Mobile Manipulation in Unstructured Environments

Sachin Chitta[1] E. Gil Jones[1] Matei Ciocarlie[1] Kaijen Hsiao[1]

***Abstract*—Unstructured human environments present a substantial challenge to effective robotic operation. Mobile manipulation in typical human environments requires dealing with novel unknown objects, cluttered workspaces, and noisy sensor data. We present an approach to mobile manipulation in such environments using a combination of 2D and 3D visual processing, tactile and proprioceptive sensor data, fast motion planning, reactive control and monitoring, and reactive grasping. Our approach allows a robot to perform the full range of operations required for mobile manipulation. We demonstrate our approach by using a two-arm mobile manipulation system to pick and place objects. Our approach attempts to maximize safety and robustness by continuously monitoring the task using visual sensors and replanning or aborting the task if necessary. Furthermore, reactive components attempt to correct the desired plan when presented with unexpected information from the changing environment.**

I. Introduction

Human environments provide incredible challenges for effective mobile manipulation. The remarkable dexterity and mobility of humans results in environments where tasks require a high degree of flexibility in perception, motion, and control that robots currently lack. Large variations in objects, lighting, and clutter make household manipulation a very difficult problem. Most households have very limited space for robots to move around in, and objects are often hidden in clutter, behind other objects, or inside containers. Furthermore, with humans moving around in the environment, safety is an important consideration for any household manipulation robot. Mobile manipulation in such environments will require a tightly-integrated effort, combining techniques in perception, motion planning, grasping, and control.

Significant progress has been made in recent years in advancing the state of the art in mobile manipulation. Mobile manipulation platforms such as the PR2(see Fig. 1), Intel HERB Personal Robot, ICub, and ASIMO, have a combination of high resolution sensors and computation useful for realtime operation in human environments. Realtime 3D sensing using a combination of stereo and laser ranging sensors is often used in such systems to build a consistent representation of their changing environments. However, visual information is often not sufficient for robust operation; robustness can be increased with the addition of tactile and proprioceptive feedback. Safe, collision-free motion can be achieved by combining fast realtime motion planning with smooth reactive controllers.

In this paper, we present an integrated approach to manipulation and grasping in household environments. Our approach starts by building a consistent model of the environment using input from both 3D and 2D visual sensors. The environment is represented using a combination of pre-generated mesh models for known objects and a voxel-based representation for the other parts of the environment. The environment representation serves as input to motion planners that generate collision-free, smooth motions for the robot. Fast realtime trajectory controllers are used to perform the planned trajectories on the robot. Grasp selection for objects consists either of selection of an appropriate grasp from a set of grasps pre-computed offline, or computation of grasps online.

[1] Sachin Chitta, E. Gil Jones, Matei Ciocarlie and Kaijen Hsiao are with Willow Garage Inc., Menlo Park, CA 94025, USA (sachinc,gjones,matei,hsiao)@willowgarage.com

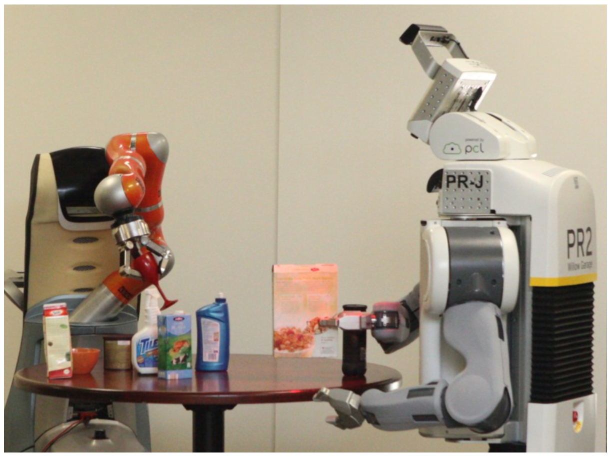


Fig. 1. The PR2 (right) and the Care-O-Bot (left). Both robots are executing a tabletop pick and place task using the approach described in this paper.

The approach presented here builds on earlier work in [1], adding several new components. This includes the ability to track the motion of the manipulator and to react to unforeseen events or obstacles in the environment using visual sensing, and the ability to navigate while carrying larger objects in a cluttered indoor environment. We also describe a new grasp planner that is capable of accounting for uncertainty arising from noisy calibration, segmentation, or recognition. We demonstrate our approach through various tasks performed by the PR2 mobile manipulation system, but also describe new tools that allow many of these capabilities to be quickly adapted for mobile manipulation systems other than the PR2. Our approach allows robots to pick and place a wide range of objects in unstructured environments.

A. Related Work

Mobile manipulation has gained widespread recent interest ([2]–[8]). The Intel HERB mobile manipulation platform has demonstrated impressive capabilities in indoor environments ranging from being able to pick and place objects [8] to push-based manipulation on tabletop environments [9]. The platform has also been used to demonstrate advances in constrained-manipulation planning [10], trajectory optimization based planning [3] and planning under uncertainty [11]. The DLR Justin robot has demonstrated exceptional manipulation capabilities, including the ability to catch balls [12] and make coffee. The ARMAR-III robots have been used for tasks in a prototype kitchen setting demonstrating impressive capabilities including combined grasp and motion planning [13]. Other examples of integrated mobile manipulation systems that have demonstrated autonomous behavior include El-E [14], STAIR [15], Care-O-Bot [16].

Several groups have demonstrated mobile manipulation tasks with the PR2 robot, including laundry tasks [17] and making breakfast [18]. Additional mobile manipulation tasks such as cart pushing [19] and door opening [20], [21] have demonstrated capabilities that require co-ordinated motion of the base and arms of a robot.

Progress in sensing, motion planning, control, grasping, and perception has significantly improved the capabilities of mobile manipulation systems. Fast motion planners now allow robots to react quickly in dynamically changing environments [2]. Reactive controllers can minimize the effects of unknown obstacles or events in the environment [22]. Reactive grasp control [23] allows for correction from errors due to inaccurate or noisy data from visual sensors, calibration errors, or inaccurate pose estimation by object recognition algorithms. Reactive planning frameworks like the elastic roadmap framework [24] allow planning and control to be closely integrated.

B. *Structure of this paper*

This paper is structured as follows: In Section II, we give an overview of our approach and the robot that we use for experiments in Section II. In Section III, we present details on the motion planning, control, and execution components used in our approach. In Section IV, we expand on our approach to grasping. Section V details our efforts to make tools that allow for easy portability of our approach to different mobile manipulation platforms. Finally, Section VI presents experimental results and applications of our approach to different mobile manipulation tasks.

II. Overview

A motivating task that we have used throughout our work is mobile pick and place, i.e. the ability to pick up objects and move them to new locations in an environment. Our aim is to develop an approach to this task that is particularly suited for unstructured environments. Towards this end, our approach makes extensive use of information from the sensors of the robot to model the environment online. We also intend for our approach to be easily deployable on different mobile manipulation systems. To that end, we have developed a software framework (built on ROS [25]) and a set of software tools that make porting our approach to other mobile manipulation systems much easier. Our framework is also modular, with the intention of allowing incorporation of alternate components for motion planning, control, grasping, or perception.

The mobile manipulation platform that we use for validating our approach is the PR2 (Fig. 1), which has an omnidirectional base and two 7-DOF arms. It is equipped with a tilting laser scanner mounted on the body; two stereo cameras and a texture projector are mounted on a pan-tilt platform (the head). An additional laser scanner mounted on the base and a body-mounted IMU are used extensively for localization and navigation. Encoders on each joint provide joint angle information. The end-effector is a parallel jaw gripper whose fingertips are equipped with capacitive sensor arrays, each consisting of 22 individual cells; the gripper also contain an accelerometer. For some of our work, we have also utilized an additional RGBD sensor (a Microsoft Kinect®) mounted on the head of the robot.

Our approach builds on the ROS software framework, utilizing it for all communication and configuration needs. Generic interfaces are defined between different components in our framework as ROS interfaces. Any new component, e.g. a new grasp planning algorithm, can be easily incorporated into the system if it implements the same interface. For example, several groups have been able to integrate their custom motion planners within our framework for execution on the PR2 robot. This includes CHOMP [26], Search-based planners [27], Learning dimensional descent [28], and STOMP [29].

Our overall approach to the pick and place task involves environment modeling using 2D and 3D visual sensors on the robot, deliberative planning using motion planners, reactive control using active sensing, and grasp planning and execution for known and unknown objects while taking uncertainty into account. We will now describe in detail the individual components that make up our framework and the integration required to successfully execute manipulation tasks.

III. Environment Modeling, Motion Planning and Execution

Our approach to motion planning and execution builds on online modeling of the environment, randomized motion planners, and the use of visual sensors for dealing with disturbances. Our focus is on developing a *reliable* framework for moving the robot while taking into account any obstacles that may be present in the environment. We also aim to account for possible disturbances during execution by either pausing and re-executing planned trajectories or replanning when necessary.

A. *Environment Modeling*

Building a consistent and detailed model of the environment is essential for any operation in an unstructured environment. The process starts with raw sensor data from 3D visual sensors (e.g. laser range scanners, stereo cameras, or RGBD sensors) in the form of point clouds. Data from these sensors is first passed through a series of filters. An initial filter attempts to remove spurious data points that can often appear in point cloud data, especially those associated with depth discontinuities in the scene. A second filter uses knowledge of the position of the robot parts to filter out sensor data corresponding to sensor hits on the robot's parts. The resulting point cloud is less noisy

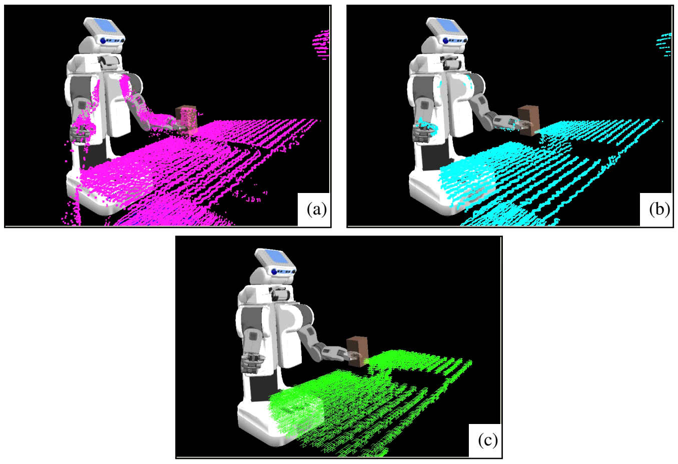


Fig. 2. Raw sensor data (a) is filtered to remove spurious points (b) and points associated with the robot body (c). Note that the object held in the hand is also considered a part of the robot body. (This figure is best viewed in color).

than the original data and includes only data corresponding to the environment. A series of snapshots in Fig. 2 illustrate this process.

The environment model uses two representations: a probabilistic occupancy grid representation for unmodeled or unrecognized parts of the environment and a semantic representation for known parts of the environment. The semantic representation is built using a generic, ROS-based interface to object segmentation and detection algorithms. Our current implementation of these algorithms operates on the filtered 3D point clouds to segment support surfaces (such as tables or shelves) and objects on the support surfaces. A database of pre-built 3D models for common household objects is then used to recognize and register some of the objects in the scene. Mesh models of these objects are used to represent them in the environment model. More details of this process are explained in the section on grasping (Section IV).

Unmodeled parts of the environment are incorporated into an octree-based representation (called Octomap) [30]. Octomap is an occupancy-grid-like probabilistic representation of the environment that can account for *unknown* space, i.e. space that has not been observed by the robot's sensors. It also maintains a persistent view of the environment, automatically incorporating new sensor data and clearing out obstacles that might have moved away. The environment model serves as the primary source of input for collision checking and is used by both the motion planning and grasping components. Fig. 3 illustrates a typical environment generated using this process.

B. Motion Planning and Execution

An overall schematic representation of our motion planning and execution framework used for the robot's arm is shown in Fig. 4. The particular implementation we describe here is intended for use with robotic arms. Motion planning and execution for the base is decoupled from that for the arms, and is described in Section III-C. Future work will involve the development of components for whole-body planning and control.

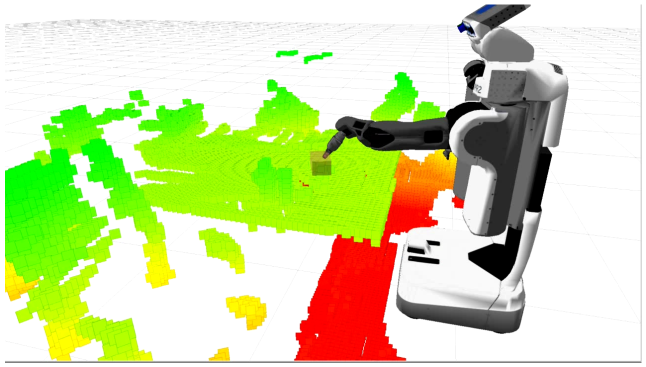


Fig. 3. A typical environment model for a tabletop manipulation task. (This figure is best viewed in color).

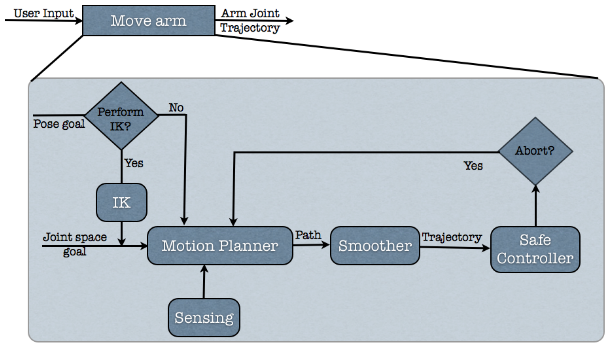


Fig. 4. System architecture for motion planning and execution.

Motion planning for the arms is carried out using randomized motion planners integrated using the OMPL Library [31]. These motion planners can operate on both joint space or end-effector pose goals. Inverse kinematics is used in the latter case to sample joint-space goals that the planner can then plan to. The output of the motion planner is a path that can often be jagged and long. The path is refined and parametrized in time using *shortcutting* techniques designed for smoothing and shortening the path. Our particular implementation of shortcutting uses cubic splines and can account for bounds on joint velocities and accelerations.

Each arm of the PR2 has 7 degrees of freedom and thus has a redundant degree of freedom. We exploit this redundancy by developing a custom inverse kinematics solution for the arm that is parametrized by one of the joint angles. The joint limits on the robot's joints are also taken into account by the kinematics solver. Thus, given the end-effector pose and a value for the free parameter, we can deterministically compute the corresponding inverse kinematics solution for this pose. In general, because of joint limits, we tend to find only a single solution (if it exists) for a given end-effector pose and free angle parameter. If a solution does not exist, it is possible to step through the full range of motion of the redundant joint to *search* for an inverse kinematics solution. For robots other than the PR2, we provide an implementation of a numerical inverse kinematics solver based on the KDL toolchain [32].

Execution of trajectories is handled using a state machine approach that incorporates *active monitoring* of the trajectories executed on the arm. A controller designed to accurately track the desired trajectory is used to execute the planned trajectory

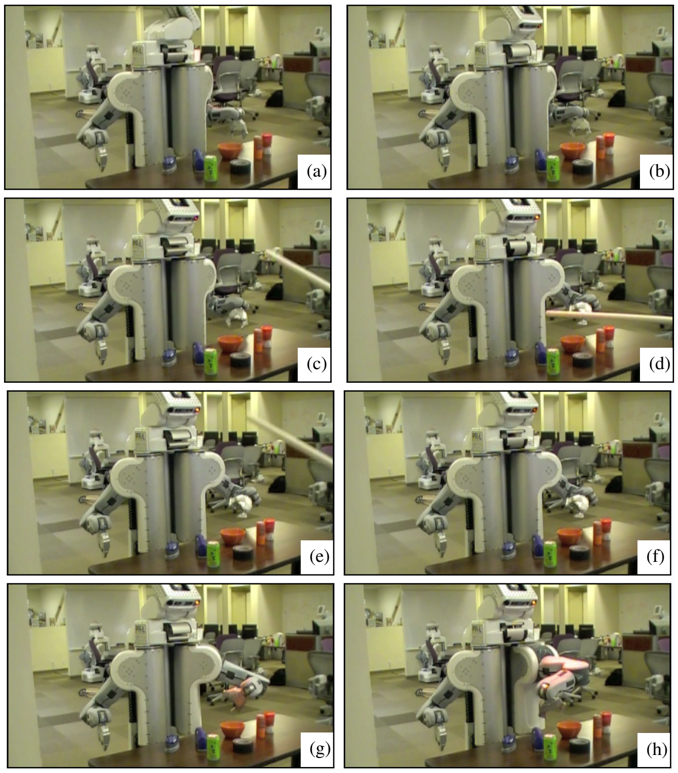


Fig. 5. Active Monitoring: The PR2 starts to execute a trajectory(a). A new obstacle is introduced into the environment(b) and further trajectory execution is aborted in (c) and (d). The obstacle is moved away in (e). The PR2 executes the remaining part of the trajectory once the object has cleared away in (f), (g) and (h).

on the robot. Active monitoring involves moving the head of the robot (on which the sensors are mounted) to maintain visibility of the arm as it is executing the trajectory. If a new obstacle is detected along the expected execution path, the arm is brought to a stop along the path. If the obstacle clears out of the path of the robot within a specified timeout, the trajectory continues to be executed. If the timeout is exceeded, a new path is planned and executed, taking into account the presence of the new obstacle. Fig. 5 shows two snapshots from a scenario where the robot stops when it sees a new obstacle, waits for the obstacle to clear, and then re-executes its planned path [33].

The motion planners can also handle geometric constraints, e.g. constraints on the position and orientation of a grasped object. A constraint that is often specified in manipulation tasks is to keep an object horizontal, e.g. if it contains liquids. Such constraints can be dealt with using projection techniques that attempt to project a random sample onto the constraint manifold [10]. However, we choose to take a simpler approach to such constraint planning by working directly in task space, i.e. in the space of position and orientation of the end-effector of the robot. The planner now functions in a 7 dimensional space: $(x, y, z, roll, pitch, yaw, \phi)$. Here $(x, y, z,)$ represents the position of the end-effector, $(roll, pitch, yaw)$ represents the orientation of the end-effector in a coordinate frame attached to the base of the robot, and ϕ represents the redundant degree of freedom in the 7-DOF PR2 arm. In this space, an orientation constraint on the gripper can be easily accounted for by placing appropriate bounds on the roll, pitch and yaw states of the end-effector, and thus the constraints are easily accounted for in the sampling stage for randomized planners. Fig. 6 shows the result of a constrained plan executed on the robot while carrying out a tabletop manipulation task [33].

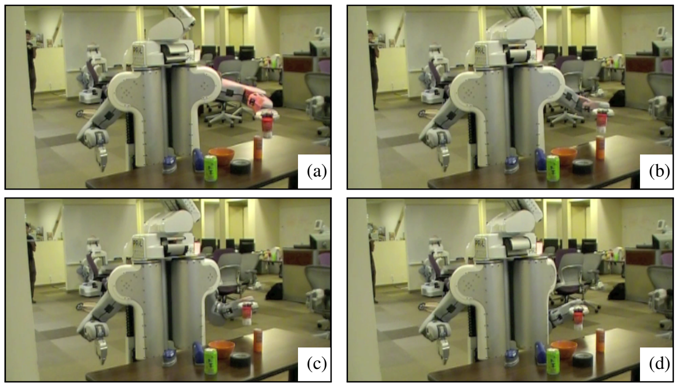


Fig. 6. A constrained motion plan being executed by the PR2 robot. Note that the object is kept close to horizontal throughout the plan.

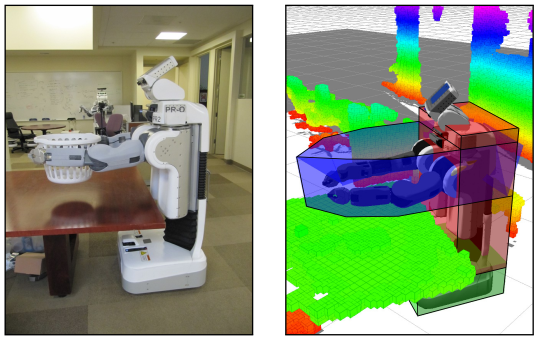


Fig. 7. *Left:* The PR2 robot needs to move its base underneath the table and its arms above to pick up a basket. In a standard 2D navigation approach, the robot's footprint is in collision with a projected 2D costmap, and thus the corresponding planning problem would be unsolvable. *Right:* A multi-layer representation for the PR2 consists of projected layers for the base (green), spine (red), and arms (blue) in addition to a full 3D collision map. This allows for efficient collision checks while considering the 3D structure of the environment and the robot. (This figure is best viewed in color).

C. 3D navigation

True mobile manipulation requires the ability to navigate through a cluttered environment while carrying objects. Traditional navigation approaches typically use a projected 2D footprint and cannot handle, for instance, cases where a robot might have to move its arms over a table while carrying an object (Fig. 7). Environment sensing also becomes harder during such actions, since the object and the arms of the robot occlude the view of the sensors.

In [34], we presented an approach to this problem (which we label *3D navigation*). The approach combines a representation of large-scale environments using Octomap with a hierarchical collision checking scheme integrated with anytime motion planners. The approach used multi-layered 2D costmaps and a layered decomposition of the robot body (Fig. 7) to reduce the number of full-body 3D collision checks required during planning, thus providing motion plans within reasonable times in cluttered environments. The approach was successfully

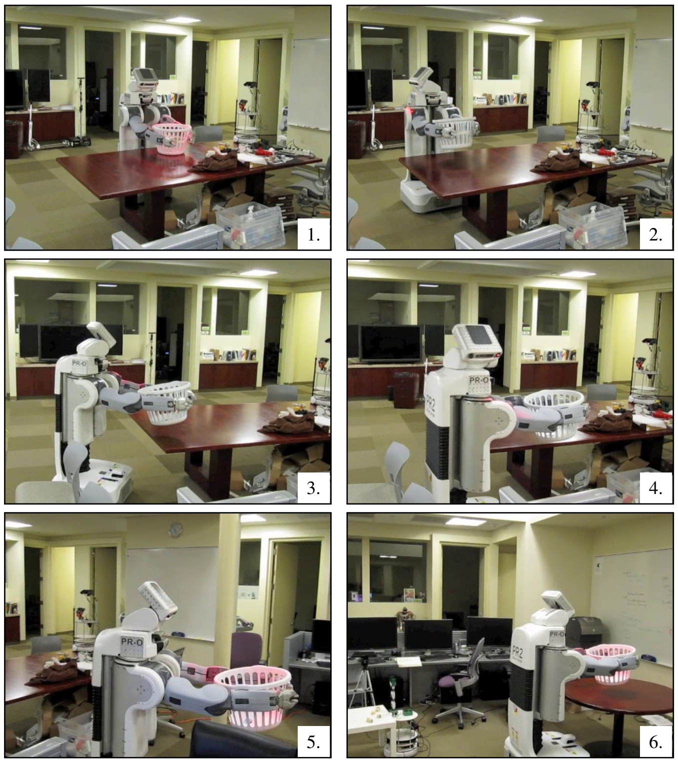


Fig. 8. Sequence of snapshots showing the PR2 robot navigating autonomously with a laundry basket in a cluttered environment.

implemented on the PR2 robot, thus allowing it to navigate with a large object(a laundry basket) in a cluttered environment (Fig. 8). Note that our current implementation of 3D navigation does not account for (or plan for) the arms moving while the base is also in motion. It also does not plan the grasps for large objects like the laundry basket. Future work is intended for developing whole-body grasping capabilities.

IV. Grasping

We define grasping as the ability to secure an object inside the robot's end-effector while resisting external disturbances. It is a key prerequisite for enabling a large number of mobile manipulation tasks, such as object transport and retrieval, tool use, *etc*. Robotic grasping is common in industrial environments, where pick-and-place robots are able to execute grasps with a high degree of reliability. However, this level of performance is usually achieved by taking advantage of the structure inherent in industrial settings: a small and well-defined set of objects to be manipulated, which allows for the design of dedicated end-effectors for known object poses. In environments lacking this kind of structure, such as typical homes or offices, robotic grasping is still an open area of research.

We believe two of the key problems that must be addressed for robust grasping in unstructured environments are *variability*, requiring the ability to handle a very large set of possible objects and scenarios, and *uncertainty*, requiring the ability to cope with errors affecting many levels of the system from sensing and scene interpretation to mechanism calibration. In this section, we describe our initial work on grasp planning and execution for a wide range of typical household objects, followed by the methods we developed in order to improve their handling of uncertainty and hence the overall reliability of the system.

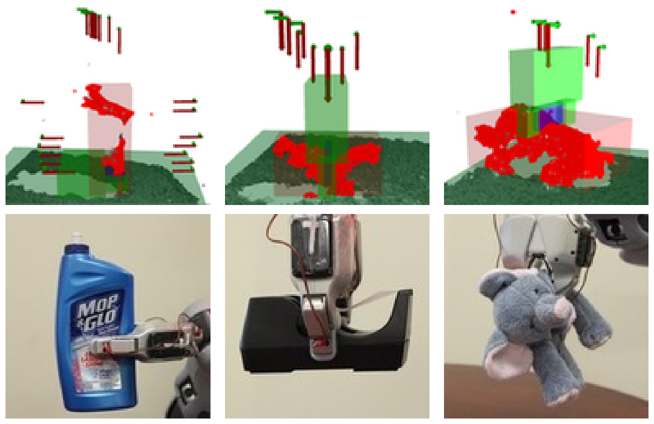


Fig. 9. Cluster-based grasp planning. Top row: individual object point clouds (red points) and grasps (red arrows) associated with them. Bottom row: execution of planned grasps.

A. *Grasp Planning and Execution*

The ability to plan a grasp for an object depends on the nature and quality of information available to the grasp planner. In environments such as households or offices, the grasp planner receives this information from a perception module that attempts to parse and label the scene. While scene interpretation algorithms are constantly improving, they still can not achieve perfect reliability in general human settings. We believe that a robot must be equipped to both use high-level perception results when possible, and cope with situations when less information is available.

1) Cluster-based Planner: The first grasp planning algorithm we present operates on individual point clouds of graspable objects. It requires that a point cloud from a depth camera, such as the Microsoft Kinect® or the PR2 stereo pairs, be separated in individual "clusters", each belonging to one target object.

Our reference implementation of the perception module associated with this planner makes a set of assumptions exploiting the fact that household objects are typically found on flat surfaces such as table or counter tops. We compute a planar fit to the point cloud to identify the support surface, and then use Euclidean clustering on the points above it to obtain individual object clusters. This perception mechanism is not equipped to handle cluttered scenes; however, more advanced algorithms that also use color information in order to segment cluttered or stacked objects, such as the one presented by Bjorkman and Kragic [35], can also be used as input to our planner.

Once an object's point cloud has been segmented, grasps are computed using heuristics based on both the overall shape of the object and its local features. The intuition behind this approach is that many human-designed objects can be grasped by aligning the hand with the object's principal axes, starting from either above or to the side of the object, and searching for parts of the object that fit inside the hand; if no such grasps are found, then grasps from above of high points on the object

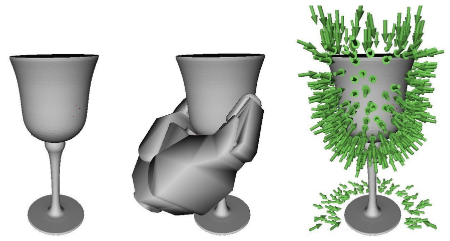


Fig. 10. Grasp planning in a simulated environment for a known object (the object model, a simulated grasp and the complete set of pre-computed grasps).

are also likely to be useful. Grasps found according to these principles are then ranked using a small set of simple feature weights, including the number of sensed object points that fit inside the gripper, distance from object center, *etc*. A number of examples are shown in Fig. 9, and additional information about this component can be found in [23].

2) Recognition-based Database Planner: If an object recognition component is present as part of the robot's perception toolset, additional information regarding a recognized model can be pulled from a database of known objects. Complementing the methods and algorithms discussed here, we have introduced a database of common household objects available from major U.S. retailers. For each object, the database contains a triangular surface mesh, as well as additional tags such as maker and model, barcode, class labels, *etc*. This dataset, described in detail in [36], is available as a relational database using the SQL standard.

When the target is a recognized database object, the robot can plan grasps using the complete object mesh. Furthermore, grasps for each object can be pre-computed in advanced and also stored in the database. For each object in the database, we used the *GraspIt!* simulator [37] to pre-compute a large number of grasp points for the PR2 gripper. We used a grasp quality function requiring both finger pads to be aligned with the surface of the object (finger pad surfaces contacting with parallel normal vectors) and further rewarding postures where the palm of the gripper is close to the object as well. However, the grasp planning process outlined here for the PR2 gripper can be extended to other robot hands as well. For more dexterous models, a different grasp quality metric can be used, taking into account multifingered grasps, such as metrics based on the Grasp Wrench Space.

Our grasp planning tool used a simulated annealing optimization, performed in simulation, to search for gripper poses relative to the object with the value of the grasp quality function above a given threshold. For each object, this optimization was allowed to run over 4 hours, and all the grasps satisfying our requirements were saved in the database; an example of this process is shown in Fig. 10. This process resulted in an average of 600 pre-computed grasp points for each object.

We use the database of known objects and grasps in conjunction with an object recognition module that attempts to match each object point cluster, segmented as described in the previous section, to each of the meshes in a pre-defined set of objects. Matching is performed using an iterative technique similar to the ICP algorithm [38], and the best fit is returned. However, other object recognition techniques can be used as

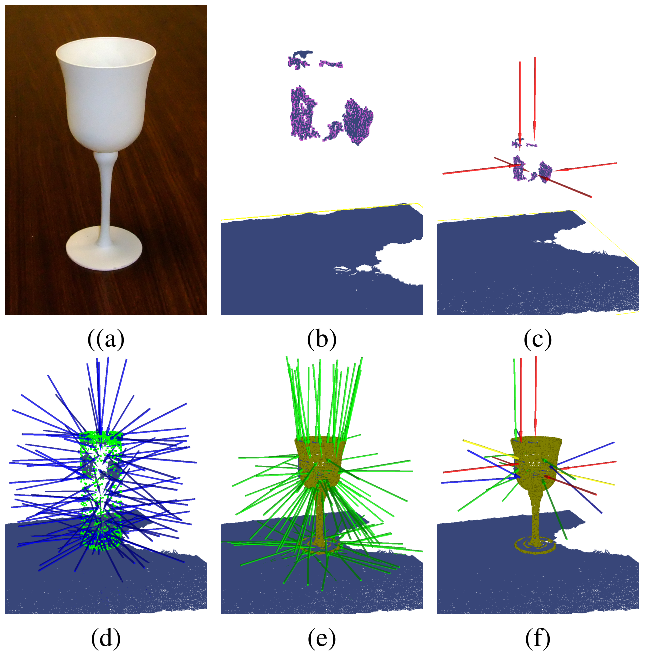


Fig. 11. Combining multiple grasp planners. **(a)** A cup to be grasped, and the point cloud as seen by the stereo camera **(b)**. **(c)-(e)** Different object representations, and grasps planned for each. These include grasps planned on the segmented point cluster (c), and on meshes for results from object recognizers, one incorrect shown in (d) and one correct shown in (e). **(f)** A set of grasps that combines information from all of them.

well. Recent methods, such as the MOPED framework [39], have shown the ability to recognize objects even when stacked closely together, and could allow the robot to perform grasps in highly cluttered scenes.

3) Grasp Execution: The grasps generated by either of the two planners discussed above consist of end-effector poses relative to the target object. After planning, a collision-aware Inverse Kinematics module checks each grasp for feasibility in the current collision environment, constructed as described in the previous section. Once a grasp is deemed feasible, the motion planner generates an arm trajectory for achieving the grasp position (as described earlier in Section III), and the grasp is executed.

B. Coping with Uncertainty

The methods described in the previous section can be used to grasp a wide range of common household objects, taking advantage of high-level information (such as object identity) when it is available but also handling novel objects when needed. In practice, however, we also found them to be sensitive to errors in either the perception or execution modules, where an object was misrecognized, segmentation was incorrect, or calibration between the sensors and the end-effector was imperfect.

1) Bayesian Grasp Planning: One method of dealing with uncertainty in object shape or pose due to such errors is to combine information from multiple object recognizers, grasp planners, and/or grasp evaluators. Fig. 11 illustrates the general concept: when trying to grasp the cup shown in (a), the robot's stereo cameras see only the point cloud shown in (b). The robot's object recognizers provide two hypotheses, with corresponding quality scores: the tennis ball can (incorrect) shown in (d) and the cup (correct) shown in (e). Grasps can be planned on all possible hypotheses, including different object shapes and poses (using, for instance, GraspIt!), and even just the segmented point cluster (using, for instance, the cluster grasp planner described in section IV-A).

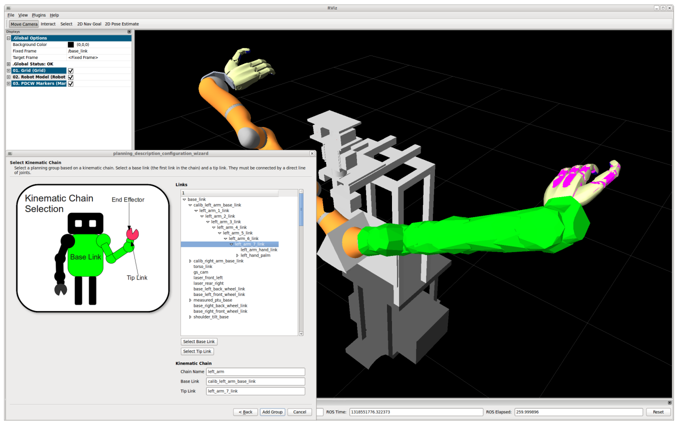
The entire pool of grasps, generated on all available hypotheses, can be evaluated using any number of available grasp evaluators. In this case, the grasp evaluators used are the same ones used to generate the grasps: *GraspIt!* can be used to evaluate arbitrary grasps on each possible object mesh/pose, and the cluster grasp planner can be used to evaluate arbitrary grasps based just on the segmented cluster. By combining both object recognition results and grasp evaluation results, we can estimate the probability of success for each grasp in the generated pool of grasps, and thus select only those grasps most likely to succeed.

Although each object recognizer and grasp evaluator's quality metrics are arbitrary, previous data with ground-truth-labeled recognition attempts/grasps (in real-life and/or in simulation) and their corresponding quality values can be used to model the conditional probabilities of obtaining various quality values given successful or unsuccessful recognition/grasping. Such probabilities allow us to combine results from multiple object recognizers and multiple grasp evaluators in a principled way. More details are available in [40].

2) Reactive Grasp Execution: Grasping objects based on visual information can be affected by errors in object localization, perceived shape, or calibration between the robot's cameras and its end-effectors. We can compensate for such errors by using tactile information acquired during the grasping process. The PR2 gripper is equipped fingertip with capacitive sensor arrays, each consisting of 22 individual cells distributed on the tips, sides and inner pads. During the final stages of the grasp, we use a set of "reactive behaviors" based on information from these sensors to adapt the grasp to unexpected contacts.

The first reactive behavior attempts to maneuver the gripper around the object when unexpected contacts are detected by the fingertips during the approach; this is done by backing up, moving in the direction of the observed contact, and then attempting to continue on to a shifted grasp goal. The second behavior accounts for cases where one fingertip comes into contact with the object before the other by executing a *compliant grasp* that coordinates the motion of the arm and the gripper so that the object is not pushed out of the grasp while the gripper is closed. The final behavior adjusts grasps that are likely to be unstable, as judged by seeing contacts only at the fingertips or only on one side of the fingertip sensor arrays, by attempting to shift the the end-effector position until contacts are seen at the centers of the fingertip sensor arrays. Although these behaviors are simple and deal with objects in a model-free way, they are successful in fixing many minor errors that would have caused the grasp to fail. More details on the reactive grasping behaviors can be found in [23].

3) Adaptive Grasp Force Control: A further issue when grasping and transporting objects is the need to regulate the internal forces applied to the object: too much force, and



Select Kinematic Chain
Select a planning group based on a kinematic chain. Select a base link (the first link in the chain) and a tip link. They must be connected by a direct line of joints.

Kinematic Chain
Chain Name: left_arm
Base Link: calib_left_arm_base_link
Tip Link: left_arm_7_link

Fig. 12. The Arm Navigation Wizard window being used to set up a kinematic chain group for the TUM-Rosie robot. The Wizard window is on the left, with the Rviz visualizer behind it. The links that belong to the group are shown in green, and the links further down on the kinematic chain are shown in magenta.

deformable objects will be crushed and potentially damaged; too little force, and objects will be dropped. A control scheme that uses the PR2's tactile sensors and accelerometers for regulating grasp forces is presented in [41]. Real-time data from the gripper's fingertip tactile arrays and accelerometers are combined into signals designed to mimic three different human skin sensors (SA-I, FA-I, and FA-II). These signals are then used in event-based, force-regulating controllers that enable the PR2 to grasp soft objects without crushing them, to grasp harder when an object is slipping out of the hand, and to place objects down when the object hits the table, rather than dropping it above the table or attempting to push the object through the table.

V. Portability to Other Manipulation Platforms

Our algorithms and implementations for collision-free arm motion, grasping, and manipulation are currently in use on PR2s all around the world; however, to maximize the impact of our work we have also sought to make it straightforward to use our work on any manipulator. The vision of this work is to allow a user to take the physical description of his/her robot – in URDF or COLLADA format – and quickly and interactively configure a system that will allow them to manipulate objects in a simulated or physical environment.

For the first stage of this work we focused on allowing new users to configure our software for generating collision-free arm trajectories interactively using a tool called the Arm Navigation Wizard. This tool requires the physical specification of the robot and some user input; it produces all necessary configuration and application files for collision-free arm navigation, including those for self- and environment collision checking, joint- and task-space planning, and inverse kinematics.

The physical specification of the robot is assumed to contain strictly mechanical properties of the robot: joint locations and limits, and link locations and geometry. It lacks some first-order semantic information about which joints and links constitute the robot's arm or arms, and end effector or end effectors. We require that users of the Arm Navigation Wizard configure at least one group that can be used for planning. For most users this group will consist of a kinematic chain corresponding to a robot arm. The tool makes it easy to navigate a robot's kinematic tree to determine the base and tip links of the kinematic chain, and upon creation the chain is rendered graphically. Configuring groups for a robot's arm or arms is the only required interaction in the Wizard.

The primary difficulty in configuring our motion planning and execution system for a new robot involves configuring the self-collision checking capabilities. Most components in our system depend on the ability to check whether a particular configuration of the robot's links results in collision with the environment or its own body - for most robots any contact can result in damage or other undesirable outcomes. Environment collision checking is straightforward as any contact is forbidden, but self-collision checking is more nuanced. Robots invariably have links that collide with each other in all possible configurations of the robot. Links that are adjacent in the kinematic chain will frequently fall into this category, as they will intersect at the point of their shared joint. Collisions between such links must not register as a self-collision or all potential robot configurations will register as unsafe. For the sake of efficiency it is also important to identify link pairs that can never collide given the robot specification and joint limits, as checking these pairs for collision can make self-collision checking more time-consuming.

By default our auto-generation software disables all collisions between links that are adjacent in the kinematic tree. In order to determine the sets of link pairs that are always or never in collision we use a joint-space sampling strategy. We generate a joint space sample by uniformly sampling a value for each joint between the joint limits. We then invoke an exhaustive collision check that returns all pairs of links that are in collision. By taking many samples we can determine which sets of links are in collision in every sample and which are never in collision over all samples and disable collision-checking for these sets. Precisely determining which link pairs will never be in collision is a difficult problem; future work will involve dynamically focusing the samples on areas of the joint space that produce more information.

The Arm Navigation Wizard allows users to get arm navigation working on their robots in minutes or hours, whereas it previously may have taken months. With this capability, we were able to port the manipulation capabilities described in Sections III and IV to the Care-O-Bot [16] (along with two researchers from Fraunhofer IPA). The process, from generating configuration files using the Wizard to executing trajectories on the Care-O-Bot, took only a few hours. Porting our grasping and manipulation capabilities took more time, but the end result was that both robots could essentially run the same code despite having different sensing hardware, arms, and end-effectors, as shown in Fig. 1. Future work involves developing cross-robot benchmarking suites for motion planning and extending the Wizard to grasping and manipulation.

VI. Applications

A. *From Our Group*

- pick-and-place demo
- cart pushing
- beer bot

B. *From External Groups*

- MIT bake bot
- Bosch remote lab
- Ben Cohen
- Paul Vernaza

VII. Conclusions

Mobile Manipulation rocks!!!

VIII. Acknowledgments

References

[1] M. Ciocarlie, K. Hsiao, E. G. Jones, S. Chitta, R. B. Rusu, and I. A. Sucan, "Towards reliable grasping and manipulation in household environments," in *ISER*, New Delhi, India, December 2010.

[2] R. B. Rusu, I. A. Şucan, B. Gerkey, S. Chitta, M. Beetz, and L. E. Kavraki, "Real-time perception guided motion planning for a personal robot," in *International Conference on Intelligent Robots and Systems*, St. Louis, USA, October 2009.

[3] N. Ratliff, M. Zucker, J. A. Bagnell, and S. Srinivasa, "CHOMP: Gradient optimization techniques for efficient motion planning," in *IEEE International Conference on Robotics and Automation*, 12–17 May 2009, pp. 489–494.

[4] R. Diankov, N. Ratliff, D. Ferguson, S. Srinivasa, and J. J. Kuffner., "Bispace planning: Concurrent multi-space exploration," in *Robotics: Science and Systems*, Zurich, Switzerland 2008.

[5] D. Berenson, S. S. Srinivasa, D. Ferguson, A. Collet, and J. J. Kuffner, "Manipulation planning with workspace goal regions," in *IEEE International Conference on Robotics and Automation*, May 2009.

[6] D. Katz, E. Horrell, Y. Yang, B. Burns, T. Buckley, A. Grishkan, V. Zhylkovskyy, O. Brock, and E. Learned-Miller, "The UMass Mobile Manipulator UMan: An Experimental Platform for Autonomous Mobile Manipulation," in *IEEE Workshop on Manipulation for Human Environments*, Philadelphia, USA, August 2006.

[7] C. Borst, C. Ott, T. Wimbock, B. Brunner, F. Zacharias, B. Baeum, U. Hillenbrand, S. Haddadin, A. Albu-Schaeffer, and G. Hirzinger, "A humanoid upper body system for two-handed manipulation," in *IEEE International Conference on Robotics and Automation*, April 2007, pp. 2766–2767.

[8] S. Srinivasa, D. Ferguson, M. V. Weghe, R. Diankov, D. Berenson, C. Helfrich, and H. Strasdat, "The Robotic Busboy: Steps Towards Developing a Mobile Robotic Home Assistant," in *Intl. Conference on Intelligent Autonomous Systems (IAS-10)*, July 2008.

[9] M. Dogar and S. Srinivasa, "Push-grasping with dexterous hands: Mechanics and a method," in *Proceedings of 2010 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2010)*, October 2010.

[10] D. Berenson, S. S. Srinivasa, D. Ferguson, and J. J. Kuffner, "Manipulation planning on constraint manifolds," in *IEEE International Conference on Robotics and Automation*, May 2009.

[11] D. Berenson, T. Simeon, and S. Srinivasa, "Addressing cost-space chasms in manipulation planning," in *IEEE International Conference on Robotics and Automation (ICRA '11)*, May 2011.

[12] B. Bauml, T. Wimbock, and G. Hirzinger, "Kinematically optimal catching a flying ball with a hand-arm-system," in *International Conference on Intelligent Robots and Systems*, 18–22 Oct 2010, pp. 2592–2599.

[13] N. Vahrenkamp, M. Do, T. Asfour, and R. Dillmann, "Integrated grasp and motion planning," in *IEEE International Conference on Robotics and Automation*, 2010, pp. 2883–2888.

[14] A. Jain and C. Kemp, "EL-E: an assistive mobile manipulator that autonomously fetches objects from flat surfaces," *Autonomous Robots*, 2010.

[15] A. Saxena, L. Wong, and A. Ng, "Learning grasp strategies with partial shape information," in *AAAI Conf. on Artificial Intelligence*, 2008.

[16] C. Partlitz, M. Hagele, P. Klein, J. Seifert, and K. Dautenhahn, “Care-o-bot 3 - rationale for human-robot interaction design,” in *Intl. Symposium on Robotics*, 2008.

[17] J. Maitin-Shepard, M. Cusumano-Towner, J. Lei, and P. Abbeel, “Cloth grasp point detection based on multiple-view geometric cues with application to robotic towel folding,” in *IEEE Intl. Conf. on Robotics and Automation*, 2010.

[18] M. Beetz, U. Klank, I. Kresse, A. Maldonado, L. Mösenlechner, D. Pangercic, T. Rühr, and M. Tenorth, “Robotic roommates making pancakes,” in *11th IEEE-RAS International Conference on Humanoid Robots*, Bled, Slovenia, October, 26–28 2011.

[19] J. Scholz, S. Chitta, B. Marthi, and M. Likhachev, “Cart pushing with a mobile manipulation system: Towards navigation with moveable objects,” in *International Conference on Robotics and Automation*, Shanghai, China, 05/2011 2011.

[20] S. Chitta, B. Cohen, and M. Likhachev, “Planning for autonomous door opening with a mobile manipulator,” in *ICRA*, Anchorage, Alaska, 2010.

[21] W. Meeussen, M. Wise, S. Glaser, S. Chitta, C. McGann, P. Mihelich, E. Marder-Eppstein, M. Muja, V. Eruhimov, T. Foote, J. Hsu, R. B. Rusu, B. Marthi, G. Bradski, K. Konolige, B. P. Gerkey, and E. Berger, “Autonomous door opening and plugging in with a personal robot,” in *ICRA*, 2010.

[22] S. Haddadin, A. Albu-Schffer, A. D. Luca, and G. Hirzinger, “Collision detection and reaction: A contribution to safe physical human-robot interaction,” in *IROS*, Nice, France, 2008, pp. 3356–3363.

[23] K. Hsiao, S. Chitta, M. Ciocarlie, and E. G. Jones, “Contact-reactive grasping of objects with partial shape information,” in *IROS*, 10 2010.

[24] Y. Yang and O. Brock, “Elastic roadmaps: Motion generation for autonomous mobile manipulation,” *International Journal of Robotics Research*, vol. 28, pp. 113–130, 2010.

[25] M. Quigley, K. Conley, B. P. Gerkey, J. Faust, T. Foote, J. Leibs, R. Wheeler, and A. Y. Ng, “Ros: an open-source robot operating system,” in *ICRA Workshop on Open Source Software*, 2009.

[26] I. A. Sucan, M. Kalakrishnan, and S. Chitta, “Combining planning techniques for manipulation using realtime perception,” in *International Conference on Robotics and Automation*, 2010, pp. 2895–2901.

[27] B. Cohen, G. Subramanian, S. Chitta, and M. Likhachev, “Planning for manipulation with adaptive motion primitives,” in *ICRA*, Shanghai, China, 2011.

[28] P. Vernaza and D. D. Lee, “Learning dimensional descent planning for a highly-articulated robot arm,” in *Proceedings of the International Conference on Intelligent Robots and Systems*, 2011. [Online]. Available: http://www.seas.upenn.edu/~vernaza/papers/lddPR2.pdf

[29] M. Kalakrishnan, S. Chitta, E. Theodorou, P. Pastor, and S. Schaal, “STOMP: Stochastic trajectory optimization for motion planning,” in *International Conference on Robotics and Automation*, Shanghai, China, May 2011.

[30] K. M. Wurm, A. Hornung, M. Bennewitz, C. Stachniss, and W. Burgard, “OctoMap: A probabilistic, flexible, and compact 3D map representation for robotic systems,” in *Proc. of the ICRA 2010 Workshop on Best Practice in 3D Perception and Modeling for Mobile Manipulation*, Anchorage, AK, USA, May 2010, software available at http://octomap.sf.net/. [Online]. Available: http://octomap.sf.net/

[31] “The Open Motion Planning Library (OMPL),” http://ompl.kavrakilab.org, 2010.

[32] “The Kinematics Dynamics Library (KDL),” http://www.orocos.org/kdl.

[33] “Constrained planning and active monitoring on the pr2 robot,” http://vault.willowgarage.com/wgdata1/vol1/ram_2011_mobile_manipulation/planning_and_execution.m4v, 2011.

[34] A. Hornung, E. Jones, S. Chitta, M. Bennewitz, M. Phillips, and M. Likhachev, “Towards navigation in three-dimensional cluttered environments,” in *The PR2 Workshop: Results, Challenges and Lessons Learned in Advancing Robots with a Common Platform, IROS*, 2011. [Online]. Available: http://vault.willowgarage.com/wgdata1/vol1/iros_2011_pr2_workshop/hornung_freiburg.pdf

[35] M. Bjrkman and D. Kragic, “Active 3d scene segmentation and detection of unknown objects,” in *International Conference on Robotics and Automation*, 2010.

[36] M. Ciocarlie, C. Pantofaru, K. Hsiao, G. Bradski, P. Brook, and E. Dreyfuss, “A side of data with my robot: Three datasets for mobile manipulation in human environments,” *IEEE Rob. and Autom. Mag.*, vol. 18, no. 2, 2011.

[37] A. Miller and P. K. Allen, “GraspIt!: A Versatile Simulator for Robotic Grasping,” *IEEE Rob. and Autom. Mag.*, vol. 11, no. 4, 2004.

[38] P. J. Besl and M. I. Warren, “A Method for Registration of 3-D Shapes,” *IEEE Trans. on Pattern Analysis*, vol. 14, no. 2, pp. 239–256, 1992.

[39] A. Collet, M. Martinez, and S. S. Srinivasa, “The MOPED framework: Object Recognition and Pose Estimation for Manipulation,” 2011.

[40] K. Hsiao, M. Ciocarlie, and P. Brook, “Bayesian grasp planning,” in *ICRA 2011 Workshop on Mobile Manipulation: Integrating Perception and Manipulation*, 2011.

[41] J. Romano, K. Hsiao, G. Niemeyer, S. Chitta, and K. J. Kuchenbecker, “Human-inspired robotic grasp control with tactile sensing,” *IEEE Transactions on Robotics*, In Press.