was necessary to add support for the UDP network protocol to support the video system. It was also necessary to add support for the Opaque Interface as it was not in functioning order when we started using the project.

On-board processing

The onboard component splits into two components; specifically the video processing was isolated to its own computer. This was done to minimize the impact of image encoding on the rest of the system.

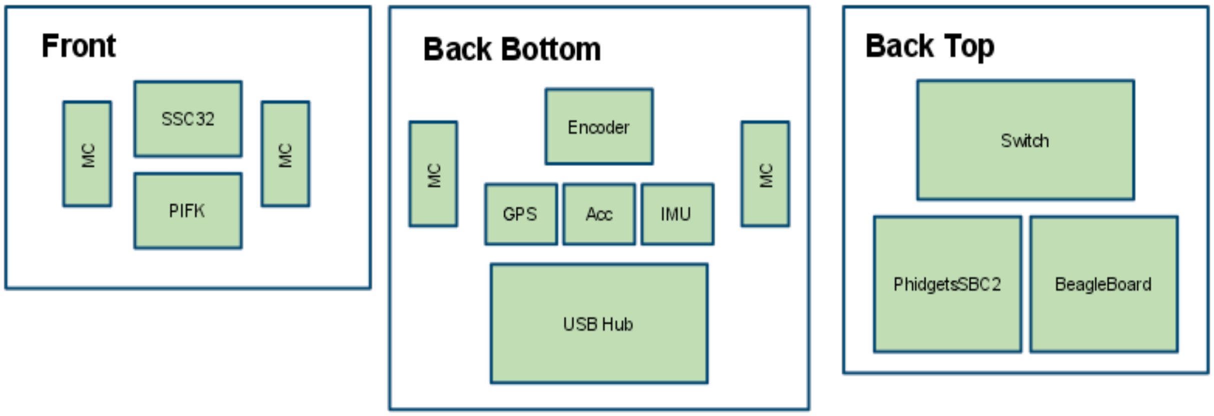

Figure 5. Position of components on the front and back modules

Primary processing is distributed across two main computers on the rover. One is a BeagleBoard-xM and is used to handle image processing. The other is a PhidgetsSBC2 and is responsible for aggregating all other telemetry as well as managing all actuators. These boards are supported by:

- PhigetsGPS module
- Phigets 3/3/3 IMU
- Phidgets Accelerometer
- Phidgets 8/8/8 Interface Kit
- LynxMotion SSC-32

Each of these devices interface with the control boards via USB and the two computers communicate with the Command Center via a CradlePoint MBR 1000.

Drive

The drive system on the rover is the most complicated component from a software perspective. The complexity is a result of the non linear nature of determining and controlling the curvature of the spine. Simplifying this complex problem to simple directional commands is a very challenging problem to overcome.

The operator simply commands the rover's speed and direction. From the user's perspective they are just driving the front module of the rover. The drive control system must then determine how the rear module should move. Data from the strain gauges along the spine is used to estimate the its configuration. From this information the relation between the front and rear modules can be estimated at which point the trajectory for the rear module can be calculated. The rear motors will then align the spine to insure the correct trajectory as the rover traverses the field. The RoboUtes are experimenting with the IMU to estimate the orientation of the front and rear modules. This information will be fed back into the system to improve the estimation of the spine configuration.

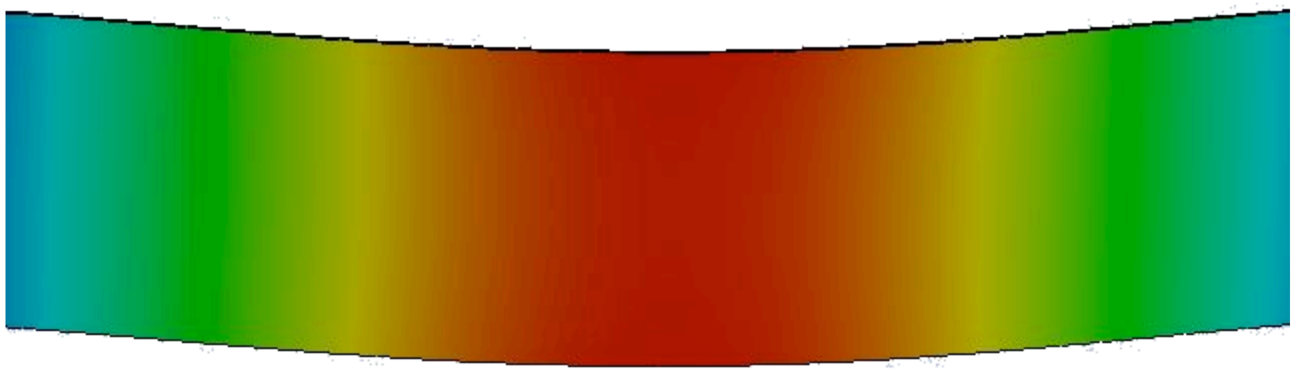

Figure 6. Image showing bending within the spine during horizontal flexure

Vision

The camera system relies heavily on the program Player. Player (along with its associated simulator program Stage, or Player/Stage) is an interface for rover devices that provides a straightforward way for programs to connect to and interact with them. It was designed under a client/server model in which the rover runs a server and a connecting control program acts as client. When various devices are attached to the rover's BeagleBoard, the user writes a configuration file that Player uses to set up interfaces through "drivers" written by the project developers. These provide a level of abstraction where, for example, a particular camera model will be accessible via the Player server as a generic "camera" interface.

Using Player, the camera system's design is relatively straightforward. We write a configuration file that defines each attached camera by their device address (e.g. /dev/video0) and sets up a camera interface. The program that receives camera data is written in Java and uses a wrapper for Player functions called Player/Stage Javaclient. This program connects to the Player server, registers interfaces to each of the cameras, and then displays received camera data from the selected camera to a viewport window in the program.

One important challenge when developing this camera system was that the provided camera interfaces, while intuitive and useful for capturing high-resolution data infrequently, were insufficient for sending camera data that required near-live feedback over a low-bandwidth connection. The Player drivers for our cameras either provided no compression or offered nothing more than JPEG compression – simply unusable for sending frames many times a second. To solve this, our team wrote our own Player driver. This driver first reads raw camera data and then uses the FFMPEG and libav-codec libraries to encode each frame into an MPEG2 packet, sending across this compressed data instead. This addition saw greatly improved efficiency, as frames that used to be 40-50 KB JPEGs were reduced to 1-10 KB MPEG frames.

To mirror the MPEG encoding on the rover, our viewer program must be able to properly decode it. The readily-available methods of abstracting away MPEG decoding (e.g. the Java Media Framework) geared towards connecting to a typical URL-based video stream to decode video, which did not apply to our connection which uses Player. Instead, we used the library FFMPEG-Java, a Java wrapper for the same FFMPEG libraries used to encode the video. With all these parts in place, the camera system can connect to the rover and receive compressed frames to display to the rover operator.

In addition, the camera system needed to support pan-tilt controls on any attached cameras that possessed that functionality. The Player program offers a “ptz” interface that lets a client program package up desired pan/tilt/zoom settings to send as a command to a supported camera device.

EDUCATION AND PUBLIC OUTREACH

The RoboUtes Robo-Ops team has actively worked on bringing the public along through the competition and generating interest in space exploration and robotics. The group has assembled a marketing and community team which has worked with individuals from across the campus. The RoboUtes have created a dynamic web experience and have engaged the public in multiple outreach events. Team members also met with the outreach department in the University of Utah's College of Engineering where they formed a partnership to interact with K-12 affiliations and demonstrate the project.

To engage the science and robotics oriented youth communities, RoboUtes filled many positions at the 2011 Utah FIRST Lego League and the 2011 Utah FIRST Robotics competition including, judges, referees, scorekeepers and safety volunteers. We also set up high traffic tables at these events to let the public know about the competition. Early in the build season the RoboUtes attended Meet an Inventor Night, an event where local K-12 students could come up to the University of Utah and meet some of the more prolific inventors on campus. RoboUtes Vice President gave a half hour presentation about extra-planetary rovers, the competition, and the engineering process. The RoboUtes also tabled at the Mechanical Engineering Design Day and the Grand Kerfuffle where we interacted with University of Utah students. The RoboUtes also held workshops open to the public to learn about various concepts from the rover design. This includes sessions on Solidworks, Player, and other processes used in the project.

Along with person interactions, the RoboUtes marketing team created a dynamic web experience including multiple social media websites to engage the community. The marketing team developed the team web page which includes info on the team and the competition and will stream the incoming video from the rover. The RoboUtes Robo Ops Facebook page and Youtube channel were put up very early in the competition. The Facebook page offered a great way to update the community on what was going on throughout the design process, with new videos being uploaded on Youtube. The page was also used to link to interesting news and information about space exploration throughout the season.

The RoboUtes planned to take the rover to the local planetarium and local K-12 schools. They also wanted to drive the rover around the University of Utah campus to generate buzz on campus. Funding delays, however, pushed back the construction of the rover, forcing these activities to be pushed into May. Loss of personnel has forced Xbox Live App functionality to be abandoned as well.

CONCLUSION

The RoboUtes RASC-AL Robo-Ops competition team faced numerous setbacks but was able to create a sophisticated mobile robotics platform to accomplish the various challenges presented. The rover utilizes a passive compliant spine system and a four degree of freedom manipulator arm to traverse the variable terrain on NASA’s analogue test field and collect rock samples. The rover will has multiple camera systems and will relay information to an operator team. The rover will be driven with an Xbox 360 controller and the manipulator arm will be controlled using a Novint Falcon haptic device. The RoboUtes have engaged the local community numerous times and gathered a lot of public interest and support.

ACKNOWLEDGMENTS

REFERENCES

1. Zhu, X., Kim, Y., Merrell, R., Minor, M.A., “Cooperative Motion Control and Sensing Architecture in Compliant Framed Modular Mobile Robots,” *IEEE Trans. Rob.,* Vol 23, No. 5, pp 1095-1101, October 2007.
2. Minor, M.A., Albiston, B.W., Schwensen, C.L., “Simplified Motion Control for Compliant Framed Wheeled Mobile Robots,” *IEEE Transactions on Robotics,* Vol 22, No. 3, pp 491-506, June 2006.
3. Minor, M.A., Merrell, R., “Instrumentation and Algorithms for Posture Estimation in Compliant Framed Modular Mobile Robotic Systems,” *International Journal of Robotics Research*, Vol. 26, No. 5, pp 491-512, May 2007.
4. Player/Stage http://playerstage.sourceforge.net/