SUPERVISING A HUMANOID ROBOT ACROSS TIME DELAY

Robert Burridge[1] and Seth Gee[2]

[1]*TRACLabs, Inc., Houston, TX, USA, E-mail: burridge@traclabs.com*

[2]*TRACLabs, Inc., Houston, TX, USA, E-mail: seth@traclabs.com*

ABSTRACT

The Predictive Interactive Graphical Interface (PIGI) was developed at NASA/JSC to provide supervisors with better situational awareness when commanding mobile robots across time delay. By simulating the expected outcome of queues of discrete commands already sent to the robot, the supervisor is able to explore, plan, and submit future commands, thus reducing robot idle time. In the work presented here, PIGI is extended in several ways: to work with a dual-armed robot, to model internal state variables of the robot and the state of the environment, and to provide the ability to browse historical data. These features enable the supervisor to remain an active participant in the planning and execution of tasks, despite a five-to-ten-second time delay. A comparison is made between PIGI for commanding NASA's R2 humanoid and the Prophesy system for commanding Boston Dynamic's Atlas humanoid.

1 INTRODUCTION

Recent advances in robotic systems components and capability make it reasonable to expect that robots will play an important part in any future missions beyond low earth orbit. In particular, dexterous humanoid robots, such as NASA's Robonaut-2 and Valkyrie, show that mechanisms already exist that are capable of complex manipulation. However, the state of the art in control systems has not yet reached a level of maturity that would allow such robots to be fully autonomous. For now, human expertise is still sometimes needed to make decisions about sensor interpretation and action planning.

The extent to which a remote supervisor may participate in robot operations is governed by the length of the time delay between robot and supervisor. When the round-trip time is short (< 2 seconds), the supervisor may act as a tele-operator, commanding the robot directly with bilateral master-slave control methods. On the other hand, time delays on the order of tens of minutes or longer (such as between earth and mars) preclude any reasonable interaction between operator and robot, and require the robot to be nearly autonomous – perhaps receiving a new plan every few hours or day, and simply stopping action if a problem arises. The work in this paper addresses the intermediate realm of time delays up to tens of seconds, where a remote supervisor may still be an active participant in exploration activity, but certain closed-loop skills must reside on the robot.

2 PIGI AT NASA/JSC

The Predictive Interactive Graphical Interface (PIGI) was initially developed at NASA/JSC as part of the ETDP Multi-Center Cockpit [1]. It was extended as part of the RAPID Workbench [2] and used to control five different NASA robots in four different states (AZ, CA, TX, and WA) in various analog and Desert RATS field tests – all from a supervisor cockpit in Houston [3]. The round-trip time delay in these field tests ranged from 10 to 100 seconds. To distinguish the original version of PIGI developed at NASA/JSC from the version developed at TRACLabs that is described here, we will refer to the former as "JSC-PIGI" and the latter as "TL-PIGI". The place where the robot's supervisor runs the Operator Interface is the "Cockpit". There is no significant latency among processes within the Cockpit or on-board the robot, but there is a multi-second time delay between the two. For historical purposes, we refer to this connection as the "Long Haul" (LH) link.

In this section we describe the primary components of PIGI. For a more detailed account, see [3].

2.1 Fundamental Command Unit

The time delay on LH makes it inadvisable to send a stream of essentially continuous motion commands from the Cockpit to the robot (as with bilateral control), because the supervisor would not be able to react to sensor data in a timely manner. Instead, discrete commands (with possibly continuous parameters) are sent, such as DriveTo(X, Y, Theta). The robot executive parses these into local closed-loop skills on-board the robot. Although most commands used in the NASA field tests were motion commands, other commands were occasionally added to the command queues, such as a SCOUT sensor sweep at a waypoint or triggering a scripted action by the R2 upper body on Centaur-2.

The fundamental feature of PIGI is its management of queues of commands. There are three basic queues for commands kept on the robot: PENDING, ACTIVE, and COMPLETED. When a command arrives at the robot across LH, it is placed on the tail of the PENDING queue. When appropriate resources become available for the task at the head of the PENDING queue, it is removed and placed on the ACTIVE queue. This is the cue for the robot to begin executing that task. When execution is finished (with either success or failure), it is removed from the ACTIVE queue and placed on the COMPLETED queue. It is possible for a command to be "canceled" directly from PENDING to COMPLETED, depending on decisions made by the on-board robot executive.

In the Cockpit, a module called the "Robot Command Server" (RCS) keeps a duplicate set of command queues, updated by telemetry from the robot. Additionally, there is a fourth queue called SENT, which contains commands that have been committed and sent to the robot, but have not been confirmed by return telemetry yet (and thus are not yet PENDING).

2.2 Telemetry

For JSC-PIGI, telemetry from the robot to the Cockpit across LH consists of three primary components: Robot state, including 3-DOF robot location and a few important state variables, such as battery level, sensor operational status, and e-stop status; video stream; and the state of the command queues.

If the primary functionality of the robot is passive exploration – sensors and mobility, but no manipulation – then it is a "mobile eye". In this mode, the supervisor uses the (delayed) video stream and other sensor data to decide where to send the robot next. Most of the robots in NASA's field tests of PIGI operated in this mode.

2.3 Predictor

The Predictor receives the latest robot location from telemetry and the latest queue of committed robot motion commands (SENT, PENDING, and ACTIVE together) from RCS. It generates a prediction of the trajectory the robot will take and the final state of the robot when all the commands are completed. These predictions are updated every time new telemetry is received regarding the location of the robot and/or the command queue.

In order to model the behavior of the robot quickly enough, the Predictor uses a low-fidelity analytic model that generates entire trajectories geometrically.

2.4 Explorer

The Explorer is a module that allows the supervisor to assess possible next commands for the robot. It can start with any initial robot location and queue of commands and will predict the trajectory of the robot. By connecting this directly to the output of the Predictor, the operator is able to explore possible new commands to add to the queue of commands that have already been committed. Because the Predictor is constantly updating the likely final state based on the latest telemetry and the SENT command queue, the operator is immediately aware of any situation that may affect the behavior of the robot. In JSC-PIGI, the Explorer and Predictor use the same underlying software to model trajectories.

2.5 Visualization

The Operator Interface provides the Supervisor with information about the latest telemetry, predicted trajectories for commands already committed to the robot, expected final committed state, and potential new trajectories for commands being explored. Current robot location and trajectories are overlaid on a "birds-eye-view" image of the region being explored.

Trajectories are discretized into strings of X-Y-Theta states separated by 50cm, and represented by a series of arrows, which we call "breadcrumbs".

In Figure 1, we see the supervisor's view when commanding NASA/JSC's Space Exploration Vehicle (SEV) in JSC's Rockyard. A few minutes prior to capturing this image, the robot was sitting on the road (white vertical stripe) near the silo, facing the bottom of the image. The supervisor used the Explorer to plot a trajectory (white breadcrumbs) to a new target location (blue arrow). After observing the intended trajectory, the supervisor concluded that it does not traverse any known obstacles, so the command was committed and (after a delay) the robot started moving. The image in Figure 1 was captured about halfway through the drive. The robot icon represents the latest telemetry, and it can be seen that the robot has drifted left of its intended course. Given this information, the Predictor plots a course to the goal (orange breadcrumbs) that represents the path the robot is expected to take to achieve the goal location, given its true current location.

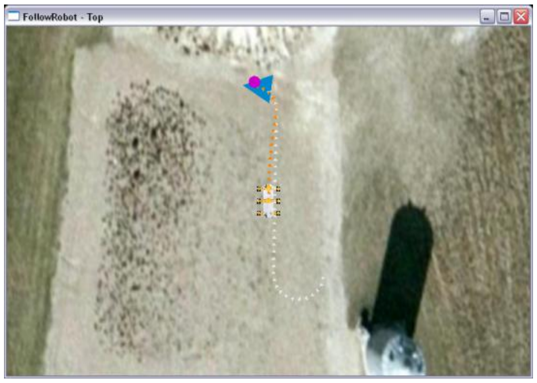


Figure 1:SEV in JSC's Rockyard

2.6 Results and Limitations

JSC-PIGI was successful for what it was designed to do. It enabled an supervisor to participate in the exploration of unknown terrain, making decisions about where the robot should go. Using the Explorer, the operator was able to stay "one step ahead" of the time delay, exploring options for the next motion and committing new commands with enough lead time that the queue on the robot side of the time delay was rarely empty, and thus keeping robot idle time to a minimum.

There are several limitations to JSC-PIGI. First, it is only designed for modeling trajectories of a mobile base on roughly planar terrain. Although this worked well for "mobile eyes", it fell short when manipulation of the environment was desired, as with Centaur-2, and did not take advantage of the unique capabilities of the robots, such as the incredible dexterity of ATHLETE's legs. Second, it does not model any internal state of the robots, even when such state is available in the telemetry. Thus, for example, the predicted response to a drive command would be the same regardless of whether the control system was in drive mode. Finally, it does not model the environment. Again, this is reasonable for "mobile eyes", but is insufficient if we desire the robot to manipulate the environment: collecting samples, interacting with a device, or assembling a structure. The next section describes extensions to PIGI designed to address these limitations.

3 PIGI FOR MANIPULATION

With the development of Robonaut-2, particularly when mounted on a mobile base or its ISS climbing legs, it became clear that PIGI needed to be extended for manipulation. Unfortunately, the "breadcrumb" approach for visualizing trajectories was not practical for so many degrees of freedom (19 just in the torso, arms, and head). Furthermore, it was not clear how to represent the various internal states of the robot, nor the state of the environment. To solve these issues, the notion of a breadcrumb was expanded into a "snapshot", which is a complete record of all state values of robot and environment tied to a particular point in time. The snapshot is the basic building block for communication among the components of a new version of PIGI called TL-PIGI developed by TRACLabs under a NASA grant. Snapshots whose timestamps are sequential may be bundled together into a data structure called a SnapshotSeq.

In this section, we describe the components of TL-PIGI and how they differ from JSC-PIGI to accommodate the new functionality.

3.1 Telemetry

The telemetry coming back from the robot consists of a video stream, snapshots, and queue state messages, all sent at a low frequency – about 2 Hz. Each snapshot contains all robot joint state values, robot location (if mobile), and environment data from sensor interpretation on-board the robot. The low update rate gives the simulations in the Cockpit time to run before new values arrive.

3.2 Visualization

The latest state of the robot reported by the telemetry (CurState) is rendered as a white robot within the 3D visualization window. The location of objects and the state of the environment are all also reconstructed based on the latest telemetry. In Figure 2, the current state of R2 and the state of its Taskboard are shown in the 3D visualization window.

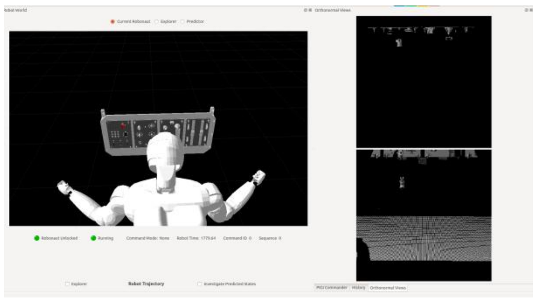
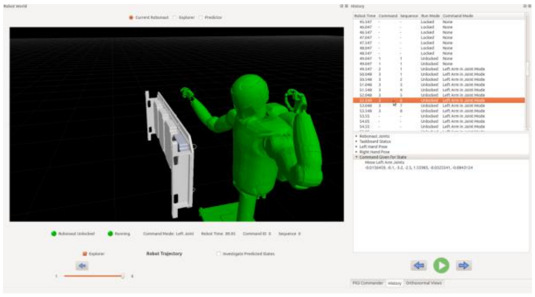


Figure 2: OI showing R2.

3.3 Historical Playback

Every telemetry snapshot that arrives is added to the History SnapshotSeq. The Operator Interface allows the supervisor to select intervals of time (i.e., subsequences of the History SnapshotSeq) and either play them back in real-time, step through them, or use a slider to quickly navigate through them. Robot and environment states are shown in the 3D visualization window. Internal robot state values are displayed in a panel below the 3D window, either as green/red radio buttons (for binary states) or as text boxes (for enumerated states). Thus, the operator can see changes in all modeled states at different times in the SnapshotSeq. To the right of the 3D window, a list of snapshots associates each timestamp with the active command at that time. Historical playback robot states are rendered in green to distinguish them from CurState. In Figure 3, the historical robot is shown in the 3D window, while the SnapshotSeq is listed in the panel on the right.



History

Robot Time	Command	Sequence	Run Mode	Command Mode
45.547	-	-	Locked	None
46.047	-	-	Locked	None
46.547	-	-	Locked	None
47.047	-	-	Locked	None
47.547	-	-	Locked	None
48.047	-	-	Locked	None
48.547	-	-	Locked	None
49.047	1	1	Unlocked	None
49.047	1	1	Unlocked	None
49.547	2	1	Unlocked	Left Arm in Joint Mode
50.048	3	1	Unlocked	Left Arm in Joint Mode
50.548	3	2	Unlocked	Left Arm in Joint Mode
51.048	3	3	Unlocked	Left Arm in Joint Mode
51.548	3	4	Unlocked	Left Arm in Joint Mode
52.048	3	5	Unlocked	Left Arm in Joint Mode
52.548	3	6	Unlocked	Left Arm in Joint Mode
53.048	3	7	Unlocked	Left Arm in Joint Mode
53.548	3	8	Unlocked	Left Arm in Joint Mode
53.55	-	-	Unlocked	Left Arm in Joint Mode
54.05	-	-	Unlocked	Left Arm in Joint Mode
54.55	-	-	Unlocked	Left Arm in Joint Mode

- Robonaut Joints
- Taskboard Status
- Left Hand Pose
- Right Hand Pose
- Command Given For State

Move Left Arm Joints
-0.0156459, -0.1, -3.2, -2.5, 1.55985, -0.0323541, -0.0843124

Figure 3: Historical playback.

3.4 Predictor

It was hoped that a powerful off-the-shelf simulation engine, such as Gazebo [4], could be used to model the robot and its environment in both the Predictor and Explorer. However, the Predictor needs to be able to predict up to tens of seconds of robot behavior within half a second, requiring a speedup on the order of 100x. Unfortunately, Gazebo was only capable of about 2-3x real-time, even when its graphical display was disabled. As a result, a custom state machine was created that was able to quickly predict the motion of the robot and the changes in internal state that result from commands, albeit at lower fidelity. The complexity of such state machines can grow quickly with the size of the command set and with complicated robot/environment interactions. This potential issue is managed by keeping the command set small, making a closed-universe assumption, using geometric interpolation for arm trajectories, and using symbolic approximations for questions of contact, such as grasping.

Every time new telemetry arrives, the Predictor models the expected behavior from commands on the SENT, ACTIVE, and PENDING queues. A 2Hz SnapshotSeq is created for the robot/world state moving forward in time from CurState to the final committed state (FinComState), which is the expected state of the robot after completing all the commands that have been sent to it.

The operator may investigate the Predictor SnapshotSeq in the same manner as the History SnapshotSeq, except that the robot is rendered in blue. This sequence of states provides the operator with the best available idea of how the robot will react to the commands that have been sent, and is constantly updating as new telemetry comes in. If desired, the operator may temporarily "unhook" the sequence from the updates, allowing more careful scrutiny without the 2Hz updates. In Figure 4, the Predicted FinComState is shown alongside CurState for a single command to move the arm to the Taskboard.

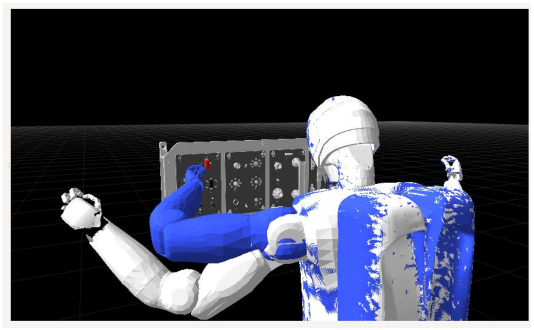


Figure 4: CurState and FinComState.

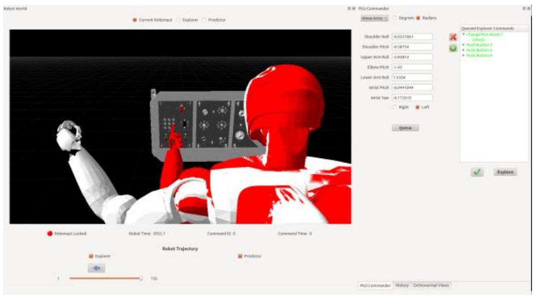
3.5 Explorer

The Explorer enables the operator to try out new commands to add to the end of the SENT queue. In order for this to be effective, the operator must have a clear understanding of what state things will be in at the start of any additional command. Fortunately, this is precisely what the Predictor is producing in FinComState. Thus, FinComState is automatically passed from the Predictor to the Explorer, where it is constantly ready as the initial state of that module.

Using the Operator Interface, the operator may choose one or more commands to try, such as setting a target location for the end effector. These commands get sent to the Explorer. The Explorer uses FinComState as the initial state of the system and simulates the behavior of the robot in response to the queue of new commands. Once again, a SnapshotSeq is created that can be investigated by the operator. For the Explorer, the robot is rendered in red. If the operator is not satisfied with the predicted sequence of events, the explored commands may be deleted, modified, or rearranged. When the new commands are satisfactory, the operator commits them to the SENT queue. From that time forward, these commands will be simulated within the Predictor at the end of the committed queue of commands.

Although it wasn't fast enough to be used in the Predictor due to the required update rate, the higher-fidelity Gazebo model was used in the Explorer.

In Figure 5, the supervisor is trying out a command to place the index finger next to a button on the Taskboard.



Shoulder Roll -0.0337801
Shoulder Pitch -0.58734
Upper Arm Roll -2.83813
Elbow Pitch -1.43
Lower Arm Roll 1.5326
Wrist Pitch -0.0441944
Wrist Yaw -0.172619
Right [✓ Left]
Robot Time: 3052.1
Command ID: 0
Command Time: 0

Figure 5: Exploring a new command.

3.6 Demonstration Platforms

Three platforms were used for demonstrations of TL-PIGI: The TRACBot physical robot, the TRACBot simulation, and the R2 simulation. These platforms were stand-ins for the ultimate target system, which is a live R2 robot.

TRACBot

TRACBot is a dual-arm mobile robot developed at TRACLabs for the purpose of research into mobile manipulation. The base is a differential-drive MR-100 from Sensible Machines. The two 7-DOF Reconfigurable Modular Manipulators (RMM) were designed under a previous NASA grant. Sensors include a Hokuyo laser scanner and a Bumblebee stereo vision system. The control architecture is based on the Robot Operating System (ROS) [5].

For this project, a simple "Blocks World" was developed, which included the mobile manipulator and three different colored blocks on a table. Snapshots include the locations of all three blocks as well as the state of the robot.

The Gazebo-based TRACBot Simulation (see below) was used as the modeling engine for the Explorer, although without the graphical display.

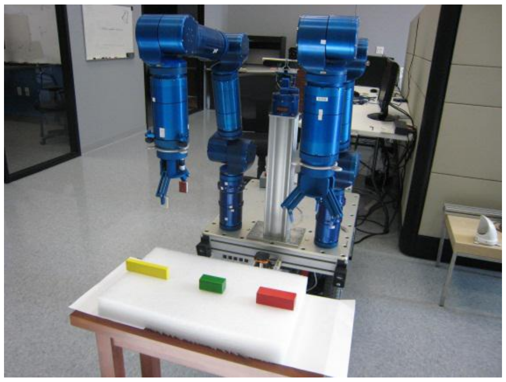


Figure 6: TRACBot in Blocks World.

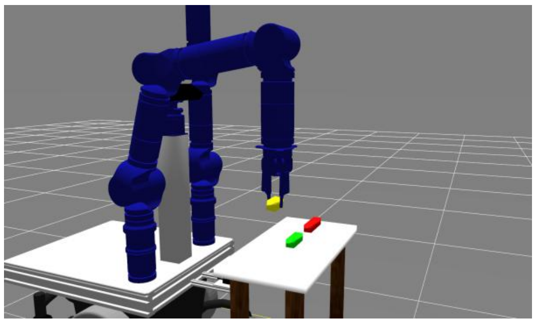


Figure 7: TRACBot Sim in Blocks World.

TRACBot Simulation

The TRACBot Simulation uses ROS and Gazebo. Although Gazebo has improved tremendously as a result of OSRF's efforts with the DARPA Virtual Robotics Challenge [6], it still has difficulties with complex physical interactions. The Blocks World described in the section above was modeled in Gazebo.

The API for the TRACBot Simulation is identical to that of the TRACBot itself. For demonstrations with this platform, the TRACBot Simulations was also used for the Explorer. Needless to say, the Explorer predictions matched the "ground truth" of the TRACBot Simulation quite well.

R2 Simulation

The R2 Simulation was developed at NASA/JSC using ROS and Gazebo. The robot's torso is mounted in front of a Taskboard that contains buttons and switches of various kinds. The Taskboard model is derived from a physical Taskboard that has been used on ISS for R2 experiments. Snapshots include the state of all of the buttons and switches, as well as the state of the robot. The R2 Simulation may be seen in Figures 2 through 5.

For demonstrations with this platform, the R2 Simulation is also used for the Explorer.

4 PROPHESY

During the DARPA Robotics Challenge (DRC) Finals, [7], TRACLabs developed a different way to handle degraded network communication by predicting robot action. Unlike the predictable latency that comes from the time-of-flight for light over long distances, the DRC Finals introduced total telemetry blackouts with random one-second data bursts at full bandwidth. Because time was limited, it was important to reduce robot idle time between data bursts by enabling the operator to make progress planning the next actions for the robot.

Every data burst contributed to the 3D point cloud of the Finals arena. In the Cockpit, the human operator would match an *affordance template* (AT) [8] representing the desired task to the sensor data. An AT encapsulates a parametrized script for the robot actions necessary for accomplishing a task. Once the AT was matched to the sensor data, the system conducted a search for a stance that allowed it to perform the manipulation task. When a good stance is detected, the system can find a trajectory of footsteps that lead from the current robot location to the desired stance.

The *prophet* is an animation of the robot executing the AT script within the 3D point cloud [9]. The operator observes the prophet to make sure that the behavior look acceptable and no unforeseen collisions will occur. If so, the script may be triggered.

As with PIGI, the Prophesy system enables the human to operate in a 3D visualization tool that incorporates the latest sensor data and models of the robot. By decoupling the human from the live data stream, the details of the network degradation become less important and progress can be made in spite of it. If new data arrived while the operator was working with Prophesy, the operator would need to choose to adjust the AT to fit the updated point cloud. This differs from PIGI, where the Predictor would update FinComState automatically.

5 CONCLUSION

TL-PIGI introduces a methodology for supervising a dexterous manipulator across time delay. The Snapshot is introduced as a common unit of communication, and operator tools were developed for examining sequences of Snapshots. The Predictor and Explorer function similarly, but the Predictor uses a very fast, lower-fidelity finite state machine to model the behavior of robot and environment while the Explorer uses the slower, higher-fidelity Gazebo simulation. The ability to browse the history of past commands and robot actions is very useful, especially for diagnosing failed commands.

Acknowledgement

Portions of this work were supported by NASA grants NNJ06HD34C and NNX11CB49C. The authors would like to thank Kim Hambuchen of NASA/JSC for her support.

References

[1] Mittman, David S., Norris, Jeffrey S., Torres, Recaredo J., Hambuchen, Kimberley A., Hirsh, Robert L., Allan, Mark B., Utz, Hans H., Burridge, Robert R., and Seibert, Marc A. (2008) The Exploration Technology Development Program Multi-center cockpit. In: *proceedings of 2008 AIAA Space conference*, San Diego, CA.

[2] Fong, Terrence, Provencher, Chris, Micire, Mark, Diftler, Myron, Burka, Reginald, Bluethmann, Bill, and Mittman, David (2012) The Human Exploration Project: Objectives, Approach, and Testing. In: proceedings of *2012 IEEE Aerospace Conference,* Big Sky, MO.

[3] Burridge, Robert R., and Hambuchen, Kimberly A. (2009) Using prediction to enhance remote robot supervision across time delay. In: *proceedings of 2009 IEEE/RSJ international conference on intelligent robots and systems (IROS)*, St. Louis, MO, pp. 5628-5634.

[4] N. Koenig and A. Howard, \Design and use paradigms for gazebo, an open-source multi-robot simulator," Tech. Rep. CRES-04-002, USC, 2004.

[5] Morgan Quigley, Ken Conley, Brian Gerkey, Josh Faust, Tully Foote, Jeremy Leibs, Eric Berger, Rob Wheeler, and Andrew Mg. Ros: an open-source robot operating system. In: *proceedings of the international conference on robotics and automation (ICRA)*, 3:5, 2009.

[6] Aguero, Carlos E., Koenig, Nate, Chen, Ian, Boyer, Hugo, Peters, Steven, Hsu, john, Gerkey, Brian, Paepcke, Steffi, Rivero, Jose L., Manzo, Justin, Krotkov, Eric, and Pratt, Gill (2015) Inside the Virtual Robotics Challenge: Simulating Real-Time Robotic Disaster Response. In: *IEEE Transactions on Automation Science and Engineering,* 12:2, pp. 494-506.

[7] Defense Advanced Research Projects Agency (DARPA), “DARPA Robotics Challenge (DRC) and Virtual Robotics Challenge (VRC).” 2015. [Online]. Available: http://theroboticschallenge.org/about

[8] Hart, Stephen, Dinh, Paul, Hambuchen, Kimberley (2015) The affordance template ROS package for robot task programming. In: *proceedings, IEEE international conference on robotics and automation (ICRA).* Seattle, WA.

[9] James, Joshua, Weng, Yifan, Hart, Stephen, Beeson, Patrick, and Burridge, Robert (2015) In: *proceedings of the IEEE/RSJ international conference on humanoids,* Seoul, South Korea.