Toward A User-Guided Manipulation Framework for High-DOF Robots with Limited Communication

Nicholas Alunni*, Calder Phillips-Grafflin*, Halit Bener Suay*, Daniel Lofaro†
Dmitry Berenson*, Sonia Chernova*, Robert W. Lindeman*, Paul Oh†
*Worcester Polytechnic Institute
{nalunni, cnphillipsgraffl, benersuay, dberenson, soniac, gogo}@wpi.edu
†Drexel University
dan@danLofaro.com, paul@coe.drexel.edu

***Abstract*—This paper presents our progress toward a user-guided manipulation framework for High Degree-of-Freedom robots operating in environments with limited communication. The system we propose consists of three components: (1) a user-guided perception interface which assists the user to provide task level commands to the robot, (2) planning algorithms that autonomously generate robot motion while obeying relevant constraints, and (3) a trajectory execution and monitoring system which detects errors in execution. We have performed quantitative experiments on these three components and qualitative experiments of the entire pipeline with the PR2 robot turning a valve for the DARPA Robotics Challenge. We ran 20 tests of the entire framework with an average run time of two minutes. We also report results for tests of each individual component.**

I. Introduction

We seek to create a user-guided manipulation framework for High Degree-of-Freedom (DOF) robots such as humanoids and mobile manipulators operating in environments with limited communication. Application of our framework to these robots is conducive to greater autonomy and enables tasks ranging from home maintenance and care for the elderly or disabled to disaster response in conditions that are hazardous to humans. While a great deal of research has explored methods for perception [1], error-recovery [2], motion planning [1, 3, 4], and tele-operation [5, 6], for such applications our goal is to unify existing algorithms in a reliable general-purpose manipulation framework.

This paper presents our progress toward such a framework. We will evaluate our framework by performing valve turning, which is one of the tasks required for the DARPA Robotics Challenge (DRC) [7]. The task requires that a robot locate, approach, grasp, and turn an industrial valve with two hands. Valve turning presents a challenging test-case for our system due to the perception and dexterous manipulation required.

A core constraint for the DRC is that communications with the robot are limited, making conventional tele-operation infeasible and necessitating the use of a framework such as ours. Thus, the valve-turning task requires a straightforward way for a user to command the robot to perform complex actions. These actions require accurate localization of the valve relative to the robot, constrained motion planning for closed-chain kinematic systems, and autonomous error detection to report problems back to the user. These goals align well with creating a general-purpose manipulation system.

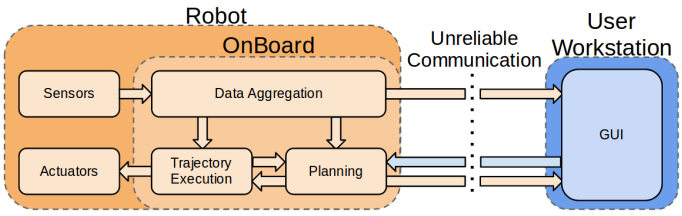


Fig. 1: System diagram showing data flow through the framework.

The system we propose consists of three main parts: (1) a user-guided perception interface which provides task-level commands to the robot, (2) planning algorithms that autonomously generate robot motion while obeying relevant constraints, and (3) a trajectory execution and monitoring system which detects errors in execution. Our goal is that all three of these parts be usable on different robots in both the real world and simulated environments.

In the first component, a user roughly aligns a model of the relevant object (e.g. a valve) to a point cloud provided by the robot's sensors. While autonomous perception algorithms have previously been developed for such tasks, they are unsuitable for highly unstructured environments and underspecified tasks like those encountered in real world situations. Thus, we use Iterative Closest Point algorithms to reduce error and "snap" the rough user-generated alignment into place locally. Once satisfied with the alignment, the user commands the robot to perform the task.

The manipulation planning component of the system consists of the CBiRRT algorithm [8], which is capable of generating constrained quasi-static motion for High-DOF robots. Once a motion path is constructed by the planning component, it is executed by the execution monitoring component. The monitoring component compares the execution of the current trajectory to a library of previous executions of the same task (generated from previous runs) to detect errors. This component uses Dynamic Time Warping (DTW) [9] to compute an error metric between trajectory executions. We have found that our user interface has a success rate of 97% when given a good

user guess at the object's position. The planning algorithm we used successfully generated feasible object manipulation trajectories under constraints 93.84% of the time, and our trajectory execution error detector correctly identified 88% of trajectories.[1]

The rest of the paper is structured as follows: Section II gives a background on the relevant technologies and topics, Section III describes the system architecture and components. Section IV shows the quantitative analysis of our framework and Section V shows the preliminary results on the PR2 and Hubo robots. In Section VI we discuss future work and finally Section VII concludes the paper.

II. Background

There are a variety of robot frameworks and simulation sofware that is freely and commercially available [10, 11]. Different robot control architectures are made based on these frameworks or designed from scratch for different purposes such as remote teleoperation and control of unmanned vehicles from a command post [12, 13, 14]. Although previous research has covered the effects of limited bandwidth communication channels [15], and planetary exploration with limited communication [16], to the best of our knowledge, there is no available framework for high degree of freedom robots, unlike UAVs or rovers, that is tailored for user guided object manipulation in unstructured environments with limited connection to the robot.

III. Architecture

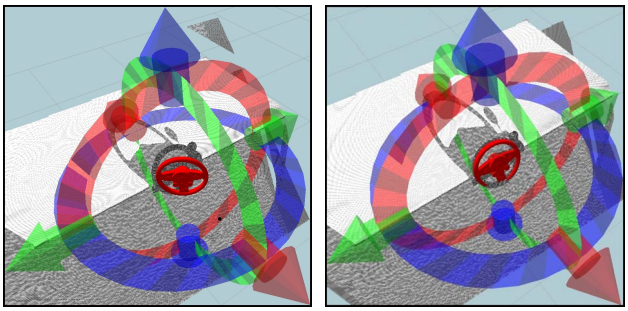
Our framework,shown in Figure 1, is implemented using ROS [17] for communication and robot control and OpenRAVE [18] for motion planning.

A. Data Aggregation

The principle function of the data aggregation package is to format data coming from the robot. The data aggregation package takes in sensor data coming from the robot, which varies depending on the robot, and re-publishes it in a standard format so that the framework can be easily implemented on a variety of robots. As shown in Figure 1, data aggregation is the only component of the framework that receives data such as point clouds, encoders, and accelerometers directly from a robot's sensors. This design allows the system to be highly modular and quickly switch between different robots, including switching between robots operating in real and simulated environments. If necessary, this component can be reconfigured online to handle changes in the available sensor data, such as changing which point cloud topic to use throughout the system.

The aggregation package also provides synthesized information such as collision maps and object proximity that is derived from raw sensor data. This information synthesis is performed on-board the robot to reduce the need for communication. For instance, collision maps generated from downsampled point clouds reduce the data needs by nearly 90%, and object proximity information provides an even greater reduction.

[1]A video of the framework in operation can be seen at: http://www.youtube.com/watch?v=xRcUO2mXt3s



(a) Misaligned Interactive Marker (b) Auto-aligned Interactive Marker

Fig. 2: Iterative Closest Point being used to align an object in RVIZ. (a) the object before ICP has been run, (b) the final translation after ICP has finished.

B. User Interface

Due to the difficulty of autonomous perception, a graphical user interface (GUI) was created to aid the detection of objects. Using the GUI, as shown in Figure 2a, the user manipulates an interactive marker [19] to hint at an object's location. Object alignment is performed using the Iterative Closest Point (ICP) algorithm which minimizes the error between two specified groups of points. ICP "snaps" a given input to the target world, as shown in Figure 2b, by iteratively computing the transformation between the two groups of points. Larger transformations can be found by increasing the number of iterations performed. To decrease computation time, a bounding box is used to extract a subset of the point cloud. For our testing, the point cloud from the robot is generated using an ASUS Xtion RGBD camera.

In addition to user input and feedback, the GUI controls the flow of data over the unreliable link to the robot. Data from the robot is only transmitted when specifically requested to minimize communication. This architecture takes advantage of the assumption that the robot inhabits a largely static environment, such as the DRC's valve turning task, while still remaining suitable for use in more dynamic environments.

C. Planning

The planning package plans trajectories for high degree of freedom robots so that they can perform object manipulation. The initial configuration of the robot is critical to manipulation because the robot must be able to:
(1) reach and manipulate the object for the entirety of the desired trajectory,
(2) maintain balance during execution,
(3) avoid self-collisions and collisions with the environment.

Motion planning is provided by the Constrained BiDirectional Rapidly-exploring Random Tree (CBiRRT), an efficient and probabilistically complete manipulation planning suite. CBiRRT consists of three main components: constraint representation, constraint-satisfaction, and a general planning

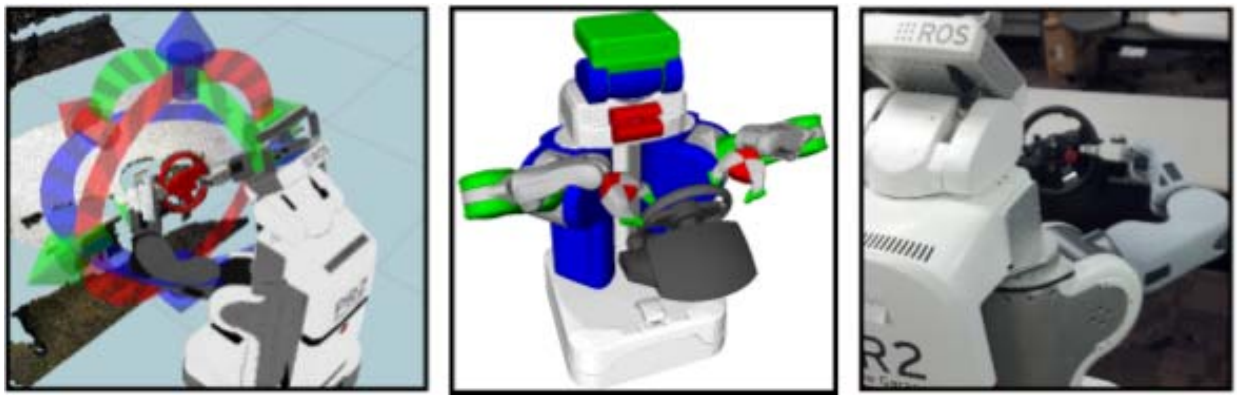


Fig. 3: The PR2 Robot as seen in the RVIZ visualization engine performing valve alignment (left), OpenRAVE for motion planning (middle), and the real world performing valve turning (right).

algorithm. For full details of CBiRRT and its implementation, see [8].

D. Trajectory Execution

The trajectory execution package executes a planned trajectory and detects errors encountered during execution. For this error detection, trajectories are recorded during execution using only the data available from joint encoders. No other contextual data, such as the planned trajectory or the pose of the object being manipulated, is required.

Error in trajectory execution is identified by using the dynamic programming technique Dynamic Time Warping (DTW) to match executed trajectories against a library of known successful and unsuccessful trajectories. DTW iteratively calculates the best alignment between elements of two or more time sequenced data [9] and produces a cost metric that quantitatively represents the similarity of those sequences to either the successful or unsuccessful class, which facilitates error detection during execution. To account for trajectories significantly different from those in the library, cases in which the computed DTW cost metric is greater than an experimentally-determined threshold can be automatically identified as error conditions.

This method of error detection using DTW is ideal as it requires no complex visual feedback and no special sensors, and is thus applicable to a wide range of robots using only the data already available from basic joint encoders. In particular, this method is well-suited for the DRC where it may not be possible to determine the state of the valve though other means. For our testing, we used this approach to determine if the valve manipulated by the PR2 was successfully turned.

IV. Framework Validation

The framework we have developed allows for a user to hint at the location of an object in the world and have the robot approach and manipulate the object. In order to perform this action the pose of the object needs to be determined, a trajectory generated to manipulate it from a start position, and finally the trajectory must be monitored for errors. Quantitative experiments were performed on the three aforementioned components of the architecture, and qualitative experiments were run with the PR2 robot turning a wheel in both simulation and the real world (see Figure 3). Additionally, we report on preliminary validation experiments performed using the Hubo humanoid robot.

A. Valve Alignment

To enable semi-automated testing, the GUI provides an option to automatically generate an object alignment with a configurable amount of noise. This testing configuration includes the number of tests to run, a maximum amount of translation error, and a maximum amount of rotational error. We used the semi-automated tester to evaluate the valve alignment system by randomly perturbing the valve's position. A simulated user guess error is calculated by adding the total translation offset in cm to the total quaternion angle offset in degrees. The true position of the valve is denoted as $V_o = [X_o, Y_o, Z_o]$ and the guessed position of the valve is denoted as $V_g = [X_g, Y_g, Z_g]$. The total translation value E_t is calculated as the euclidean distance between the two points. Each valve's pose also contains a quaternion that represents its orientation in space. The difference in angle between the quaternion representing the valve's position and the guessed position is represented.

$$E_q = \arccos(2*((x_o*x_g)+(y_o*y_g)+(z_o*z_g)+(w_o*w_g))^2-1) \quad (1)$$

Finally, the user guess error is calculated by the sum of the translational error and rotational error, and is denoted by $E = E_t + E_q$.

Figure 4 shows the success rate of 450 sample alignments with random perturbations, where success is defined as a final error of less than .3 units. The "user guess error" is the amount of error that was present when the semi-automated tester requested ICP to align the valve to the point cloud. We qualitatively categorized the results through experimentation. One to five error units is considered a good user guess, six to ten error units is considered an acceptable user guess, and eleven to fifteen error units is a "poor" user guess. ICP was capable of matching the interactive marker to the point cloud 97% of the time for a good guess, 79% of the time for an

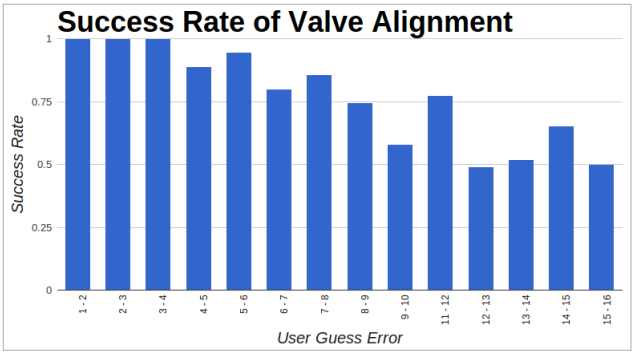


Fig. 4: Success rate (final error less than .3 units) of Iterative Closest Point to find the actual valve position given a randomly generated perturbation in the user's estimated position.

(a) Initial Configuration (b) Start Pose (c) Goal Pose

Fig. 5: Configurations for valve turning path.

acceptable guess, and 58% of the time for a poor guess. The number of ICP iterations, 500, remained constant throughout all tests. The number of iterations was determined so that ICP returns with a new valve alignment in under one second. The number of iterations ICP runs can be increased, allowing larger transformations to be found at the cost of longer runtime.

B. Planning

We tested the CBiRRT trajectory planner's performance with the constraints described in Section III under different sensing disturbances of the object (in this case, the valve). We first fixed the position of the object to a point in space, where we knew the planner was able to succeed. We then added random translation and rotation perturbations to the transform of the valve and tried to generate a trajectory plan for the translated and rotated model of the valve.

For each test generated, we defined six pose constraints and two path constraints in the CBiRRT framework. For the valve turning task, we designed a path that consists of four trajectories:
(1) From initial pose (*init*) to valve grasping configuration (*start*),
(2) From valve grasping configuration (*start*) to the configuration right after clockwise turning the valve 45 degrees (*turned*),
(3) From *turned* back to *start*,
(4) From *start* back to *init*.

Init, start and goal configurations are shown in Figure 5. For all four trajectories, there were three positions that each hand must pass through: an initial position, a start position and a goal position. There was one path constraint for each end effector for turning the valve about its axis of rotation during trajectories (2) and (3).

We used random quaternion angle difference perturbations between zero and 85 degrees and translational perturbations between 0 and .1 meters. The randomly generated poses were first validated by the inverse kinematics (IK) solver to ensure that the start and end configurations necessary for the robot were collision-free; 422 points were confirmed valid by the IK solver and used for testing. Of those 422 points in the space, the trajectory planner succeeded in planning all four trajectories 93.84 percent of the time.

C. Trajectory Execution

We generated a library of known valid and invalid trajectories to test the validity of the execution of planned trajectories. When both hands maintained contact with the object throughout the entire trajectory it was considered valid, if either hand missed or lost contact with the valve it was considered invalid.

We introduced random rotational noise between 0 and 10 degrees and translational noise between 0 and .025 meters. These values were selected to represent reasonable valve misalignments given the observed noise and error of the sensors on the PR2.

We manually created an initial library of 22 known valid and invalid example trajectories by providing correct and incorrect alignments of the valve to the planner, and tagged the resulting trajectory as either successful or not. The library was then grown to 526 trajectories through experiments.

In this system, DTW is sufficiently fast to enable online evaluation of individual trajectories. Each evaluation against the 526-element library is completed in less than four seconds, whereas execution of the actual trajectories requires approximately 32 seconds. We performed leave-one-out testing of the library itself in which each known trajectory in the library was compared against the other trajectories in the library to test the ability of the error detector to correctly identify if an error was encountered. This testing resulted in an overall correct identification rate of 88%, the false positive rate was 10% and the false negative rate was 16%.

This discrepancy between identification rates is acceptable because the cost potentially incurred from a false negative identification (which would result in a retry of the alignment and planning) is significantly lower than that from a false positive (which would result in prematurely terminating the valve turning task).

D. Full Framework Testing

In order to test the communication and data collection across the system, we ran 20 complete test cycles of the full framework. The test procedure was as follows: we manually drove the PR2 to a random location in the room from where it could see the valve, then an expert user identified the location of the valve using the GUI, and sent the location of the valve to the planning component. Once the PR2 received the valve's location, it approached the valve autonomously, planned valve turning trajectories, and then executed those trajectories. The average time to run the entire framework from

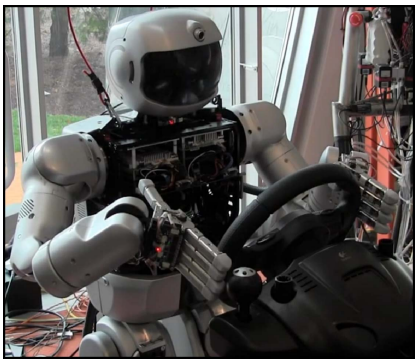


Fig. 6: Hubo2 Plus turning a valve.

start to finish was approximately two minutes. On average, turning towards the valve and driving to a location where it could be manipulated took 30 seconds, planning took two to three seconds depending on the PR2's position relative to the valve, and turning the valve took approximately 90 seconds. Trajectory execution accounted for 64 of those seconds, trajectory classification with DTW took 12 seconds, and closing and opening the grippers accounted for the remainder.

V. Robot Experiments

We have applied this framework to a Willow Garage PR2 performing the valve turning task. We created a valve analog from a commercially available force feedback racing wheel. We tested our framework on the PR2, after validating the safety of all tests in a simulation environment. The distance the PR2 started away from the valve was varied randomly between one meter and three meters. The orientation of the PR2 was also varied so that it was not directly facing the wheel, with the requirement that the wheel be in the point cloud. The addition of a search algorithm to find valves not visible in this starting position is discussed in the Future Work section.

Observed performance of the trajectory execution system on the PR2 shows that false negative identifications correlate to cases where the compliance in the PR2's arms can cause the same trajectory to either succeed or fail. Notably, the overall correct identification rate for unsuccessful trajectories is higher than that for successful trajectories, most likely a result of known unsuccessful trajectories outnumbering known successful trajectories in the library.

A. Hubo Testing

In contrast to the more systematic testing on the PR2 robot described above, our preliminary experiments with the Hubo were centered around validating our method of motion planning for the robot and evaluating the robot's capabilities in relation to the requirements of our DRC task (turning the valve). These tests were performed on the Hubo2 Plus at MIT, housed in the lab of Professor Russ Tedrake.

Hubo2 Plus is a 130 cm ($4'3''$) tall humanoid robot commonly refered to as Hubo, see Fig. 6. It was designed and constructed by Prof Jun-Ho Oh at the Hubo Lab in the Korean Advanced Institute of Science and Technology (KAIST) [20, 21]. Hubo is anthropomorphic to a human meaning it has two arms, two legs and a head. There are six degreese of freedom (DOF) in each leg, six in each arm, five in each hand, three in the neck, and one in the waist, all totalling 38 DOF.

We executed several open-loop valve-turning trajectories generated by the planning system on the Hubo. Our experiments confirmed that the planning system enabled control of the Hubo and that the Hubo was physically capable of turning the valve.

VI. Future Work

Our framework provides significant room for expansion to the user interface, planning system, and trajectory execution monitor. The current interface provides neither a method of searching for an object nor autonomous object detection. Future versions will combine these with a variety of improvements to increase user situation awareness.

Our current motion planning system is dependent on the manual generation of both initial configurations and task constraints. Automatic generation of these is an important avenue for further work. We plan to develop an automated configuration predictor using human-agent knowledge transfer techniques that have been shown to be effective for teaching agents different types of tasks [22, 23].

For trajectory execution, which is the final stage in our system, we will extend our implementation to identify different kinds of error conditions. We plan to improve the performance of the underlying DTW implementation using a variety of established [9] and novel techniques [24] to use not only larger trajectory libraries, but also to increase the resolution at which we evaluate the trajectories for errors.

VII. Conclusion

In this paper, we presented the foundation of a novel framework for user-guided manipulation with High Degree-of-Freedom robots in environments with limited communication. We described techniques for object identification, constrained trajectory generation, and trajectory monitoring. We presented a quantitative evaluation of all major components in simulation, and qualitative experiments on the framework as a whole.

Acknowledgement

This work was supported in part by the Defense Advanced Research Projects Agency (DARPA) award #N65236-12-1-1005 for the DARPA Robotics Challenge. We would also like to thank Russ Tedrake of CSAIL, MIT for allowing us to use his Hubo for testing.

References

[1] S. Chitta, E. G. Jones, M. Ciocarlie, and K. Hsiao, "Perception, planning, and execution for mobile manipulation in unstructured environments," *IEEE Robotics and Automation Magazine, Special Issue on Mobile Manipulation*, vol. 19, 2012.

[2] S. Butner and M. Ghodoussi, "Transforming a surgical robot for human telesurgery," *Robotics and Automation, IEEE Transactions on*, vol. 19, no. 5, pp. 818 – 824, oct. 2003.

[3] *Bayesian Grasp Planning*, 2011.

[4] O. Khatib, "A unified approach for motion and force control of robot manipulators: The operational space formulation," *Robotics and Automation, IEEE Journal of*, vol. 3, no. 1, pp. 43 –53, february 1987.

[5] *Strategies for Human-in-the-Loop Robotic Grasping*, Boston, MA, 2012.

[6] T. L. Chen, M. Ciocarlie, S. Cousins, P. Grice, K. Hawkins, K. Hsiao, C. C. Kemp, C.-H. King, D. A. Lazewatsky, A. Leeper, H. Nguyen, A. Paepcke, C. Pantofaru, W. D. Smart, and L. Takayama, "Robots for humanity: User-centered design for assistive mobile manipulation," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, oct. 2012, pp. 5434 – 5435.

[7] D. R. C. T. T. O. (TTO), "Darpa robotics challenge," 2012, [Online; accessed 16-Jan-2013].

[8] D. Berenson, S. Srinivasa, and J. Kuffner, "Task space regions: A framework for pose-constrained manipulation planning," *International Journal of Robotics Research (IJRR)*, vol. 30, no. 12, pp. 1435 – 1460, October 2011.

[9] P. Senin, "Dynamic time warping algorithm review," *Information and Computer Science Department University of Hawaii at Manoa Honolulu, USA*, pp. 1–23, 2008.

[10] A. Harris and J. Conrad, "Survey of popular robotics simulators, frameworks, and toolkits," in *Southeastcon, 2011 Proceedings of IEEE*, march 2011, pp. 243 –249.

[11] J. Craighead, R. Murphy, J. Burke, and B. Goldiez, "A survey of commercial open source unmanned vehicle simulators," in *IEEE International Conference on Robotics and Automation*, april 2007, pp. 852 –857.

[12] A. A. D. Medeiros, "A survey of control architectures for autonomous mobile robots," *Journal of the Brazilian Computer Society*, vol. 4, 04 1998.

[13] B. Hannaford, "A design framework for teleoperators with kinesthetic feedback," *Robotics and Automation, IEEE Transactions on*, vol. 5, no. 4, pp. 426 –434, aug 1989.

[14] R. C. Arkin and T. Balch, "Aura: Principles and practice in review," *Journal of Experimental and Theoretical Artificial Intelligence*, vol. 9, pp. 175–189, 1997.

[15] P. Rybski, S. Stoeter, M. Gini, D. Hougen, and N. Papanikolopoulos, "Effects of limited bandwidth communications channels on the control of multiple robots," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, vol. 1. IEEE, 2001, pp. 369–374.

[16] P. Pirjanian, T. Huntsberger, A. Trebi-Ollennu, H. Aghazarian, H. Das, S. Joshi, and P. Schenker, "Campout: A control architecture for multi-robot planetary outposts," in *Proc. SPIE Symposium on Sensor Fusion and Decentralized Control in Robotic Systems III*, vol. 4196, 2000, pp. 221–230.

[17] M. Quigley, B. Gerkey, K. Conley, J. Faust, T. Foote, J. Leibs, E. Berger, R. Wheeler, and A. Ng, "Ros: an open-source robot operating system," in *ICRA workshop on open source software*, vol. 3, no. 3.2, 2009.

[18] R. Diankov and J. Kuffner, "Openrave: A planning architecture for autonomous robotics," *Robotics Institute, Pittsburgh, PA, Tech. Rep. CMU-RI-TR-08-34*, 2008.

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

[20] B.-K. Cho, S.-S. Park, and J. ho Oh, "Controllers for running in the humanoid robot, HUBO," in *IEEE-RAS International Conference on Humanoid Robots*, dec. 2009.

[21] D. Lofaro and P. Oh, "Humanoid throws inaugural pitch at major league baseball game: Challenges, approach, implementation and lessons learned," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, nov. 2012.

[22] B. D. Argall, S. Chernova, M. Veloso, and B. Browning, "A survey of robot learning from demonstration," *Robot. Auton. Syst.*, vol. 57, no. 5, pp. 469–483, May 2009. [Online]. Available: http://dx.doi.org/10.1016/j.robot.2008.10.024

[23] M. E. Taylor, H. B. Suay, and S. Chernova, "Integrating reinforcement learning with human demonstrations of varying ability," in *The 10th International Conference on Autonomous Agents and Multiagent Systems - Volume 2*, ser. AAMAS '11. Richland, SC: International Foundation for Autonomous Agents and Multiagent Systems, 2011, pp. 617–624. [Online]. Available: http://dl.acm.org/citation.cfm?id=2031678.2031705

[24] D. Sart, A. Mueen, W. Najjar, E. Keogh, and V. Niennattrakul, "Accelerating dynamic time warping subsequence search with GPUs and FPGAs," in *Data Mining (ICDM), 2010 IEEE 10th International Conference on*, dec. 2010, pp. 1001 –1006.