Supervisory Control of a Humanoid Robot in Microgravity for Manipulation Tasks

Logan C. Farrell, Phil Strawser, Kimberly Hambuchen, Will Baker, Julia Badger

Abstract— Teleoperation is the dominant form of dexterous robotic tasks in the field. However, there are many use cases in which direct teleoperation is not feasible such as disaster areas with poor communication as posed in the DARPA Robotics Challenge, or robot operations on spacecraft a large distance from Earth with long communication delays. Presented is a solution that combines the Affordance Template Framework for object interaction with TaskForce for supervisory control in order to accomplish high level task objectives with basic autonomous behavior from the robot. TaskForce, is a new commanding infrastructure that allows for optimal development of task execution, clear feedback to the user to aid in off-nominal situations, and the capability to add autonomous verification and corrective actions. This framework has allowed the robot to take corrective actions before requesting assistance from the user. This framework is demonstrated with Robonaut 2 removing a Cargo Transfer Bag from a simulated logistics resupply vehicle for spaceflight using a single operator command. This was executed with 80% success with no human involvement, and 95% success with limited human interaction. This technology sets the stage to do any number of high level tasks using a similar framework, allowing the robot to accomplish tasks with minimal to no human interaction.

I. Introduction

For future space missions, NASA is currently considering the need for pre-locating mission equipment prior to astronaut arrival as well as maintenance and upkeep of unmanned, but still operational, habitats and spacecraft before and between human visits. During these missions, robots that can interact with the human environment are necessary to perform these construction, maintenance, and upkeep tasks. Robonaut 2 (R2) is designed with a human form factor to enable it to interact with this human environment. Therefore, the environment does not need to be specifically designed for both a special robot manipulator and a human hand [19]. Thus, Robonaut is a critical development platform for these future "care-taker" robots.

Due to the large distances from Earth that future missions will be conducted, the ability to teleoperate robots with latent and low bandwidth connections diminishes. Based on data from the International Space Station (ISS), the number of items and amount of time spent on logistics such as unpacking, stowing, setup, etc., for deep space missions will be substantial [18]. The burden of these logistics tasks can be decreased by the presence of a robot that is able to accomplish them during the dormancy phase before astronauts arrive, allowing the humans to focus more time on science and exploration. Therefore, for NASA, there exists a clear need to develop robotic capabilities to control dexterous manipulation tasks far from Earth with large latencies. Supervisory control can mitigate these time delays while still providing an effective means to command a dexterous robot [22]. While the robot does need a certain level of autonomy for safety and efficacy, it is not unreasonable for the robot to stop an operation when an issue occurs, asking for human clarification or assistance. In these scenarios, it is desirable to provide the robot high-level tasks, and allow it to perform some of its own error mitigation, while also providing a pathway for the robot to "ask for help".

The Robonaut team has experience commanding dexterous manipulation tasks with the R2 unit that resides on the International Space Station (see Figure 1). Examples of manipulation tasks conducted with the R2 on orbit include soft goods manipulation (blankets, wipes), switches and knobs, and tool manipulation such as RFID readers and air flow measurement tools [10]. The lessons learned from these experiments have driven the development presented in this paper.

Figure 1. Robonaut on the Internation Space Station

II. Background

For decades, the robotics research community has attempted to define the appropriate level of autonomy for robot control, ranging on a sliding scale from direct teleoperation to fully autonomous operation. In practice, tasks that require dexterous manipulation have been dominated by teleoperation. Some examples of this include the well-known DaVinci Surgical system[1], improvised explosive device disposal [3], and the Quince robots used after the Fukushima

[1]http://www.davincisurgery.com/

Daiichi disaster [4]. The cognitive power of humans and their ability to quickly make decisions cannot currently be replicated by any autonomous systems technology. In addition, the need for situational awareness for the operator in relatively low latency conditions has been demonstrated through many experiments [1, 2].

Communication between a remote robot and its operator can greatly affect situational awareness, due to bandwidth limitations, latency, and lossy networks. Many current applications for teleoperation of remote robots experience disruptive network scenarios. The recent DARPA Robotics Challenge, inspired by the Fukushima disaster, was a prime example of this: part of the challenge was navigating the poor information link [5, 6, 20]. When these connections become latent or otherwise poor, the role of the human operator must necessarily be reduced. The most common method for accommodating poor communications is to increase autonomous functions, either in the robotic system, in the user interfaces, or both [7, 8]. As autonomous functionalities take on more decision making roles for a task, the control mode transitions from teleoperation to supervisory control.

Many supervisory control interfaces for dexterous robots have been developed and tested in both research and real-world environments. Path planning and obstacle avoidance are common autonomous functions such as in [12]. Other works add features to user interfaces to reduce operator workload [13]. Some groups have attempted to make dexterous manipulators fully autonomous, for example, the DARPA ARM project [11]. Results generally point to fragility in the solution due the complexity of the task.

In this paper, an integrated control methodology that allows supervised control of a dexterous robotic manipulation task is presented and tested in a relevant space environment. To accomplish this supervised manipulation task, three distinct building blocks are integrated which are outlined in Section III. These building blocks are Affordance Templates, object recognition and template placement, and both task execution and supervision using TaskForce. The paper concludes with the results of an experiment of this supervised autonomy in a relevant manipulation task, the removal of a Cargo Transfer Bag (CTB) held in a rack with a restraint similar to those used in resupply vehicles seen in Figure 2.

III. METHODS

To accomplish a supervised dexterous manipulation task, three distinct software components are used in combination. First, in order to understand how the robot can manipulate and interact with objects the Affordance Template (AT) Framework is utilized. Next, a series of supervisory controllers are used to detect, localize, and place these Affordance Templates in the robot's planning scene without human input. Finally, the task is executed using TaskForce, a new commanding infrastructure that allows for easy development of task execution, simple feedback to the user to aid in off-nominal situations, and the capability to add autonomous verification and corrective actions.

A. Affordance Templates

The Affordance Template (AT) framework provides a remote robot operator with a user interface for shared control [9]. The framework consists of templates that describe *affordances* of objects to a robot and exists within a visual environment, RViz[2], for operator interaction. These affordances indicate to the robot how it should or can interact with an object to successfully accomplish a task. Affordances may consist of waypoint locations for end-effectors, compliance gains for manipulators, or forces applied to an object, among other information. The templates in the AT framework define the affordances and provide an interactive control marker [14] that visualizes the object with which the robot should interact. This marker also allows the operator to move the template around in a virtual environment as needed.

The library of affordance templates for R2 currently consists of items such as drills, CTBs, buttons, switches, screwdrivers, and ISS handrails. Affordances described currently consist of waypoint locations for R2's hands, and grasp positions for its end-effectors. These affordances define a discrete trajectory in space to reach the object, and how the robot should grasp the object with its highly dexterous hand.

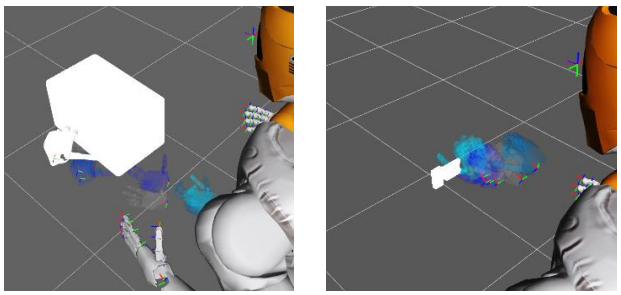
Each template may have several sets of waypoints/grasps defined for it, and in the initial framework the operator is the arbiter of which set of waypoints/grasps is appropriate for the robot to interact with the object given the current understanding of the robot's environment. For example, Figure 2 shows multiple sets of waypoints to grab the CTB's handles: one right handed set of waypoints to grab the small handle, and a left handed set of waypoints to grab the larger handle. This enables the operator to decide how a robot with multiple end-effectors should manipulate an object. An example of these affordances can be seen in Figure 3. The goal is to replace this operator task with autonomous sensing and decision making proposed in Sections III.B and III.C.

In order for the template to be useful, the template must be registered to sensory data returned from the robot. While past operations using Affordance Templates have relied on the human to register the template to sensory data, current testing uses autonomous template matching to register to data. If for any reason the human sees that template placement is incorrect, the operator can interact with the template to adjust placement accordingly. Once the template is placed, motion is planned along waypoints and is visually presented to the operator. R2 currently uses MoveIt![3], and the OMPL[4] library to plan trajectories for waypoints. These plans are then sent to the robot to execute the motions.

[2] http://wiki.ros.org/rviz
[3] http://moveit.ros.org/
[4] http://ompl.kavrakilab.org/

(a) 2 (b)

Figure 2. *(a)* Cargo Transfer Bag (CTB), *(b)* CTB with restraint in the rack



(a) (b)

Figure 3. *(a)* CTB left handle and front handle affordance, *(b)* Restraint affordances

The two objects manipulated during this test were a restraint button to simulate a restraint on a spacecraft and can be seen in Figure (3.b). The second object is the CTB seen in Figure (2.a) which is a standard bag used in spaceflight for storing cargo. The affordances for these bags can be seen as the slightly transparent hands, showing the robot the waypoints to manipulate the object in Figure 3.

B. Autonomous Template Placement

To decrease the burden on the operator due to latency, a method for visual detection of objects to allow their corresponding Affordance Templates to be placed in the planning scene was implemented. Three separate algorithms were used for this phase of operations: a method to localize an object using R2's camera system, a process that looks for that pose and creates a new, filtered pose at the objects center, and finally, a process looking for the final filtered pose and adding and removing Affordance Template objects.

Visual detection, recognition, and pose estimation of objects of interest is an area of substantial ongoing research in the community [15-16], however, these are outside of the scope of this work. Instead, an architecture to allow any object detection algorithm to be integrated was developed. For these experiments, fiducial markers were used to localize the objects in question using the Robot Operating System *(*ROS) package AR Track Alvar[5]. This package will return a position and orientation transform of the marker as detected through a monocular camera. The physical example of this can be seen in Figure 4. By separating the object detection and placement algorithms, the object localization method can be replaced by any that will give a transform for the object as technology advances.

Once the marker pose is known, this must be translated into useful information for the robot. The first step of this automatic conversion is a continuously monitoring method, using the Transform Supervisor. The Tansform Supervisor monitors for any poses that are associated with objects based on a custom object description dictionary yaml file. If a pose is found, it will do a static transform to convert the marker pose to the objects center. Next, the Transform Supervisor will look for these known object poses as dictated by the object dictionary and filter them to take out noise in the positions, as the fiducial marker detection can sometimes return errant poses. The filter chosen for this experiment eliminates the top 25% and bottom 25% of transforms as determined by position from the positional average of the middle 50%. The average for just the middle 50% is then calculated, giving the filtered position for the object. Using the SLERP method for spherical linear interpolation between two quaternions, the interpolated rotation is found using the middle 50% of points [17]. This method results in a stable center pose for the object.

The final step in the object placement process is the Affordance Template Supervisor, or AT Supervisor. This supervisor method will look for the filtered pose of a known object based on the object dictionary, and if one is found, it will add that Affordance Template to the planning scene of the robot. It will continue to monitor the existence of the pose, and if the pose goes stale for longer than a user-specified time, it will then remove the object from the scene. This final step creates a scene of known objects and collisions for the robot, as well as a method for interacting with the known objects via Affordance Templates.

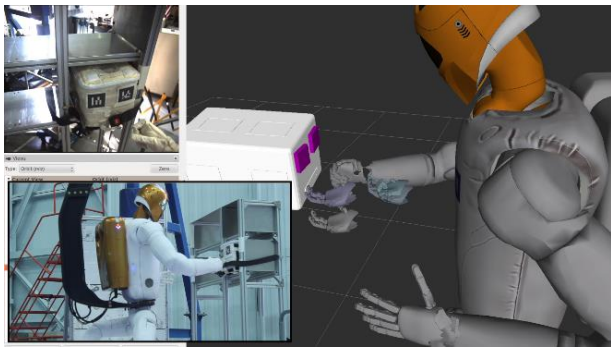


Figure 4. Visual identification and object placement using fiducial markers and visual supervisors

[5] https://github.com/sniekum/ar_track_alvar

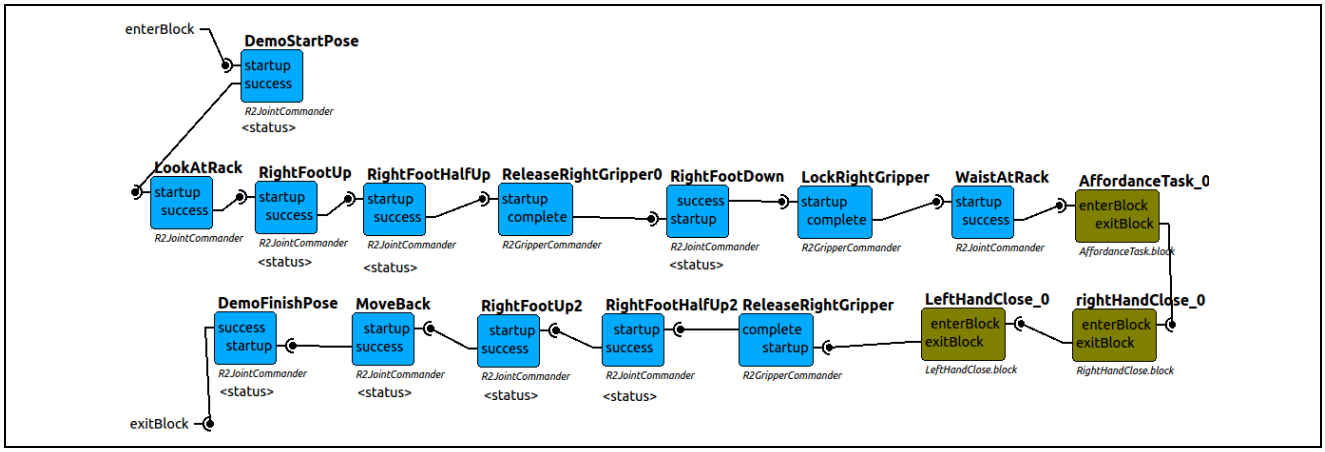


Figure 5. TaskForce overall experiment execution block

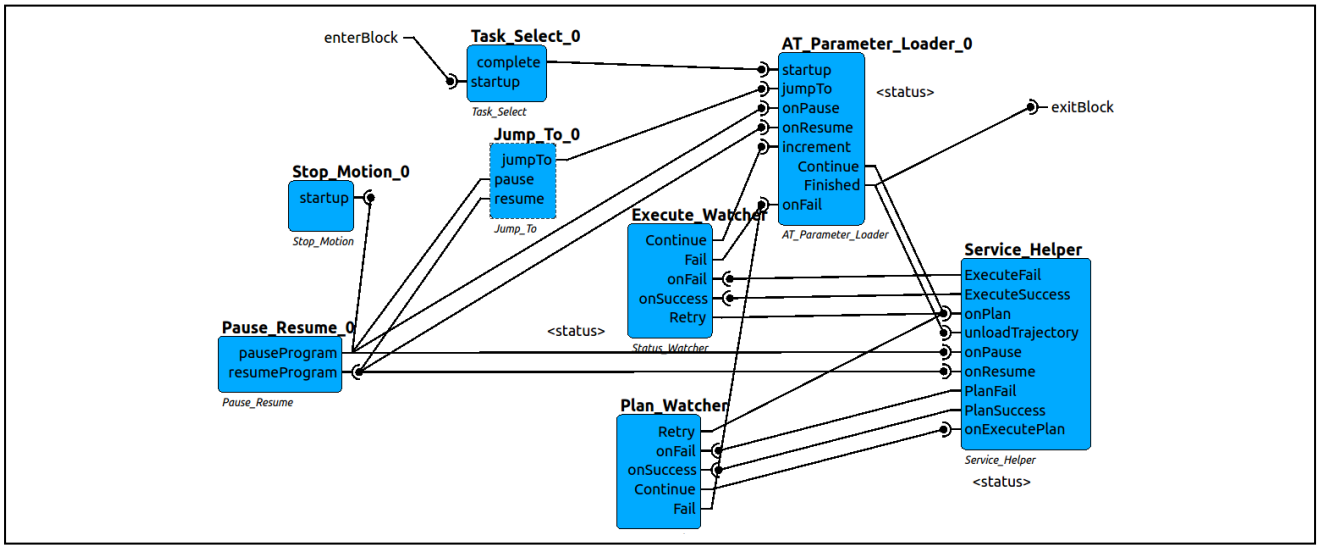


Figure 6. TaskForce Affordance Task Block for supervised execution of an Affordance Task

This approach also allows for any individual piece of the algorithm to be replaced in the future by more advanced work from the community. Each only looks for its individual transform, with no knowledge of the pieces around it, allowing more advanced visualization to be easily adopted, different filtering methods to be attempted, and different affordance types to be used.

C. Task Execution via TaskForce

TaskForce is an Integrated Development Environment (IDE) for developing software tasks using a component-based programming model [21]. Users define individual code elements, called Tasks, which can then be composed together to form a Block. A Block can be composed not only of Tasks, but also other Blocks as well. With this composition pattern, an arbitrarily complex hierarchy of Tasks can be defined. Tasks interact with one another using a system of events. Each Task can define an arbitrary number of named events which it can emit, and then other Tasks can subscribe to those events. When a Task subscribes to an event, it declares an internal callback method to be called when that event has been received. This implicit invocation architecture is very beneficial in that it reduces any sort of coupling between Tasks.

Using the Task Editor, Blocks are constructed using a visual programming interface. The user is presented with a canvas which represents the definition of the Block. Then, the user can drag-and-drop Tasks and Blocks from a library of existing Blocks and Tasks that have been previously created. Once the canvas is populated, the connection tool is used to draw subscriptions between two elements. There are several use-cases in which Blocks will be constructed using a very linear flow: task “A” triggers task “B” which then triggers task “C”, and any following tasks if they are required. However, an arbitrary set of subscriptions can be defined: task “X” may call task “Y” which will return to “X” which will then call “Z” and “W.”

This framework allows rapid development of arbitrarily complex series of executions, insight into the progress and state of the execution, and the ability for the user to interact with the execution to modify or help the robot. It enables the developer to create a linear set of functions to do a task if required, or to create any number of task loops, supervisory checks, retry steps, and fail operations. Examples of both of these can be seen in the experimental task performed by the robot. Figure 5 shows the block diagram for the overall execution of the task. The linear series of blocks moved the

robot through a set of poses to move into position in front of the objects to be manipulated.

During this execution, it reaches the "Affordance Task." This block can be seen in Figure 6 and is nonlinear in nature. The block executes an Affordance Task, which is an *a priori* series of waypoints for one or many Affordance Templates in the scene to accomplish a high level task. As part of this affordance task, other actions may be put in such as pauses, commands to look at different positions, and different checks to perform before proceeding.

The Affordance Task block uses the specified dictionary of tasks corresponding to the requested high level task to be performed. The robot begins to execute the waypoints, or additional commands sequentially. As seen in Figure 6, there are other processes watching this commanding flow, giving authority to the robot to retry and correct different actions. The "Plan_Watcher" task is monitoring the trajectory plan status as new plans are made. If a plan is not found, it forces the process to go back and replan. If no plans are found in 5 attempts, it pauses the block and alerts the user there is an error. Similarly, the "Execute_Watcher" block monitors executions, and forces retries if the execution is not successful before continuing. These are basic building blocks to allow the robot to take simple corrective actions to accomplish tasks before alerting the operator of any issues or failures.

This very simple monitoring method allows levels of autonomy previously unrealized in R2's dexterous manipulation capability. In addition, further autonomy can be built in with this framework. For example, the monitoring of an object in the planning frame could indicate whether that object has been successfully manipulated. If the object is supposed to be manipulated and moved, and the visual system does not recognize the movement, the robot can return to the previous step to re-attempt. If this attempt fails a second time, the operator will be notified. Then, the operator can modify the position or understanding of the object based on the failed execution, and resume the high level task. This was applied directly in the experiment performed with removal of the restraint before continuing to remove the CTB.

This can be easily extrapolated to further types of verification and autonomous corrective behavior based on the situation with any number of supervisor methods watching the status of the execution. While the execution is a simple list of waypoints, these could signal many different types of verification. For instance, one technology in under development is force monitoring of the arm joint, correlated to execution of tasks. For a given waypoint to be successful, a range of forces are expected and if the force profile does not match, a force supervisor could re-attempt that waypoint. This could be extended to visual checks, or any number of other verification methods to build up the autonomous functionality while remaining a generic commanding tool.

IV. Test Setup

To test the supervisory control method presented, a relevant task was outlined involving logistics of an unmanned module. Cargo is stored in visiting vehicles on spacecraft in large racks, held in by various buckle systems. The crew time needed to do these unpacking tasks could be better spent performing research and science on the spacecraft, which makes them a desirable task for the helper robot to accomplish before the crew arrives. In addition, due to the large communication delays present in deep space flight, it is necessary to reduce the overall number of interactions the operator must have with the robot to accomplish the task.

To test R2 in a relevant microgravity environment, the robot was suspended in the Active Response Gravity Offload System (ARGOS) which offloads the robot's center of mass and follows in the three dimensional environment to simulate a microgravity effects on the legs. The joints still have local weight, but the response of the robot to the leg torques is similar to that in microgravity. In this space there are panels with handrails similar to that found on the space station to allow for climbing testing of the robot. On one end of the panels, a rack is placed with shelves approximately 1m from the floor that can hold CTBs with a restraint across them.

The high level task dictated to the robot is to remove the CTB from the rack. This task is the initial step in robotic logistics management, something that astronauts currently do on the ISS. This involves a series of joint moves that brings the robot in front of the rack followed by an Affordance Task using TaskForce as defined in section III.C. This process identifies the restraint and the bag using the fiducial markers and puts them into the robots known workspace. Then, the Affordance Task works through the necessary steps to release the buckle and remove the bag with checks at each step to verify correct execution and whether the restraint was fully released. Finally, the robot removes the bag and moves to its next position. This entire process lasts fewer than 5 minutes. The experiment was repeated 20 times to determine consistency and repeatability.

V. Results

The methods described were applied to the test setup discussed in Section IV. TaskForce is implemented to move the robot towards the goal, and then the Affordance Task is executed using an appropriate affordance task dictionary. The final solution was a single operator command to execute the high level task of "Remove the CTB from the rack." Through experimentation, the robot was able to complete the task in 85% of trails with no additional human interaction. 10% of trials the robot encountered an issue and stopped to ask for user input. In both cases, the localization of the restraint was slightly lower than the restraint itself. Using the Affordance Template framework discussed, the operator was able to quickly adjust the template based on viewing the robots previous execution attempt. Then, in the TaskForce interface, the operation was resumed and the task executed successfully. The human intervention lasted less than one minute. In 5% of cases, the robot experienced a fault causing a software e-stop that required the faults to be cleared and the robot reset, resulting in a failed attempt.

VI. Conclusion and Future Work

A new framework for supervisory control of remote, dexterous robots was presented. This framework integrates Affordance Templates, which provide a visual definition for the robot to interact with objects, autonomous template placement, and TaskForce, a tool for task development and execution. Using this framework, a high level task involving many dexterous manipulations was accomplished. This framework allowed the robot to monitor its own processes and attempts to retry failed steps before alerting the user of issues. However, when issues arose, in all but one case, the operator was able to quickly rectify the situation due to the insights available from the tools. This framework has laid the foundation for building large levels of supervised autonomy that can be useful in long distance, long duration space missions.

The framework presented can be used for the development and control of multiple robotic tasks, which inevitably leads to a library of tasks available for supervisory control, for example, removal of packed objects, stowing objects, and cleaning. Currently, development of autonomous climbing in a microgravity environment for R2 is ongoing. This climbing capability will allow the robot to move from task to task through the station, allowing many dexterous tasks to be strung together in a single, overall operation. For example, the robot could unstow a CTB, climb across the lab to place it into the required drawer, and return to access the next CTB using a series of Affordance Tasks. In addition, this framework is currently being coupled with a procedures system used by astronauts to enable more efficient human interaction with a robot and to eventually port human procedures into robotic tasks.

References

[1] Hirzinger, G. (1994). "ROTEX- The first space robot technology experiment." *In Experimental Robotics III* (pp. 579-598). Springer Berlin Heidelberg.

[2] Imaida, T., Tokokohji, Y., Doi, T., *et al.* "Ground-Space Bilateral Teleoperation of ETS-VII Robot Arm by Direct Bilateral Coupling Under 7-s Time Delay Condition." *IEEE Trans. On Robotics and Automation*, 20.3, June 2004.

[3] Elliott, Linda R., *et al*. "Robotic telepresence: Perception, performance, and user experience". No. ARL-TR-5928. ARMY RESEARCH LAB ABERDEEN PROVING GROUND MD HUMAN RESEARCH AND ENGINEERING DIRECTORATE, 2012.

[4] Nagatani, Keiji, et al. "Emergency response to the nuclear accident at the Fukushima Daiichi Nuclear Power Plants using mobile rescue robots." *Journal of Field Robotics 30.1* (2013): 44-63.

[5] Johnson, Matthew, et al. "Team IHMC's lessons learned from the DARPA Robotics Challenge trials." *Journal of Field Robotics 32.2* (2015): 192-208.

[6] Kohlbrecher, Stefan, et al. "Human-robot Teaming for Rescue Missions: Team ViGIR's Approach to the 2013 DARPA Robotics Challenge Trials." Journal of Field Robotics 32.3 (2015): 352-377.

[7] McGill, Stephen, Seung-Joon Yi, and Daniel D. Lee. "Team THOR's adaptive autonomy for disaster response humanoids." *Humanoid Robots (Humanoids)*, 2015 IEEE-RAS 15th International Conference on. IEEE, 2015.

[8] Karumanchi, Sisir, et al. "Team RoboSimian: Semi-autonomous Mobile Manipulation at the 2015 DARPA Robotics Challenge Finals." *Journal of Field Robotics* (2016).

[9] Hart, Stephen, Paul Dinh, and Kimberly Hambuchen. "The affordance template ROS package for robot task programming." *Robotics and Automation (ICRA)*, 2015 IEEE International Conference on. IEEE, 2015.

[10] Ahlstrom, Thomas, et al. "Robonaut 2 on the International Space Station: Status update and preparations for IVA mobility." *AIAA SPACE 2013 Conference and Exposition. 2013*.

[11] Hudson, Nicolas, et al. "Model-based autonomous system for performing dexterous, human-level manipulation tasks." *Autonomous Robots* 36.1-2 (2014): 31-49.

[12] Mainprice, Jim, et al. "From autonomy to cooperative traded control of humanoid manipulation tasks with unreliable communication: System design and lessons learned." I*ntelligent Robots and Systems (IROS 2014)*, 2014 IEEE/RSJ International Conference on. IEEE, 2014.

[13] García, J. C., et al. "User Interface Oriented to the Specification of Underwater Robotic Interventions." *Journal of Maritime Research 8.2* (2014): 47-62.

[14] D. Gossow, A. Leeper, D. Hershberger, and M. T. Ciocarlie, "Interactive markers: 3-D user interfaces for ros applications [ros topics]," *IEEE Robotics & Automation Magazine*, vol. 18, no. 4, pp. 14–15, 2011

[15] R. B. Rusu and S. Cousins, "3D is here: Point Cloud Library (PCL)," *2011 IEEE International Conference on Robotics and Automation*, Shanghai, 2011, pp. 1-4.

[16] M. Quigley et al., "High-accuracy 3D sensing for mobile manipulation: Improving object detection and door opening," *2009 IEEE International Conference on Robotics and Automation*, Kobe, 2009, pp. 2816-2822.

[17] N. Dantam and M. Stilman, "Spherical parabolic blends for robot workspace trajectories," *2014 IEEE/RSJ International Conference on Intelligent Robots and Systems*, Chicago, IL, 2014, pp. 3624-3629.

[18] Lopez, Pedro, Jr., Bryan Mattfield, Chel Stromgren, and Kandycee Goodliff. *Logistics Needs for Potential Deep Space Mission Scenarios Post Asteroid Redirect Crewed Mission*. Proc. of 2015 IEEE Aerospace Conference, Big Sky, MT. 07 Mar. 2015. Web.

[19] M. A. Diftler *et al*., "Robonaut 2 - The first humanoid robot in space," *2011 IEEE International Conference on Robotics and Automation*, Shanghai, 2011, pp. 2178-2183.

[20] H. A. Yanco, A. Norton, W. Ober, D. Shane, A. Skinner, and J. M. Vice, "Analysis of Human-robot Interaction at the DARPA Robotics Challenge Trials.," *J. Field Robotics*, vol. 32, no. 3, pp. 420–444, 2015.

[21] P. Strawser, L. C. Farrell, K. Hambuchen, M. Goza, S. Azimi, J. Badger, "TaskForce: A Task Design and Execution Framework" *2017 IEEE/RSJ international conference on intelligent robots and systems*, Vancouver, 2017, submitted for publication

[22] Sheridan, Thomas B. *Telerobotics, Automation, and Human Supervisory Control*. Cambridge: MIT, 1992.