Realization of Vision-Based Navigation and Object Recognition Algorithms for the Sample Return Challenge

Velin Dimitrov, Mitchell Wills, and Taşkın Padır
Robotics Engineering
Worcester Polytechnic Institute
100 Institute Road, AK123
Worcester, MA 01609
{vdimitrov, mwills, tpadir}@wpi.edu

***Abstract*— We present the improvements to AERO, the Autonomous Exploration Rover, developed for the 2014 NASA Sample Return Robot competition with the intent of enabling more robust and reliable autonomous operation for sample return rovers. The competition requires the robot to navigate a large outdoor area, find and collect various geologic samples, and return to the starting platform all autonomously and utilizing only space compatible technologies. We highlight improvements made in the implementation and deployment of the vision, navigation, and planning systems. We describe the process of modifying the software to be closer aligned with ROS standard practices, resulting in more predictable and stable operation. We conclude by providing a roadmap for the integration of multiple heterogeneous systems in a shared control framework to enable efficient exploration of large unknown environments.**

TABLE OF CONTENTS

1. INTRODUCTION

Many scenarios in robotic planetary exploration require interaction of multiple agents, both human and robotic, but such interaction is impeded by significant time-delays and bandwidth restrictions of the communication channels. Our research focuses on the intersection between autonomous and teleoperated systems, and more specifically on how autonomous behaviors in a system can be combined with user input to enable shared control modalities previously not possible. Prior robotic missions to the Moon and Mars have followed mostly teleoperated mission scenarios, consisting mainly of highly scripted and preplanned task sets uploaded for semiautonomous execution. As humankind's exploration capabilities on the Moon and Mars continue to improve [1], and as missions evolve to include multiple robotic platforms (including rovers, micro-rovers, hoppers, and orbiters) with varying complexity and resources, a paradigm shift in mission design approach is required to rely more on the autonomy of these systems. While this has proven to be an effective approach especially for minimizing mission risks, it poses significant challenges when scaled to multiple systems. In order to effectively implement shared control operation though, more robust and reliable autonomous behaviors are needed. In addition, these behaviors must be validated in a variety of different environments so their actions are well understood.

Figure 1. AERO during field testing in a gravel mining pit.

In this paper we present our work towards more robust and reliable autonomous operation for sample return rovers, with the intent to intelligently integrate the human operator input and intent in future iterations. The work was guided by the 2014 NASA Sample Return Robot (SRR) Centennial Challenge, and focused on the integration of vision-based navigation and object recognition algorithms for the previously developed Autonomous Exploration Rover (AERO), shown in Figure 1. The presented work builds on experiences from developing the rover for the 2013 SRR Challenge [2], and previous work with the Oryx 2.0 rover [3–5]. New training procedures for the cascaded classifiers used previously in [2] led to significantly improved detection and tracking performance. In addition, modifications were made to the previous software architecture to speed-up detection and therefore enable detection in situations where the rover is moving. A GPU based implementation was also developed with significant improvements in detection rates in specific situations. AERO also saw improvements in navigation performance, with a

978-1-4799-5380-6/15/$31.00

completely redesigned navigation architecture based on the best practices suggested in the new Robot Operating System (ROS) control framework.[2] A new high level state machine significantly improved in handling conditions that previously caused navigation faults, and tighter integration between the subsystems in the navigation architecture led to better accuracy. A new method to integrate global information for virtual reality tags was developed and led to improvements in docking back at the starting location. We discuss all of these improvements in detail, and provide insight into our experiences with the algorithms and how they need to be further modified and improved.

Our long-term goal is to leverage these improvements in autonomous operations to design, develop, and validate a human-in-the-loop control framework for enabling operators to control multiple semi-autonomous rovers to complete cooperative manipulation tasks. Control inputs to the system from operators will be limited to high-level, abstract commands, and the framework is expected to autonomously handle the coordination between the robotic systems to comply with the intent of the operator commands. Cooperative manipulation will likely be required to explore areas where the abilities of one rover system are insufficient, because the terrain is too rough, steep, or loose, and specifically when a sample cache may need to be handed off between two systems. These actions will be safer with less risk to mission success and the health of the individual systems with improvements in the autonomous behaviors and actions of the system as a whole.

This paper is organized as follows: Section 2 introduces the AERO platform, Section 3 describes the improvements made to the vision system for sample detection and navigation, Section 4 describes the new navigation and planning framework, and Section 5 outlines the roadmap for future work.

2. AERO

The Sample Return Robot Challenge is an annual NASA Centennial Challenge hosted by Worcester Polytechnic Institute for the first time in June 2012. The premise of the competition is to encourage teams composed of engineers, students, and tinkerers to build fully-autonomous robots that can navigate a large outdoor area, find and collect various geologic samples, and return them to the starting pad within the time limit [6]. The caveat is that only space-compatible technologies are allowed, meaning that global positioning systems, sonar, compasses, magnetometers, and similar technologies are not allowed. These limitation create significant challenges to accurate localization and navigation, requiring solutions similar to what many military systems use in GPS-denied or GPS-corrupted areas. Samples for collection are split into three categories, easy, intermediate, and difficult, based on how much information and detail is provided on each sample apriori. For example, the easiest samples are fully described, and the difficult ones are just noted to be metallic objects of interest that visually do not appear to belong in the environment. For level one only a single easy sample is on the field, but for level two a combination of many easy, intermediate, and difficult samples are on the field.

AERO, the Autonomous Exploration Rover, is a research platform designed originally for the 2013 NASA Sample Return Robot Centennial Challenge. Shown in Figure 1, the rover is comprised of a differential-drive four-wheeled mobility platform and 6-degree of freedom (DOF) manipulator with a fixed suspension. AERO has a footprint of about 99cm by 67cm with a mast height of just under 1.5m and weighs about 80kg. AERO is instrumented with four Allied Vision Manta G-095C cameras in a two stereo pair configuration, a KVH 1750 fiber optic gyroscope based inertial measurement unit (IMU), a 50m SICK LMS151 LIDAR, and wheel encoders. The robot uses the ROS framework to maintain a flexible and modular software architecture. [3] The ROS architecture encourages re-use of code and modules reducing development time significantly.

Fusing a combination of data from the LIDAR, IMU, and wheel encoders, AERO localizes and navigates in the search area without the aid of GPS or compasses. The lower stereo pair has a shorter base-length designed to help the robot see nearby obstacles potentially missed by the LIDAR and to categorize any samples immediately in front of the robot. The information provided by the stereo cameras helps the rover localize samples with sub-centimeter accuracy with respect to the base, used to move the arm to manipulate and store the sample. A second, longer-base length stereo pair is used to identify potential areas of interest to explore. Many samples can be flagged, but not necessarily categorized, up to distances of 20m using the top stereo pair. Once closer, the rover can use its lower stereo pair as described above. Finally, the top stereo pair can locate the home beacon, a box covered in AprilTags [7], at ranges in excess of 50m helping guide the robot back to the starting platform at the end of the search period.

3. VISION SYSTEM

To detect the samples, classify them, and retrieve using the vision system, we implement the standard Haar cascade classifiers from the OpenCV libraries within the ROS system. Once an object is detected, its 3D location information is extracted from the disparity maps generated by the stereo cameras. The details of the detection algorithms is not explained, because the approach has not significantly changed from the 2013 competition. Improvements in the structure and ROS implementation of the detection algorithms were made, but the core approach is the same as in [2, 4].

Significant improvements though were made in detection performance, by generating a better training data set, and iterative adjustment of the training parameters exposed by the OpenCV Haar cascade training utility. In order to train the classifier, training images of each sample need to be provided to classifier training utility. In the previous year, we utilized images of the sample with cloth-lined, lit table-top stages, and random noise backgrounds, resulting in detection performance that was usable but limited in cases of mixed-shade or unexpected orientations of the sample object. We determined that the practice of taking a few sample images and generating thousands of synthetic images using distortions may be sufficient in many cases, but does not provide the necessary robustness in the variable conditions expected during the competition.

We therefore changed the approach to utilize over 3000 hand-cropped images for the precache hook objects in various conditions to generate a very diverse and wide-ranging data set. A small utility that varies the exposure was used during the collection of the data sets to simulate varying lighting

[2] http://wiki.ros.org/ros_control

[3] http://www.ros.org

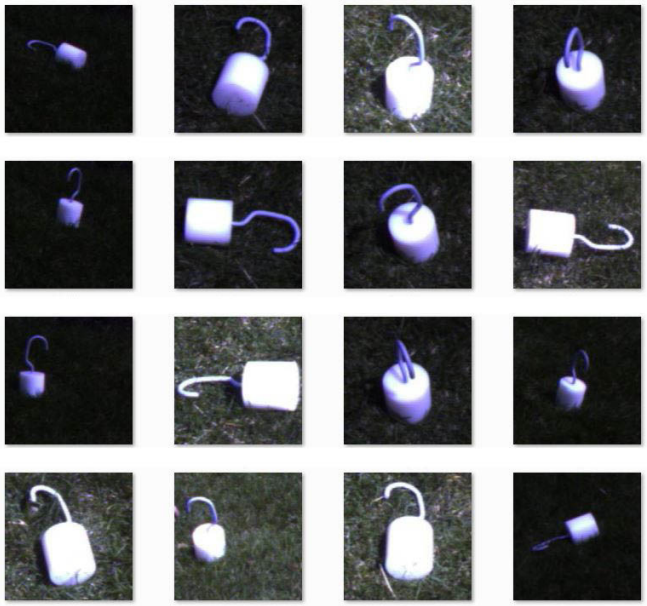

Figure 2. A sample subset of the training images used for the precached hook sample of the NASA Sample Return Robot Competition. Over 3000 such images were used to generate the classifier for the hook.

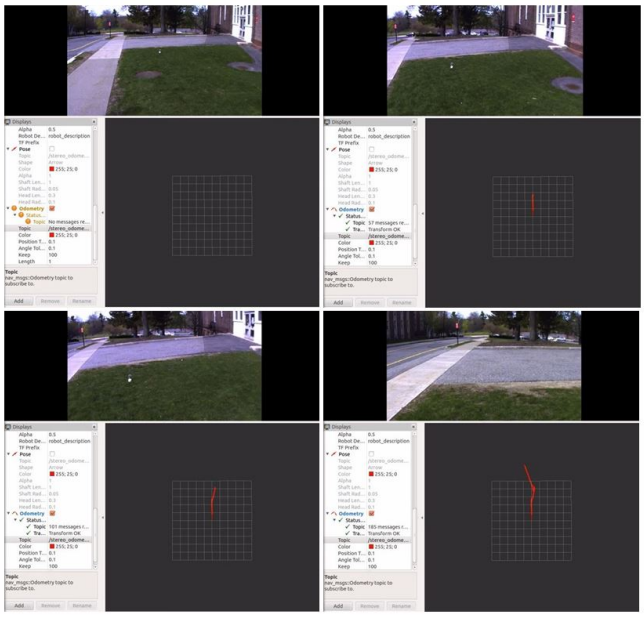

Figure 3. Four screen shots from the ROS visualization tool, rviz, showing the visual odometry package tracking the motion of the robot. The robot drives forward first, takes a slight right, and then a slight left. The red line traces out the path the visual odometry package calculated for the robot. The visual odometry was tested and proved useful in certain cases, but ultimately not used during competition.

conditions. In addition, data sets from various locations were included with various leaves, sticks, and natural debris in the negative sample set to help eliminate false positives from debris in the field. Training on the data-set with over 3000 positive images and 9000 negatives would take on the order of 2-3 days on a modern Core i7 computer. By running multiple simultaneous instances of the single-threaded training program, we could quickly generate multiple classifiers with different training parameters, leading us to a set of parameters for the training that yielded very good classification, even in darkly shaded areas, areas with leaves and sticks, and mixed shade.

In addition to using the CPU based Haar cascade classifier provided by OpenCV, the GPU based classifier, trained on using the same data-set, was also tested. When running on the NVIDIA Tesla K20 GPGPU inside AERO the classifier ran 7 times faster while utilizing only 20% of the GPU's capacity and with minimal CPU usage.[4] Although the GPU classifier significantly outperforms the CPU classifier, the number of detections were significantly fewer than the CPU. Using the CPU based classifier introduced additional delay before the results were reported; however, in practice the delay was not a limiting factor for the speed that the robot could operate at. The effects of the delay that may have caused error in the detected position of the sample while the robot was moving were mitigated by ensuring that the robot had reached a full stop before attempting to accurately detect the position of a sample.

A feature we implemented on AERO, but did not end up using during competition because we only tested it in specific cases, was visual odometry through the ROS implementation of libviso2 [8]. Despite not using it during competition, the algorithm does work well and provides another source of information that can be provided to the Kalman filter resulting in a more accurate estimate of the pose of the robot throughout its operation. The algorithm works by tracking how objects within the frame of the stereo cameras move, and then correlates that motion to the motion of the stereo vision system. Since the camera sensors are over a megapixel each, the processing rate of the visual odometry is slow compared to the dynamics of the system. To overcome this issue, we tweaked several parameters of the default implementation to reduce the number of regions the algorithm is searching, tailor them to be more sensitive in the yaw of the platform by making the regions tall and skinny across the camera frame, reduce the number of points matched in each region, and decimating the input images to a quarter of the source resolution. These modification did not significantly affect the accuracy of the algorithm, but raised the frame rate of the processed pose estimates to about 12-15 frames per second. These simplification proved sufficient to provide usable visual odometry data from the stereo cameras.

4. Navigation and Planning

The implementation of the planning architecture for the robot is done by creating multiple levels of planners that ranges from controlling individual components of the robot to planning the high level tasks throughout the entire run. The planning process is divided into separate systems in order to make each system both easy to understand and to modify. The levels communicate by passing desired actions to a lower level controller that will execute the action and then communicate back the result (success or failure). This allows the higher level controller to try again or attempt to find a different way of completing the task. Figure 4 shows the organization of the different planning systems and the tasks they perform.

[4] https://github.com/RIVeR-Lab/aero_srr_14/blob/hydro-devel/vision/src/cascade_classifier.cpp

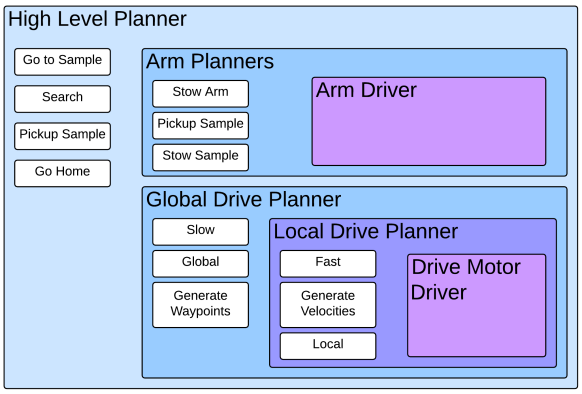

Figure 4. An overview of the organization of the planners on the robot and their tasks.

At the highest level there is a single finite state machine that represents the current high level task that the robot is trying to accomplish, such as driving off the platform, picking up the sample and driving toward the platform. Each of the states represents a different requested action that is dispatched to a lower level planner to execute. This high level view of the robot's current state allows for easy debugging and reuse of similar actions.

While a finite state machine manages the high level operation of the robot, lower level planners manage individual components of the robot. Separate planners run for the arm and drive controllers that allow the high level planner to command actions such as move the arm end effector to a pose or drive the base to a location. The drive system has two levels of planning, a higher level planner that generates waypoints that the robot will use to get to a final goal and a lower level drive planner that generates a velocity for the robot to get to each waypoint.

Local Drive Planner

A local drive planner is implemented using the ROS base local planner[5], which implements the Dynamic Window Approach for collision avoidance as described in [9]. This approach works by sampling the robot's control space and then projecting the result of those control inputs into the future. The resultant trajectories are then scored based on a number of factors including obstacle proximity and goal proximity, ignoring trajectories that would cause the robot to collide. The best control input is then chosen based on score and then executed. This process is executed continuously until the robot reaches the desired waypoint generated by the global planner. If none of the trajectories are valid, because the robot got too close to an object for example, the local planner supports a number of recovery behaviors including backing away from an obstacle in front of it.

As the local planner's goal is in close proximity to the robot (within 10 meters), the planner only operates on a small costmap that is generated as a subset for each new laser scan. As it does not operate on a global costmap that is generated from multiple laser scans, the planner is not affected by potential incorrect localization that causes errors in the global costmap, which could lead to the robot crashing into an obstacle. Additionally because it only has to search paths inside of this region, requiring much less computation than planning an entire route through the environment. This allows the planner to run at 20 Hz allowing the robot to react quickly to changes; while the environment of the competition is static, at times the robots 2D perception of the world through the laser scanner could change quickly. For example, as the robot approaches a hill it slowly inclines upward (if the hill is not too steep) causing it to see the hill further out as it moves upward. If the robot did not replan quickly enough it would see the hill as a wall and try to drive around it.

Global Drive Planner

In order to plan a path to locations that are not near by to the robot, a global drive planner is used to generate local waypoints to the goal that would be sent to the local planner as a goal. The ROS navfn package[6] global planner implementation is used; it implements a global planner that computes a plan from a start position to end position using Dijkstra's algorithm. Because this algorithm has to plan over the entire known environment it can not run nearly as frequently as the local planner. Instead the planner only runs when it receives a new goal or when the local drive planner is unable to find a path to the next waypoint in the current plan. When it runs, a new plan will be generated from the current position to the current goal. When it is detected that the robot has reached (or is very near) the most recent waypoint, a new waypoint will be sent to the local planner.

Arm Planners

Although our arm controller does support planning a path to a desired end effector pose in Cartesian space, it does not support specifying an approach vector or easily avoiding the robot itself. Instead of using a full motion planner to plan the desired trajectory, a number of finite state machines are used to implement planners to execute actions such as picking up a sample or stowing the arm using the controller's built in planning. They are each represented as sequence of positions either in Cartesian space or state space of the arm that the arm would move to. This gives well defied trajectories for the different actions that the arm needs to perform. When picking up a sample, the location of the sample is known to be in a small bounded area on the ground in front of the robot. This allows the intermediate poses that the arm takes to remain the same except for the poses where the fingers initially grasp the sample.

High Level Planner

The high level planner guides the robot through the operation of the competition. For level one of the competition, it can be divided into three sections: navigating to the approximate location of the sample, finding and collecting the sample, and returning to the start. These are implemented in a single finite state machine shown in Figure 5 and described below.

Navigating to the Sample—After the robot finishes booting, it begins executing the state machine. The first thing that is done is to wait for the hardware to finish initializing. This involves waiting for all of the drivers and software to start up and the IMU to finish calibrating. The arm is then placed in the stowed position so that is does not obscure the cameras. Once this is complete the robot prepares to leave the platform by software shuttering the laser scanner so that it does not place obstacles on the local or global costmap. This is done in a number of places throughout the state machine in situations that are known to be safe, but in some cases can lead to the

[5]http://wiki.ros.org/base_local_planner

[6]http://wiki.ros.org/navfn

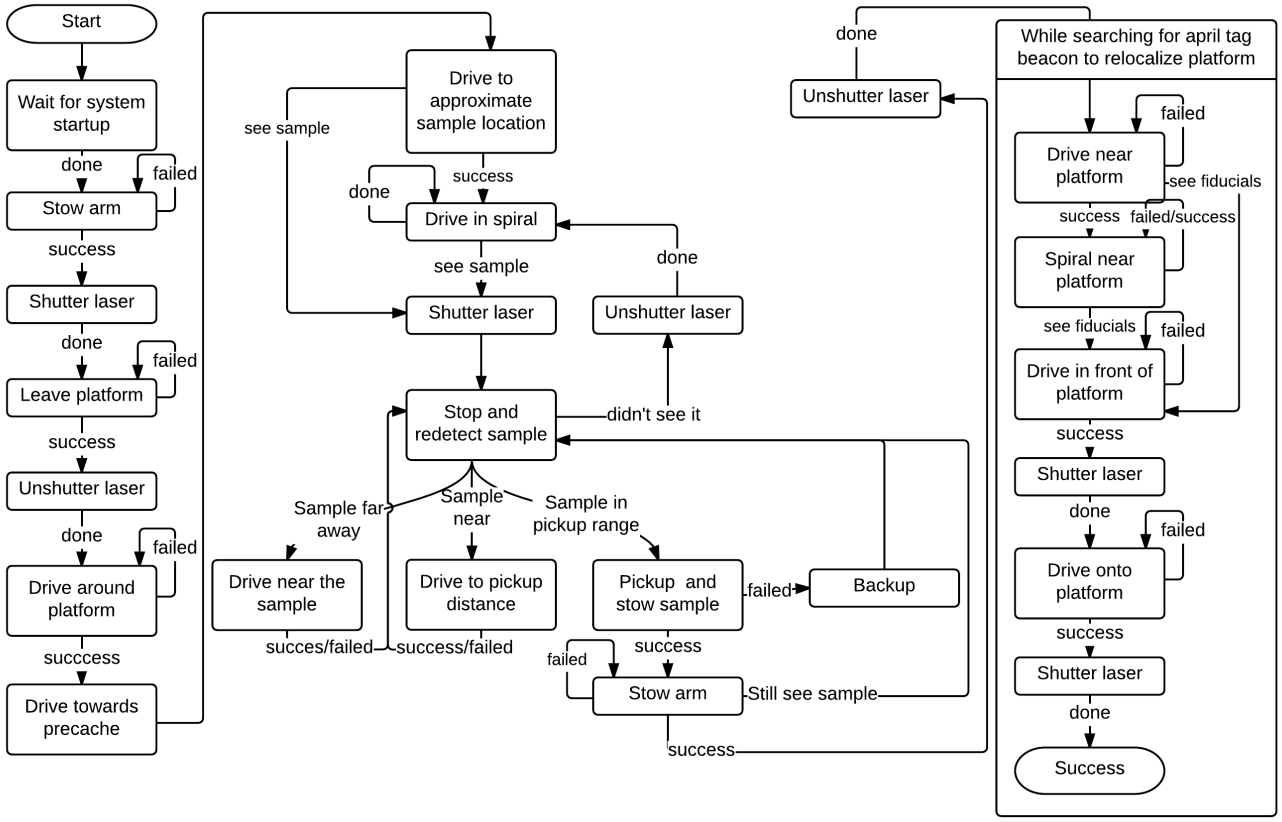

Figure 5. The finite state machine used to complete level one that represents the current high level task that the robot is trying to accomplish, such as driving off the platform, picking up the sample and driving toward the platform. Each of the states represents a different requested action that is dispatched to a lower level planner to execute. Recovery behaviors for each action are also specified so that the robot can perform a recovery behavior that makes sense in the current context.

laser scanner indicating that there is an obstacle in front of the robot when there is not. In this case the laser scanner is shuttered because when the robot leaves the platform it tilts down so that the laser scanner sees the ground. Since it is known that there are no obstacles directly in front of the platform, the laser data can be safely ignored as the robot drives off. The robot then drives straight off the platform by driving a specified distance away from the start and then unshutters the laser.

After successfully leaving the platform the robot then begins to navigate towards the known approximate location of the sample. As the level one sample was known to be behind the platform a path was chosen that would ensure that the robot traveled around the platform. To accomplish this the robot first drives to a position off to the side of the platform and then drives toward the approximate location of the sample.

Sample Location and Collection— Once the robot reaches the edge of the zone that the sample can be found in, it begins searching for the sample. It starts by driving to the center of the approximate location. If it sees the sample while it is driving it will stop and attempt to pick it up. Otherwise when it reaches the center it will begin a spiral search pattern centered there with each rotation separated by a meter ensuring that the entire area is thoroughly searched. If at any point during the spiral the sample is detected, then the robot will attempt to pick it up.

When the robot sees the sample, it immediately stops and waits two seconds so that it comes to a complete stop and any vibrations dampen. After this, the robot attempts to detect the sample again. This is done so that an accurate 3D position can be estimated for the sample using the disparity image from the stereo camera pair. Once the position is estimated, the robot then chooses what to do based on how far away the sample is. Because the position estimation is less accurate at farther distances, the robot does not immediately try to drive up to the sample and pick it up based on one detection, but instead slowly moves closer to it using multiple detections. If the robot is far away from the sample then it will drive to a position near the sample (about 1 meter away), detects the sample again, and repeats. If the robot is already at that distance then it drives up to the pickup position (so the sample is 10-20 cm away), detects the sample and repeats. If the robot is then detected to be in the pickup position then the arm is commanded to pick up the sample. If at any point in the process the sample is no longer detected the robot would abort trying to pick up the sample and restart the spiral search pattern.

After the robot attempts to pick up the sample, the robot would try to detect the sample again in order to ensure that it was actually picked up. If the sample was not detected then the robot would assume that it successfully picked up the sample and return to the starting platform. If the sample was detected again then the sample was not successfully picked

up and another pickup would be attempted by repeating the detection, drive and pickup process. If the arm is interrupted or never completes the pickup trajectory after a timeout, such as if it ran into something or could not plan to a given configuration, then the robot will abort the pickup and stow the arm. After stowing the arm the robot will then drive backward a meter and attempt to pick up the sample again. By doing this the robot is able to reposition in order to potentially improve the position for the pickup.

After the sample is detected the laser scanner is shuttered for the entire process of navigating to the sample and picking it up. This is done so that the sample is not accidentally detected by the laser scanner and interpreted to be an obstacle making it impossible to drive to it. This is also known to be safe because the samples are known to not be located near any kind of obstacle. Once the sample pickup is completed or abandoned the laser scanner is unshuttered so that the robot does not run into anything as it continues driving.

Returning to the Start Platform—After the robot successfully picks up the sample it begins to return to the starting platform. To do this the robot drives towards the starting platform while searching for the fiducial markers that were placed on the platform. If the robot does not see the fiducials it begins a spiral search pattern until it sees them. Once the platform is identified the robot drives to a position in front of the platform. The laser scanner is shuttered again so that it does not accidentally see the platform as an obstacle and then drives onto the platform and stops, completing the challenge.

In order to accurately detect the location of the home platform the AprilTags [7] fiducial system was used because they allow for the calculation of the relative position of the camera to the tag in addition to providing a collection of many unique tags that can be used. A simple C++ library[7] is available that allows for efficient extraction of the tag positions from an image. Many of these tags were placed in various orientations on the starting platform such that at least one could be seen from any angle as shown in Figure 6. Because the position of each tag was well know relative to the platform the position of the platform could easily be computed. Each detection of the beacon was then used as an input to a Kalman filter that would estimate the position of the platform. This estimated position was used as a reference frame when generating goals for the robot to drive to when returning to the platform.

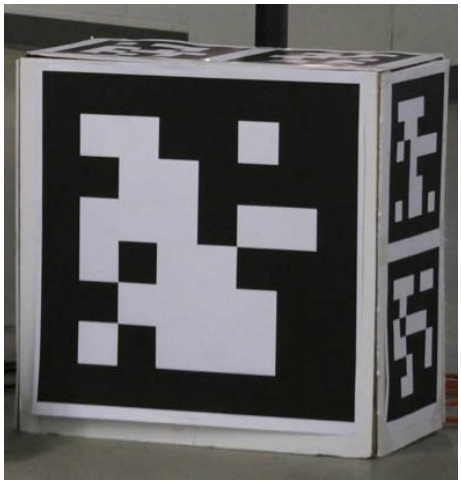

Figure 6. The beacon covered with AprilTags that was placed on the starting platform.

5. FUTURE WORK

Future work will focus on integrating the autonomous behaviors and modes developed on AERO for the SRR competition with other work on teleoperated modes with Oryx 2.0 into a shared control multi-robot system for efficiently exploring unknown terrain. It is not reasonable to assume that autonomous modes of operation would be used extensively in a space environment soon because of the risk they entail, but on the other hand teleoperation and carefully choreographed scripting of robot behaviors will not scale well to multirobot systems. The human-in-the-loop cyber physical system (HiLCPS) modeling framework developed in [10] can help verify and validate the shared control system. We have reproduced the diagram in Figure 7.

The structure of the architecture is split into quadrants related to *where* (cyber and physical) each component resides and *who* (human and robot) contributes the information in each component. The components in the cyber realm operate in an abstract area that could be in the internet (on a cloud platform for example), could be on the robot, or could be distributed across several different areas. The cyber realm is not characterized by physical location but the availability of significant computation power and bandwidth within the realm, where heavy computing and intelligence can be easily implemented. The components in the physical realm are associated with the tangible objects of the system such as the on-board computing of the systems and interfaces the operators utilize.

A second categorization splits the architecture between components that directly utilize mostly human-centric information or robot-centric information. The components base their outputs on the information that is available, so this is a natural dichotomy based on how the control within the system is split between the autonomous agents and the human-driven ones. The human robot interface provides an abstraction for the information transfer between the two realms encompassing information generated by the operators and also automatic contextual information from the other elements. It is separate, but includes information from the operator interface. In reality, these boundaries are fuzzy and not clear-cut defined in a real system, but such a categorization helps with the comprehension of information sources and flow within the system.

Concepts such as trust between agents in the system, the performance of the system, the control effort required within the system, and efficiency of control need to be easily quantified and described within the architecture. In addition to these interface-level metrics (metrics between subsystems within the HiLCPS), metrics internal to the subsystems should be available to encourage system-level optimization and model-based control of these HiLCPS. Complex intricacies in the systems can then be easily evident as opposed to being buried in the details and coupling of the individual subsystems.

The work on the vision system in Section 3 presented in this paper is relevant to the perception engine, where all the relevant sensor data is aggregated from the different platforms. It is the first place that has access to data from multiple sources, so sensor fusion algorithms to confirm localization of robots, samples of interest, or landmarks can be implemented. In

[7] http://people.csail.mit.edu/kaess/apriltags/

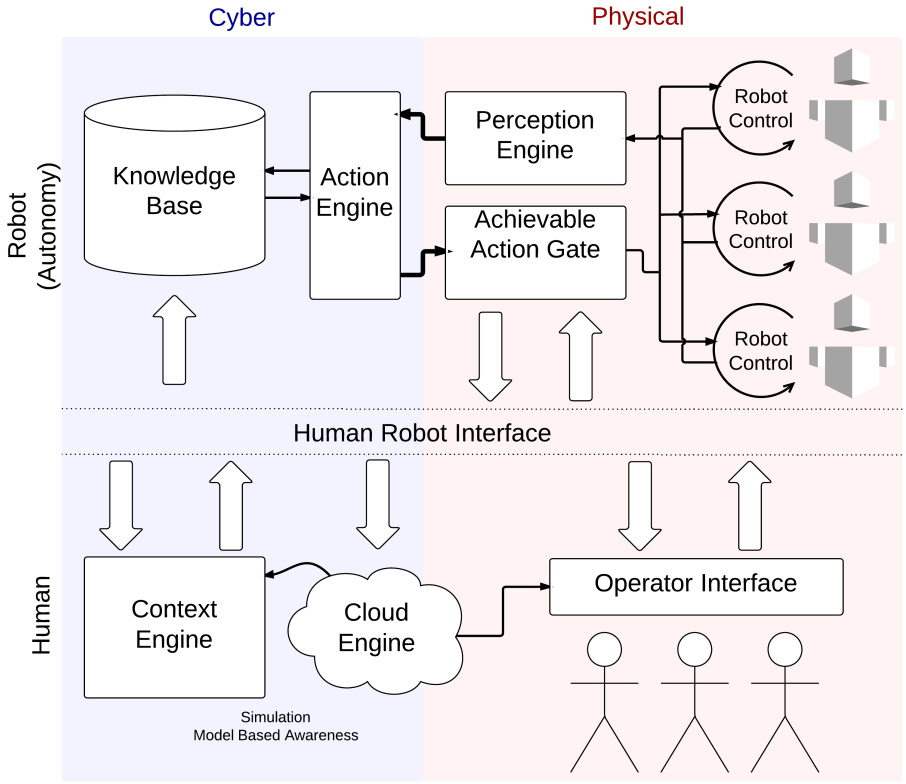

Figure 7. The shared control architecture consisting of the knowledge base, action engine, achievable action gate, perception engine, low-level robot control, human robot interface, context engine, cloud engine, and operator interface. The architecture is split in both cyber and physical domains, and also human and robot domains depending on *where* elements are located and *who* contributes information to each element.

addition, the work on the navigation and planning system in Section 4 is relevant to the action engine, achievable action gate, and low-level robot control. We are also working on integrating the different shared control modalities described in [11]. The more robust autonomy that can be pushed off to the individual robots, results in less operator load and commands that can be at higher more abstract levels. The shared control framework will inevitably change and be modified as we continue to work towards an integrated exploration system across multiple platforms.

Efforts to combine observations from multiple systems are already providing unprecedented results, especially with the collaboration between ground based rovers, the Mars Reconnaissance Orbiter (MRO), and the human operators of each system. For example, work in [12] demonstrates collaboration between orbital maps and rovers for effective long range localization. As these collaborations begin to include more systems, the infrastructure of the shared control framework can provide a guide to enable structured development of collaborations between multiple robots.

6. Conclusion

We have presented our work on AERO for the 2014 NASA SRR competition with the intent of enabling more robust and reliable autonomous operation for sample return rovers. Future work will focus on taking the improvements in autonomy to the platform and applying them to a shared control system of multiple platforms for more efficient exploration of unknown environments. We highlighted improvements made in the training procedures for the object classifiers, leading to much better classification performance, in addition to the addition of new features such as visual odometry for better navigation. Moreover, AERO competed in the competition with a completely redesigned navigation and planning system, closer aligned with ROS standard practices, resulting in more predictable and stable operation. Finally, a new method for taking into account global position information based on virtual reality tags was presented.

Acknowledgments

The authors would like to thank AGCO, KVH, Clearpath Robotics, Harmonic Drive, NVIDIA, Dragon Innovation, Gigavac, and Advanced Circuit sponsored the development of AERO. In addition, the authors want to thank Bond Construction for allowing the testing of rovers in their rock quarry. Finally, the authors would like to acknowledge the Robotics Engineering Program at Worcester Polytechnic Institute and the team that built AERO.

References

[1] J. P. Grotzinger, J. Crisp, A. R. Vasavada, R. C. Anderson, C. J. Baker, R. Barry, D. F. Blake, P. Conrad, K. S. Edgett, B. Ferdowski *et al.*, "Mars science labora-

tory mission and science investigation," *Space science reviews*, vol. 170, no. 1-4, pp. 5–56, 2012.

[2] V. Dimitrov, M. DeDonato, A. Panzica, S. Zutshi, M. Wills, and T. Padir, "Hierarchical navigation architecture and robotic arm controller for a sample return rover," *Systems, Man, and Cybernetics (SMC), 2013 IEEE International Conference on*, 2013.

[3] T. Carlone, J. Anderson, J. Amato, V. Dimitrov, and T. Padir, "Kinematic control of planetary exploration rover over rough terrain," *Systems, Man, and Cybernetics (SMC), 2013 IEEE International Conference on*, 2013.

[4] V. Dimitrov and T. Padir, "A comparative study of teleoperated and autonomous task completion for sample return rover missions," in *Aerospace Conference, 2014 IEEE*, March 2014, pp. 1–6.

[5] J. Amato, J. Anderson, T. Carlone, M. Fagan, K. Stafford, and P. T., "Design and experimental validation of a mobile robot platform for analog planetary exploration," *Proc. IECON 2012: 38th Annual Conf. of the IEEE Industrial Electronics Society*, Montreal, CA, Oct. 25-28, 2012.

[6] Sample return robot challenge. Worcester Polytechnic Institute. [Online]. Available: http://challenge.wpi.edu

[7] E. Olson, "AprilTag: A robust and flexible visual fiducial system," in *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, May 2011, pp. 3400–3407.

[8] A. Howard, "Real-time stereo visual odometry for autonomous ground vehicles," in *Intelligent Robots and Systems, 2008. IROS 2008. IEEE/RSJ International Conference on*, Sept 2008, pp. 3946–3952.

[9] D. Fox, W. Burgard, and S. Thrun, "The dynamic window approach to collision avoidance," *Robotics Automation Magazine, IEEE*, vol. 4, no. 1, pp. 23–33, Mar 1997.

[10] V. Dimitrov and T. Padir, "A shared control architecture for human-in-the-loop robotics applications," in *ROMAN, 2014 IEEE*, Aug 2014.

[11] A. Enes and W. Book, "Blended shared control of zermelo's navigation problem," in *American Control Conference (ACC), 2010*, 2010, pp. 4307–4312.

[12] P. Carle and T. Barfoot, "Global rover localization by matching lidar and orbital 3d maps," in *Robotics and Automation (ICRA), 2010 IEEE International Conference on*, 2010, pp. 881–886.

BIOGRAPHY

Velin Dimitrov *is a Ph.D. Candidate in the Robotics Engineering program at Worcester Polytechnic Institute (WPI) since the fall of 2011. Velin received his Bachelors of Science in Electrical and Computer Engineering from Franklin W. Olin College of Engineering in Needham, MA. As part of his work in the Robotics and Intelligent Vehicle Research Laboratory, Velin has been involved in NSF sponsored research on developing a semiautonomous wheelchair, DARPA funded research working with the ATLAS robot for the DARPA Robotics Challenge, and participated in two NASA RASC-AL Robo-Ops competitions and two NASA Sample Return Robot Centennial Challenges. He has completed internships at both Milara in Medway, MA, and Teledyne Benthos in North Falmouth, MA. Velins areas of interest include human-in-the-loop shared control of robots for of space exploration, disaster response, and assistive applications.*

Mitchell Wills *is an undergraduate student at Worcester Polytechnic Institute (WPI) pursuing a degree in Computer Science and Robotics Engineering. Mitchell is interested in software engineering for robotics applications, robot controls, and embedded systems development. Along with working on numerous robots for classes, Mitchell has participated in the NASA Sample Return Robot Centennial Challenge in 2013 and 2014, and assisted a team competing in the RASC-AL Robo-Ops Competition. He also works as a teaching assistant for WPI's robotics navigation and mapping course and has real world software experience from internships at companies including MITRE, Microsoft, and Google.*

Taşkın Padır *is an Assistant Professor of Robotics Engineering and Electrical and Computer Engineering at Worcester Polytechnic Institute. He received his Ph.D and M.S. degrees in electrical and computer engineering from Purdue University. He holds a B.S in electrical and electronics engineering from the Middle East Technical University in Turkey. PI Padir has significant experience with the design, development and control of robotic systems and intelligent vehicles. He is the director of the Robotics and Intelligent Vehicles Research Laboratory (RIVeR Lab) at WPI. His research interests include human-in-the-loop robotic systems, design of robot control interfaces, cooperating robots, control of redundant robot systems, control of ground vehicles, navigation, path planning, and mapping for autonomous robots. His projects have been sponsored by NSF, DARPA, NASA, AFRL, Draper Laboratory and many industry partners including AgCo, The MathWorks, Solidworks, and National Instruments. Moreover, he is the 2013 Joseph S. Satin Distinguished Fellow in WPI's Electrical Engineering, and he received the Inaugural Rho Beta Epsilon Award for Excellence in Robotics Education in 2010 and 2011 Romeo L. Moruzzi Young Faculty Award for Innovation in Undergraduate Education for his contributions to WPI's unique undergraduate program in robotics engineering. He is currently the PI of two NSF sponsored grants on the design and control of assistive robotics systems and a co-PI on WPI's DARPA Robotics Challenge Track C Team which took the 2nd place in the Virtual Robotics Challenge in June 2013, 7th place in the DRC Trials, and advanced to DRC Finals to be held in June 2015.*