Expanding Task Functionality in Established Humanoid Robots

Victor Ng-Thow-Hing*, Evan Drumwright[†], Kris Hauser[‡], Qingquan Wu[§], and Joel Wormer*
*Honda Research Institute USA
Mountain View, CA, 94041, USA
{vng,jwormer}@honda-ri.com
[†]Computer Science Department
University of Memphis
[‡]Computer Science Department
Stanford University
[§]Electrical and Computer Engineering Department
University of Illinois at Chicago

Abstract— Many humanoids robots like ASIMO are built to potentially perform more than one type of task. However, the need to maintain a consistent physical appearance of the robot restricts the installation of additional sensors or appendages that would alter its visual identity. Limited battery power for free-moving locomotive robots places temporal and spacial complexity limits on the algorithms we can deploy on the robot. With these conditions in mind, we have developed a distributed robot architecture that combines onboard functionality with external system modules to perform tasks involving interaction with the environment. An information model called the Cognitive Map organizes output produced by multiple perceptual modules and presents a common abstraction interface for other modules to access the information. For the planning and generation of motion on the robot, the Task Matrix embodies a task abstraction model that maps a high level task description into its primitive motions realizable on the robot. Our architecture supports different control paradigms and information models that can be tailored for specific tasks. We demonstrate environmental tasks we implemented with our system, such as pointing at moving objects and pushing an object around a table in simulation and on the actual ASIMO robot.

I. Introduction

When designing robots to perform a particular task, roboticists often have the freedom to choose the configuration of the robot to be suitable to the task. This includes the selection of body shape, manipulators, sensors, actuators, degrees of freedom and software. With this approach, a highly specialized and capable robot can be created to maximize the chances of successfully performing the task. Representative examples of this approach are the RiSE climbing robotic platform [1] with its hexapedal feet equipped with compliant microspines, origami-folding robots [2] and the use of compliant graspers to adapt to different object shapes [3].

In contrast, humanoid robots are designed to emulate human morphology and are therefore not built with only a single task in mind. Consequently, the constraints of humanoid robots can restrict solutions for task implementation due to limited placement of sensors and degrees of freedom in order to maintain the basic human body shape and aesthetic appearance. Specializing the robot's manipulators for one particular task may deem it unusable for many other tasks.

In the case of Honda Motor Company's ASIMO [4], the robot also serves as an ambassador to the company, and must maintain a recognizable and consistent appearance at all times. For researchers working with ASIMO, this requirement warrants careful consideration on the choice of sensors, and limits the use of external appendages that detract from the original robot design.

There are also technical limitations that can occur when altering robots. Many humanoid robots have an internal model of their mass distribution that is utilized for balancing algorithms. Substantial alteration of link properties may introduce mass perturbations that can affect operational performance. Details of the wiring or construction of a robot may not be available, creating a technical barrier to the insertion of new devices to the robot's existing hardware infrastructure. For untethered humanoid robots, the power consumption limits of the robot's batteries prevent the fastest CPU and GPU configurations from being installed on the robot.

This paper describes several solutions that we have developed to allow us to research and implement complex tasks on ASIMO without altering its mechanical configuration or changing its sensing capabilities. These same methods can be easily applied to other humanoid robots. We restrict ourselves to the original configuration of the year 2000 ASIMO model [4]: 26 degrees of freedom, 0.5kg/hand grasping force, 6-axis foot area sensors in the feet and gyroscope and acceleration sensors in the torso. In particular, we discuss approaches we pursued in organizing environmental state, localizing the robot, and commanding motion to perform complex tasks with an established humanoid robot whose hardware and software cannot be significantly altered.

A. Robot Independence

ASIMO has a collection of different sensors and motion control software whose processing was originally limited to reside onboard the robot only. Usually, it is convenient for researchers to independently develop their algorithms using external computers and sensors. However, they are often later confronted with the difficult task of reimplementing their software to adapt to the different hardware and operating systems on the robot. To minimize the changes to their own software, we developed robot independent strategies for control and accessing perceptual information. This strategy also minimizes the need to rewrite software when migrating to new robot models in the future. In the Player system for mobile robots [5], abstraction is used to provide a common logical interface to different hardware sensors and actuators. Similarly, we use abstraction when transferring information from sensors and sending desired joint angle motion commands to ASIMO. A common motion command interface hides the details of our robot's low-level control. Similarly, standardized interfaces for sensors and the information they provide allows our researchers to experiment with different implementations of vision modules or sensor modalities. We employ a blackboard architecture [6] to provide a mechanism to share and transform information between different modules (Figure 1).

B. Distributed Systems

For an autonomous humanoid robot, untethered from an external power source, the energy needs for actuators, sensors and processing boards must be supplied by onboard batteries. This constraint limits the time and space complexity of algorithms that can be run directly on the robot. Like many other robot architectures [7], [8], we designed a distributed system to offload the processing of sensor information, path planning with collision avoidance and other computationally expensive tasks from the robot. This method allows us to extend the capabilities of the robot, but requires new mechanisms for communication between all modules in the system. In Section II, we address the problem of information sharing in our distributed system.

Distributed systems allow many time-consuming functions to be computed in parallel. For example, we can simultaneously perform large searches for valid motion plans while executing other motions on the robot and extracting features from sensory data from another machine. For our researchers, switching to the distributed system in many cases no longer required them to migrate their software to a new hardware environment. Instead, they would add a single routine to their existing program's execution loop to periodically send their output to the networked data channel.

C. System Overview

Our distributed system (Figure 1) consists of several high level components responsible for internal and environmental state management and task execution. The Cognitive Map represents a blackboard system that provides a common workspace to share information between these modules. Through the Cognitive Map, perceptual modules that extract useful features from raw sensor data can deposit their information to maintain both external environmental state and internal robot state.

For robot motion, the Task Matrix decomposes a high level, parameterized motion command into a set of sequential and/or concurrent task programs that can be robustly executed on the actual robot. An abstract motion interface lies between the internal control code of the robot and the task programs of the Task Matrix. Other modules external to the robot can perform high level activities ranging from deliberative planning tasks like pushing objects on a table to more reactive pointing and gazing at moving objects in the environment (see Section V).

Section II describes several communication mechanisms we created to share environmental or robot state information between modules, while Section III explains how environmental information is used to allow the robot to carry out meaningful tasks on objects in the environment. In Section IV, methods for creating motion on our robot are explained. Section V presents two scenarios using combinations of the methods presented and Section VI concludes with discussion.

II. State Information Management

A key function for high level tasks is the ability to process sensor information to estimate the state of the environment. On ASIMO, sensor information is captured but the choice of algorithms for estimating world state is restricted due to limited computational resources. Furthermore, modern advanced GPU techniques for accelerating video processing are currently off limits to the robot's internal computers. The resulting inaccurate or infrequent state estimates would make it very difficult for the robot to robustly execute high level tasks. The distributed model allows us to use GPU techniques to process video images at a high frequency for radial distortion correction and homography transformations of planar surfaces (Figure 2). The video streams can be captured onboard the robot and redistributed to external modules through networked communication. Perceptual modules then take the raw video stream and perform image processing operations to obtain useful features or information such as object pose. In our applications, we use video streams to compute the pose of objects in the robot's work environment. Sensors can be located both on the robot and situated in the external environment and can be of different modalities: time-of-flight depths sensors, cameras, motion capture systems and the robot's own onboard gyro sensors and joint encoders.

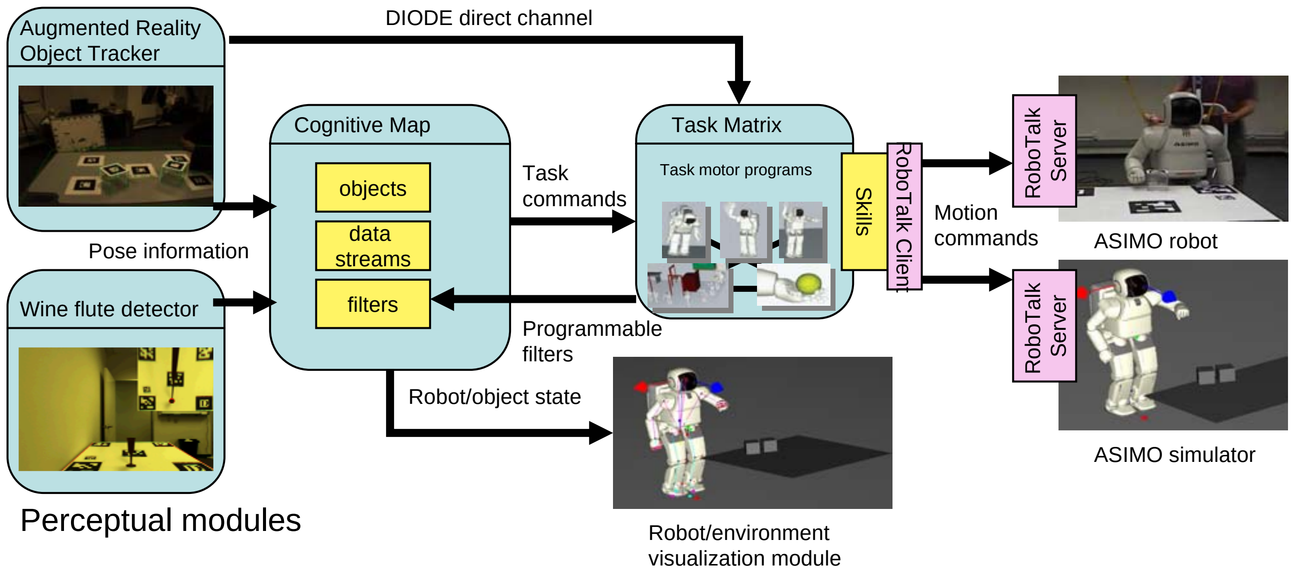

Fig. 1. System overview: Major components of our distributed system

A. Cognitive Map

We define the Cognitive Map as an information space that represents the robot's understanding of both its internal state (eg., pose information, current goals) and its external environment, such as where objects are in the world with respect to its body. All information is collected in the centralized Cognitive Map for redistribution to other modules that are connected to it. Modules can publish information and selective receive events or data samples based on a publish-subscribe model. For example, an object recognition module could subscribe for object detection events that appear within a region of interest in the robot's workspace. Similarly, an object tracker can publish current position estimates to a stream in the Cognitive Map that all subscribers can access. Modules produce information about the state of the world (e.g. perceptual modules), consume information to make decisions or create changes to the world (e.g., motor task modules), and utilize existing knowledge to generate new state information that can be stored back in the Cognitive Map (e.g., localization modules).

Since the Cognitive Map provides a common space for perceptual modules to deposit their measurements and estimates of state, details of the actual sensor hardware or state estimation algorithms are hidden within the modules. We have used this abstraction to substitute different vision algorithms to compare their relative performance in the same usage situations. We have also used the Cognitive Map to combine the information of different perceptual modules to get a more complete picture of the entire environment (see Section III).

The key element of information in the Cognitive Map are generic objects, which like object-oriented classes, can be subclassed to include any degree of custom information. All generic objects have a common set of information like pose information, uncertainty measures for state variables, identification labels and coordinate systems (if applicable). However, knowledge of the type of object may be useful for modules wanting to do object-specific tasks. For example, a physical object can be annotated with its graspable regions. A key concept of our Cognitive Map is that it is not just a passive repository for information. The Cognitive Map is built on the Psyclone "whiteboard" system [9] which combines the shared information concepts of the blackboard architecture with data streams that can be shared, have their data samples timestamped for synchronization, and data content transformed (eg., coordinate conversion) or selectively screened while being transmitted between modules. However, rather than have a predetermined set of screening criteria residing in the Cognitive Map, individual modules are free to provide their own customized filtering criteria written in the Lua scripting language [10] for tailored notification and delivery of data samples that are of interest to the module. For example, a module that plans pushing of objects on a table may only want to receive information about objects when they appear on the table.

B. DIODE

One disadvantage of the Cognitive Map is that all information must be passed through it. It acts as a switchboard for routing streamed information to different modules and generating events based on state changes in the Cognitive Map. This can result in additional overhead as information is streamed into the map from producer modules, processed, then streamed out to the consumer modules. In some applications, a dedicated data channel is preferred between a perceptual module and the user of that perceptual data. For example, a motor task that causes the robot to track and physically point at an object requires minimal lag time between the pose detection of the object and the communication of that information to an inverse kinematics module that points the hand at the object. For this reason, the DIODE (DIstributed Operation via Discrete Events) communication infrastructure provides a faster direct connection between modules that bypasses the Cognitive Map while being capable of restoring communication when the connection is lost.

III. Robot localization and object pose recovery

The current configuration of ASIMO has two cameras installed in the head for vision. Since the physical appearance of ASIMO's head must not be altered to maintain consistent recognizability of the robot, this constraint limits the installation of extra sensors in the head because of the tight space constraints within the helmet. To maximize the flexibility of the vision system, different cameras were used for each camera eye. The dual cameras led to strategies that attempt to reconstruct pose from a single camera to avoid inter-camera calibration. To get around the potential ambiguity of 3-D reconstruction from a single 2-D image projection, we depended on a priori information about the physical dimensions of objects or markers in the scene at a cost of restricting the generality of our object pose reconstruction. Another requirement was that the algorithms work at a fast enough frame rate in order for the motor commands to have access to the most recent state estimates. We built an augmented reality system using ARToolkitPlus [11] to extract the pose of selected objects in our scene as required by the task. Our main objective was to provide adequate sensing to demonstrate several high level manipulation tasks with selected objects, rather than performing pose determination for many different kinds of objects.

Although our system can support external cameras, we decided to attempt all sensing with onboard cameras only so that the robot must actuate its body to keep items of interest in view. Since we are interested in manipulation tasks, we decided to focus on tasks that involve objects on a table. Using our augmented reality system and the knowledge of the table's dimensions, we can recover the pose of the table with respect to the camera and consequently use ASIMO's internal knowledge of its joint configurations to perform the necessary transformations to localize the robot with respect to the table for manipulation. Due to drift that occurs as ASIMO walks around the table, the robot periodically needs to glance at the table to relocalize itself and remove the accumulated drift error.

We have used the knowledge that objects are placed on the table to reconstruct wine flute locations from a single view without the need to place markers on the container.

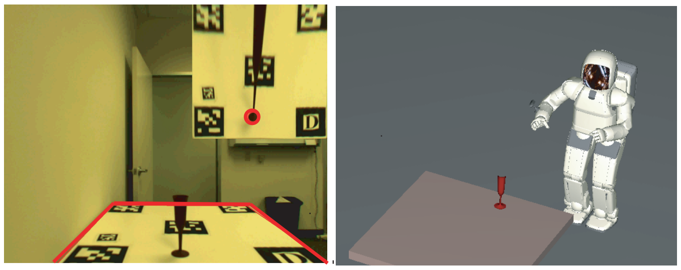

Fig. 2. Left: The cup detector can recover a wine flute's position on a table from a single view by using extracted table coordinates to compute a homography to remove perspective effects (inset). Right: This information is passed on to the robot for task execution. In this case, we send the arm trajectory motions using RoboTalk to a dynamic simulation of the robot to point at the detected wine flute.

The choice of the wine flute object was made because ASIMO's hands can only grasp items of less than 5cm in diameter. The thin stem of the wine flute allows it to be easily graspable with ASIMO's limited manual dexterity. Having the table pose and hence the coordinates of the corners of the table allows us to compute a homography [12] from the perspective view to a simulated overhead orthographic view. From this image, cups and wine flutes have a circular base accompanied by two straight edges corresponding to the sides of the container (Figure 2). Using a circular hough transform [13], the circular bases can be detected and the distance between the two lines connected to the base is used to determine if the detected container is a wine flute (narrow stem) or a can. The cup can then be located in the 2-D coordinate space of the table and subsequently transformed into 3-D coordinates since we know already know the table's pose. Using the recovered pose of the table and objects on it, there is enough information to attempt manipulation tasks on these objects. We implemented a pushing manipulation mode (see Section IV-C) and are currently working on implementing grasping for the wine flute object.

IV. Task Motion Generation

Several paradigms currently exist for controlling motion on robotic platforms. The traditional sense-plan-act (SPA) approach represents a flow of information from sensors, to environment state reconstruction, to planning and execution of the motion plan. Brooks later introduced the subsumption architecture which allowed tight coupling between action responses and sensor outputs, without the need for an internal environment state or advanced planning [14]. In either case, proponents of each control architecture have provided examples where one method would excel over the other. Recognizing this fact, research groups independently developed hybrid solutions, known as 3T (3-Tier) architectures, that combined a low-level reactive feedback layer with a slow deliberative planner via an intermediate sequencing mechanism that communicated between the two layers [15]. Our approach has been to development system components that can support any of these three approaches. Mainly, the architectural implementation can be isolated within one or more modules that connect to the Cognitive Map. This allows a robot to dynamically switch control architectures depending on the task mode it is in. For example, our robot can perform typical sense-plan-act motion programs like pushing objects around a table (Figure 3) or a reactive task by tracking moving objects by continuously pointing and following the object with its hand (Figure 2).

A. RoboTalk Motion Interface

In ASIMO, different parts of the robot's body are controlled by various subsystems. To provide an easy means for researchers to command motions on the robot without learning the details of each subsystem, the creation of a consistent motion interface allowed the robot's degrees of freedom to be set with a simple desired joint angle vector instead of accessing multiple subsystems with different parameters. At the lowest level, the control algorithms that move the various degrees of freedom for the robot reside onboard, including the walking controller which allows positioning of individual footsteps while maintaining balance. We have developed a motion interface called RoboTalk [16] to provide a common set of API calls to instigate motion on the robot by sending desired joint angle positions, task space constraints (for example, end effector conditions), or high level destination targets in stage coordinates (for the walking subsystem).

A driver module implements the native motion commands for ASIMO as described in the RoboTalk interface. In RoboTalk, a server resides on the robot that handles motion command requests from connected clients. These commands are sent over a TCP/IP socket stream connection and can arrive nonuniformly to the server. Consequently, RoboTalk supports specification of timing information and buffering to ensure a smooth motion trajectories of desired joint angles are delivered to the robot's low-level PID controllers. We have used this abstraction interface to also implement a driver for a dynamic simulation of ASIMO to allow testing in simulation before experimenting on the real robot (Figure 1). All modules using the RoboTalk interface do not have to be changed as the robot can transparently be exchanged with the simulator.

B. The Task Matrix

For higher level specification of motion, a module called the Task Matrix [17] provides a mapping apparatus from high level parameterized task commands (eg., PointAt(wine flute), Reach(box)) to low-level RoboTalk motion commands. A common skill set of robot-dependent primitives are implemented using the RoboTalk API to perform reliable, repetitive behaviors. For example, in the Reach task, motion planning is used to find collision-free trajectories for the robot. The Task Matrix queries the Cognitive Map (Section II-A) to convert symbolic object labels to actual 3-D positions and orientations relative to the robot's own coordinate system. Currently, geometry is obtained by matching prebuilt 3-D models with known object labels. However, geometric information can also be potentially obtained through a reconstruction module via the Cognitive Map. The combination of 3-D coordinate information and geometry allows operations, such as collision avoidance in motion planning and inverse kinematics for end-effector positioning, needed for successful task execution. We implemented tasks like pointing (Figure 2), reaching and body gestures within the Task Matrix framework.

C. Manipulation Tasks

As we eventually wish to demonstrate interesting tasks that affect change in an environment, it is mandatory to provide manipulation tasks for ASIMO. Unfortunately, the hands of ASIMO were not originally designed for general manipulation, being constrained to grab objects less than 5 cm in diameter and weighing less than 0.5 kg. The hand only has one degree of freedom, ranging from fully open to fully closed without individual control of the fingers or any touch sensors in the hand. This necessitated a strict reliance on vision sensors to locate the object to be manipulated. With these restrictions in mind, we developed motion modules for pushing and are currently developing grasping of limited objects in order to expand the modes of manipulation.

The pushing manipulation was chosen to expand the size and weight of objects that can be manipulated by ASIMO when pick and place grasping operations are not feasible. We define a push task that attempts to push an object to a specified region on the table. Objects can be pushed around a table, while avoiding collision with the table and non-manipulated objects resting on the table. Since the reach of ASIMO's arms are limited, ASIMO must engage different modes of motion to complete the task. Three distinct modalities were identified and incorporated into a novel multi-modal motion planner [18]: walking, reaching to the object and executing the push. Each modality operates in a constrained region of configuration space. By restricting ourselves to these distinct modalities, motion plans can be generated at an order of magnitude faster than traditional random path planning on the full configuration space of the robot.

V. Results

We have used the Cognitive Map to provide live environmental estimates of the table and block objects for our multi-modal push planner (Figure 3). Augmented reality-based vision modules were created that stream the table and object poses through the Cognitive Map to the push module. The push module can dynamically define criteria to only report the position of objects that are within the area of the table top. Once the plan is created, its progress is monitored as it selects the appropriate

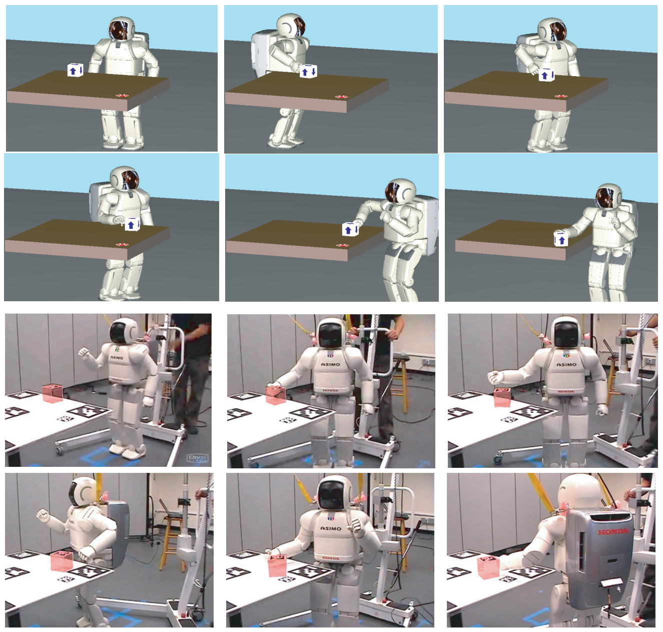

Fig. 3. ASIMO can decide between walking, reaching and pushing modalities to move an object around a table. Top) The robot reconstructs environment from information stored in the Cognitive Map from perceptual modules and can plan in the reconstructed virtual world. Bottom) Once a satisfactory plan has been found, it can be executed on the real robot.

controller for each modality. The controllers use RoboTalk to stream desired joint trajectories to ASIMO to perform the motions.

The second example uses the wine flute detector described in Section III to reconstruct the wine glass pose on a table. A visualization module connects to the Cognitive Map to allow the robot operator to review the robot's estimate of the state. This technique can help operators troubleshoot inconsistencies in the robot's perception model. Once the wine flute location is identified, the Task Matrix uses this information to activate its pointing task to simultaneously aim different body parts at the wine glass location (Figure 2). The head tracks the object with its cameras, while one arm points at the object and the other arm aims at the same object with a camera mounted on its forearm. In this example, the actual robot is substituted for a dynamic simulator without needing to change any of the other system modules due to the abstraction of the RoboTalk motion interface.

VI. Conclusion and Discussion

The use of a well-known humanoid robot like ASIMO for research purposes can be a challenging task because the hardware and software configurations are limited by the need to maintain a consistent physical appearance and by computational and storage capacity. By placing functionality in external modules of a distributed system, we are able to go beyond the computational restrictions of the robot. The Cognitive Map and DIODE mechanisms provide the necessary communication infrastructure to share information between modules and coordinate system activity. The limitations of this approach are the delays of transmission that can occur in networked communication. We have noticed this problem occasionally when sending motion commands to the robot, producing a small pause between the sending of a motion sequence and the actual motion appearing on the robot. The solution to this problem will ultimately involve a combination of using faster networking technology and selective transfer of the implementation of key parts of the motion trajectory generation to the robot, where a tighter control loop resides.

Abstraction is used to facilitate the interoperation of perceptual modules, the Cognitive Map and the Task Matrix. For example, sensor data is processed to produce environmental state estimates which are collected in the Cognitive Map for the Task Matrix to access without concern of the source of that information. It is only through the interaction of these components were we able to execute complex manipulation and planning tasks like pushing objects around a large table using multiple modalities of motion such as walking, reaching and pushing.

Our approach allows different kinds of control architectures to be tested. Currently, we have implemented tasks that follow the sense-plan-act paradigm as well as more reactive activities such as continuously pointing at a moving object. By focusing on the development of useful modules that can be used in multiple control paradigms, we are following a strategy for autonomous robots that allows dynamic changes in its control structure and information model that match the needs of the current task to be performed. For example, the robot can selectively activate the required vision modules it needs to complete the task. This would allow a robot to switch between many modes of behavior in a natural, seamless manner.

References

[1] G. Haynes and A. Rizzi, "Gait regulation and feedback on a robotic climbing hexapod," in Proceedings of Robotics: Science and Ssytems, August 2006.

[2] D. Balkcom and M. Mason, "Introducing robotic origami folding," in IEEE International Conference on Robotics and Automation, April 2004, pp. 3245–3250.

[3] A. Dollar and R. Howe, "A robust compliant grasper via shape deposition manufacturing," in IEEE/ASME Transactions on Mechatronics, vol. 11, no. 2, 2006, pp. 154–161.

[4] Honda Motor Co., Ltd., "Asimo year 2000 model," http://world.honda.com/ASIMO/technoloogy/spec.html, 2000.

[5] R. T. Vaughan, B. P. Gerkey, and A. Howard, "On device abstractions for portable, reusable robot code," in IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2003), 2003, pp. 2121–2427.

[6] B. Hayes-Roth, "A blackboard architecture for control." Artificial Intelligence, vol. 26, pp. 251–321, 1985.

[7] F. Kanehiro, H. Hirukawa, and S. Kajita, "Openhrp: Open architecture humanoid robotics platform." International Journal of Robotics Research, vol. 23, no. 2, pp. 155–165, 2004.

[8] R. T. Vaughan, B. P. Gerkey, and A. Howard, “The player/stage project: Tools for multi-robot and distributed sensor systems.” in Proceedings of IEEE Int. Conference on Advanced Robotics, 2003, pp. 317–323.

[9] K. Thórisson, T. List, C. Pennock, and J. DiPirro, “Whiteboards: Scheduling blackboards for semantic routing of messages and streams,” in AAAI Workshop on Modular Construction of Human-Like Intelligence, July 10 2005, pp. 16–23, pittsburgh, PA.

[10] R. Ierusalimschy, Programming in Lua. Lua.org, 2006.

[11] D. Wagner and D. Schmalstieg, “Artoolkitplus for pose tracking on mobile devices,” in Computer Vision Winter Workshop 2007, February 6-8 2007, st. Lambrecht, Austria.

[12] R. I. Hartley and A. Zisserman, Multiple View Geometry in Computer Vision, 2nd ed. Cambridge University Press, ISBN: 0521540518, 2004.

[13] D. H. Ballard, “Generalizing the hough transform to detect arbitrary shapes.” Pattern Recognition, vol. 13, no. 2, pp. 111–122, 1981.

[14] R. Brooks, “A robust layered control system for a mobile robot.” IEEE Journal of Robotics and Automation, vol. RA-2, April 14-23 1986.

[15] E. Gat, “On three-layer architectures,” in Artificial Intelligence and Mobile Robots, R. P. Bonnasso and R. Murphy, Eds. AAAI Press, 1998.

[16] A. Yang, H. Gonzalez-Baños, V. Ng-Thow-Hing, and J. Davis, “Robotalk: Controlling arms, bases and androids through a single motion interface,” in Proceedings of the 12th International Conference on Advanced Robotics, July 18-20 2005, seattle.

[17] E. Drumwright, V. Ng-Thow-Hing, and M. Mataric, “The task matrix framework for platform-independent humanoid programming,” in Humanoids 2006, 2006, genova, Italy.

[18] K. Hauser, V. Ng-Thow-Hing, and H. Gonzalez-Baños, “Multimodal motion planning for a humanoid robot manipulation task,” in In submission, 2007.