Auton Robot (2014) 36:11–30
DOI 10.1007/s10514-013-9365-9

An autonomous manipulation system based on force control and optimization

Ludovic Righetti · Mrinal Kalakrishnan · Peter Pastor · Jonathan Binney · Jonathan Kelly · Randolph C. Voorhies · Gaurav S. Sukhatme · Stefan Schaal

Received: 2 March 2013 / Accepted: 17 August 2013 / Published online: 7 September 2013

Abstract In this paper we present an architecture for autonomous manipulation. Our approach is based on the belief that contact interactions during manipulation should be exploited to improve dexterity and that optimizing motion plans is useful to create more robust and repeatable manipulation behaviors. We therefore propose an architecture where state of the art force/torque control and optimization-based motion planning are the core components of the system. We give a detailed description of the modules that constitute the complete system and discuss the challenges inherent to creating such a system. We present experimental results for several grasping and manipulation tasks to demonstrate the performance and robustness of our approach.

Keywords Grasping · Manipulation · Force control · Optimization

Electronic supplementary material The online version of this article (doi:10.1007/s10514-013-9365-9) contains supplementary material, which is available to authorized users.

L. Righetti (✉) · M. Kalakrishnan · P. Pastor · J. Binney · R. Voorhies · G. S. Sukhatme · S. Schaal
Computer Science Department, University of Southern California, Los Angeles, CA 90089, USA
e-mail: ludovic.righetti@tuebingen.mpg.de

J. Kelly
Institute for Aerospace Studies, University of Toronto, Toronto, ON M3H 5T6, Canada

L. Righetti · S. Schaal
Autonomous Motion Department, Max-Planck-Institute for Intelligent Systems, Paul-Ehrlich-Str. 15, 72076 Tübingen, Germany

1 Introduction

Autonomous robotic manipulation remains an open and very difficult problem. Among other reasons, this problem is difficult because it encompasses multiple areas of research, including computer vision or more generally perception to recognize the environment, higher order reasoning to synthesize sequences of actions to efficiently solve a complex task, motion planning to generate desired movements, control to execute these movements, machine learning to automatically acquire new skills and improve them, and automatic calibration to ensure precision. To add to the difficulty, we argue that is it not sufficient to address each research domain independently as each of these elements need to constantly interact with each other as they are integrated into a single system. Therefore the design of a complete system, where these different elements can efficiently work together, is another challenging aspect of the problem.

Autonomous robotic manipulation is by definition a problem that involves constant interaction with objects. Robots are supposed to frequently touch the world which creates new challenges for traditional approaches such as collision free motion planning. From that point of view, it seems natural to develop algorithms that will inherently be in phase with the nature of manipulation: controlling the contact interactions with objects. Moreover, it seems desirable to adopt approaches that can optimize arbitrary additional objectives such as trajectory smoothness, obstacle clearance, torque minimization and manipulability. The resulting motion plans will improve task performance, robustness and repeatability while providing a meaningful way to exploit kinematic redundancy.

In this contribution, we report our recent effort in developing a complete system for grasping and manipulation. Two important concepts lie at the core of our approach: (1) we

make extensive use of torque and force controllers that provide very high performance while greatly simplifying the interaction with the environment, (2) we rely on optimization methods for motion planning in order to optimize important objectives associated with manipulation tasks. Our approach results in a robot that can safely and efficiently interact with the environment during manipulation tasks by explicitly controlling interactions forces. For example, when grasping an object on the table, contact with the table can be exploited to create robust grasps. Grasping an object is achieved by controlling interaction forces between the object and the fingers, ensuring that the fingers adapt to the shape of the object. Simple grasp planning that does not require precise grasp point computations can then be used to grasp a wide range of objects. The controller also allows to adapt the stiffness of the robot in directions relevant to the task, allowing precise positioning when necessary (e.g. to put a key in a lock). Optimized motions lead to more repeatable and robust behaviors that guarantee the execution of long and complicated manipulation tasks such as drilling. And collision-free motion planning is then only used outside of the manipulation area. Building on these ideas, we propose a complete manipulation architecture. We also describe the challenges faced in integrating our different modules into a coherent system. Experimental results with several different grasping and manipulation tasks are shown in order to validate the approach. Experiments were conducted by an independent test team during the first phase of the DARPA Autonomous Robotic Manipulation (ARM) program. These experiments therefore offer an objective evaluation of our approach since we did not have any control over the evaluation. Moreover, it allows to compare our approach with the different architectures that were developed by the other teams participating in the program and that are briefly described in the next section. Finally, we discuss the benefits and shortcomings of our approach. Based on this discussion, we sketch what could be a partial research program towards truly autonomous robotic manipulation.

In the following, we first situate our work with respect to the current literature and in the context of the ARM project (Sect. 2). Then, in Sect. 3 we give an overview of the general system architecture and describe in detail the important components of the architecture. Experimental results are presented in Sect. 4, followed by a discussion about the strengths and weaknesses of our approach and future research directions to improve our system (Sect. 5).

2 Background

In the following we describe previous work and explain how it relates or differs from our approach. Then we describe the ARM project in which our research took place and briefly describe the robot we used to develop our manipulation system.

2.1 Related work

This section will cover related work in the areas of calibration, force control, grasping and manipulation, and integrated systems. Our focus will be on related work that has been evaluated on real systems in similar settings.

Calibration: A lot of research has been conducted to calibrate the kinematic parameters of robotic manipulators, see Hollerbach et al. (2008) for an overview. Fast and robust ways of calibrating for kinematic quantities are presented in Pradeep et al. (2010) and Hubert et al. (2012). The approach in Pradeep et al. (2010) is inspired by the bundle adjustment approach, and generalized to estimate robot system parameters by including measurements from various types of sensors. The approach presented in Hubert et al. (2012) can make use of prior knowledge of the system. To also achieve high accuracy even in the presence of non-geometric errors, e.g. due to gear transmission, friction, temperature, and compliance, research has been conducted towards also modeling these quantities and also optimizing those parameters, see Majarena et al. (2010) for an overview. However, accurate modeling of the non-linearities present in the ARM-S robot, for example due to cable stretch, is tedious and potentially unfeasible. An approach to account for such non-linearities on the ARM-S robot is presented in Axelrod and Huang (2012). The approach computes a position offset for the arm using k-nearest neighbors for different parts of the workspace. Orientation corrections are only computed for horizontal and vertical endeffector orientations. Furthermore, the approach cannot account for errors due to cable stretch given that two similar end-effector poses can have two very different pose errors depending on the configuration of the arm, the particular reaching trajectory as well as its execution speed.

Force control: Force control has been thoroughly investigated these past three decades. Typically two types of approaches are distinguished. Impedance control, as originally proposed in Hogan (1985), aims to control the mechanical impedance of the end-effector during contact interactions. It is a generalization of approaches that merely try to control the stiffness of the robot. Another approach, called hybrid control (Raibert and Craig 1981) aims at directly controlling forces in the direction of the interaction while controlling position in the other directions. Several extensions and generalizations of these methods have been proposed (see for example Yoshikawa (2000) for a review and Chiaverini et al. (1999) for experimental comparisons). Recent work on creating compliant motions for manipulation have focused on joint trajectory generation (Kazemi et al. 2012) to achieve desired contact forces, which is similar to admittance control (Whitney 1977). The approach has inherently a slow bandwidth

and requires explicit algorithms to modify trajectories with condition loops and thresholds. In contrast, our approach is essentially based on a hybrid approach, where the dynamics of the robot is compensated for using inverse dynamics and where a feedback control scheme similar to the parallel approach of Chiaverini and Sciavicco (1993) is used. A task-frame formalism using ideas similar to Bruyninckx and Schutter (1996) is implemented to specify which directions need to be position or force controlled and gains are scheduled in order to vary control directions and associated stiffness. It results in simple controllers with few parameters and very high control bandwidth allowing fast motion and reactivity.

Grasping: There has been a lot of research on planning good grasps, see for example Bicchi et al. (2000) and Bohg et al. (2013) for reviews. Several approaches are based on finding good grasp points (i.e. precise hand pose and locations of the fingers on the object) such as Miller and Allen (2004) and more recently model-free algorithms were able to generate good grasps for unknown objects (Herzog et al. 2013; Hsiao et al. 2010). These are in close relation to data driven approaches that have also been proposed to infer grasps from real-world data (Bohg and Kragic 2010; Li and Pollard 2005). In our approach, we rely on object models and build a database of grasp poses through human demonstration that we use to find good grasp poses. In contrast with approaches based on grasp points, we do not compute exact finger configurations for the grasp. We do not base our controllers on force-closure arguments (Bicchi 1995) either. It appears that the use of force control is sufficient to grasp most objects as the fingers can naturally adapt to the shape of the object. A model-free grasp planning algorithm based on human demonstrations and using our architecture as been successfully used in Herzog et al. (2012, 2013) to grasp a large number of unknown objects. It shows that our approach is not limited by the use of object models. It has been recently shown in Balasubramanian et al. (2012) that using human demonstrations to plan grasps can lead to more robust grasps than using state of the art grasp planning algorithms based on force closure arguments. It must be noted though that the use of force control and adaptation of finger stiffness might not be sufficient when grasping more complicated objects (e.g. a needle) and in this case the use of force closure arguments might become useful. The idea of using force controllers in the context of grasping is not new (Bicchi et al. 2000) but to the best of our knowledge it has not yet been widely used on experimental platforms.

Motion planning for manipulation: Sampling-based motion planning algorithms have proven extremely successful in addressing manipulation problems (Berenson et al. 2009; Diankov et al. 2008; Rusu et al. 2009). They are very efficient at finding collision-free paths in a high-dimensional state space. While they can quickly find feasible motion plans, they are often lacking in the quality of paths produced. Smoothing and post-processing is typically required before executing these plans on a robot (Hauser and Ng-Thow-Hing 2010), which can significantly add to the required computation time. Trajectory optimization techniques have recently been applied to motion planning for manipulation (Ratliff et al. 2009; Kalakrishnan et al. 2011). These techniques plan collision-free motions by minimizing a cost function over trajectories. Such approaches allow the user to include additional desirable objectives, such as trajectory smoothness, obstacle clearance, torque minimization, and constraint satisfaction. The use of optimization resolves the inherent redundancy in finding collision-free configurations and trajectories, making the robot behave in a more predictable and repeatable fashion.

System integration: The challenges when dealing with computational and physical limitations of fully autonomous systems are often underestimated. Integrating perception, planning, and control into a single platform is a major engineering effort. Impressive integration efforts have been demonstrated by Asfour et al. (2006), Srinivasa et al. (2010) and Chitta et al. (2012). These systems are capable of navigating in kitchen environments and successfully performing tasks such as pick and place as well as opening doors. Similarly, in Beetz et al. (2011), the authors present an integrated system that is capable of making pancakes. Nevertheless, the considered grasping and manipulation tasks are usually carefully selected according to the physical limitations of the robot and often require further modification of the environment. Such shortcuts are usually not allowed in competitive settings. A successful architecture in a recent competition has been presented in Stückler et al. (2013), a competition where each team participated with their own robot. The differences to our competition (besides the considered tasks) were (1) we had no influence on the design of the robot, (2) we were not provided access to the robot before each test, (3) our code was tested independently on a copy of our robot, which required teams to provide turn-key solutions that were robust to slight mechanical differences. Two of our competitors have published their approaches in Hudson et al. (2012) and Bagnell et al. (2012). The authors of Hudson et al. (2012) based their approach on continuous estimation of the entire system state from many different sensor modalities. This enabled the team to accurately track the arm which turned out to be crucial for many of the considered tasks. The authors of Bagnell et al. (2012) present a system focusing on automatic behavior generation that can recover from certain failures. Furthermore, their approach emphasizes on accurate object pose estimation including visual arm tracking. Their approach to trajectory optimization (see Kalakrishnan et al. (2011) for a in-depth comparison) and optimal inverse kinematics (IK) computation is similar to ours. However, in contrast to all mentioned contributions, our approach deals with uncertainty through

compliant and force control rather than relying on precise perception and accurate task execution. We were the only team to estimate the (real) dynamics of the manipulator (see Sect. 3.2.2) enabling us to compute accurate feedforward torque commands using inverse dynamics. This enabled us to significantly lower the PD feedback gains to achieve very compliant behavior while maintaining good tracking performance (see Sect. 3.4).

2.2 The ARM project

The research presented in this paper was developed in the context of the DARPA ARM project. The first phase of the project involved six teams across the United States and ran from August 2010 to November 2011. Each team was provided the same robotic platform (described in Sect. 2.3) and asked to compete in a series of grasping and manipulation challenges. In particular, each team was asked to develop software that enables the robot to autonomously perceive and grasp a total of 12 objects[1], and to perform six manipulation tasks. Each individual grasping and manipulation task was tested five times by placing the objects involved in different (unknown) locations and poses. The primary metric for evaluation was task success followed by speed. Tests were performed on a copy of the robot setup in an independent testing facility. This required each team to create software that is robust to changes in the environment (e.g. lighting conditions) and variations in the robot itself (e.g. sensor offsets). Furthermore, the teams were not provided with remote access to the robot during the final test. Thus, the teams were required to provide end-to-end solutions, both for calibration and task execution. The results presented in this paper were obtained through independent testing, i.e. the teams had no control on the outcome of the tests. We believe that it provides an appreciable degree of objectivity in the results and minimum bias during the experiments.

2.3 The ARM-S robot

The ARM-S robot consists of a Barrett WAM robot, which is a 7-DOF manipulator with a Barrett 6 axis Force/Torque sensor attached at its wrist and a three-finger Barrett Hand (BH8-280), (see Fig. 1). Each finger has one actuated DOF that moves a finger with two articulations. The spread between the left and right finger is actuated and can move these two fingers 180 ° around the palm. These fingers move in a symmetric way. Each finger knuckle contains a strain gage that measures the applied torque. Both the WAM and the hand can be torque controlled. Nonetheless, there is no direct torque sensing on the seven joints of the WAM robot and therefore it is not possible to implement a low-level torque feedback controller. We therefore assume that torque is proportional to the motor current. Another issue is that the WAM has position encoders mounted on the motors but no encoders on the joints. Therefore, due to variable stretch of the cables depending on the robot pose and its previous motions it is not possible to get an accurate position of the joints for fine manipulation. Internal testing using a VICON motion capture system confirmed a difference of up to 4 cm between the real position of the endeffector and the position computed using joint sensing and forward kinematics.

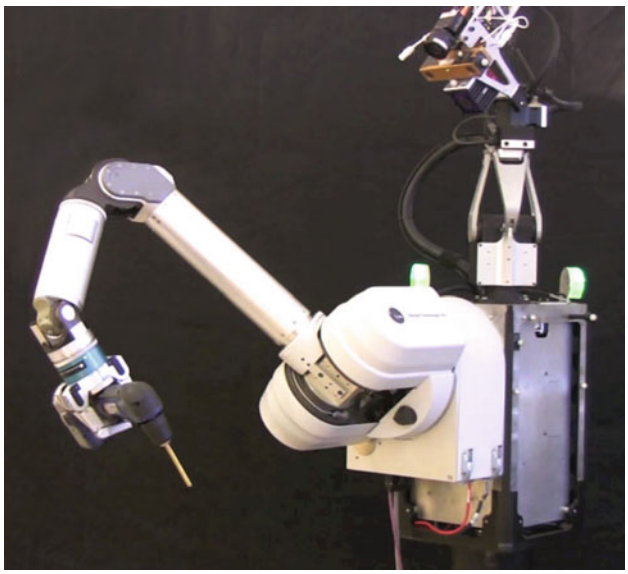


Fig. 1 The ARM-S robot holding a drill. The Barrett WAM robot arm is equipped with a force/torque sensor at the wrist and a three-finger hand. The neck is composed of two pan/tilt units mounted on top of each other. The sensor head is equipped with a high resolution camera, a stereo camera pair, and a 3D TOF camera

The neck is composed of two pan/tilt units from DirectedPerception mounted on top of each other, providing a total of four DOFs. The pan-tilt units are actuated with stepper motors. The visual sensors on the head are an Allied Vision Technologies Prosilica GC2450C for high resolution images, a Point Grey Bumblebee2 for stereo vision and a Mesa Imaging SwissRanger SR4000 for range sensing.

3 Autonomous manipulation system

In this section, we describe the overall architecture of our system, and highlight the different components that are critical to our approach. An overview of our system architecture is depicted in Fig. 2 and described in its caption. Details about each of the components of the system is described in the following subsections.

[1] A detailed 3-D scan of nine of these objects was provided to each team and for the remaining three objects each team was only informed about the class of object.

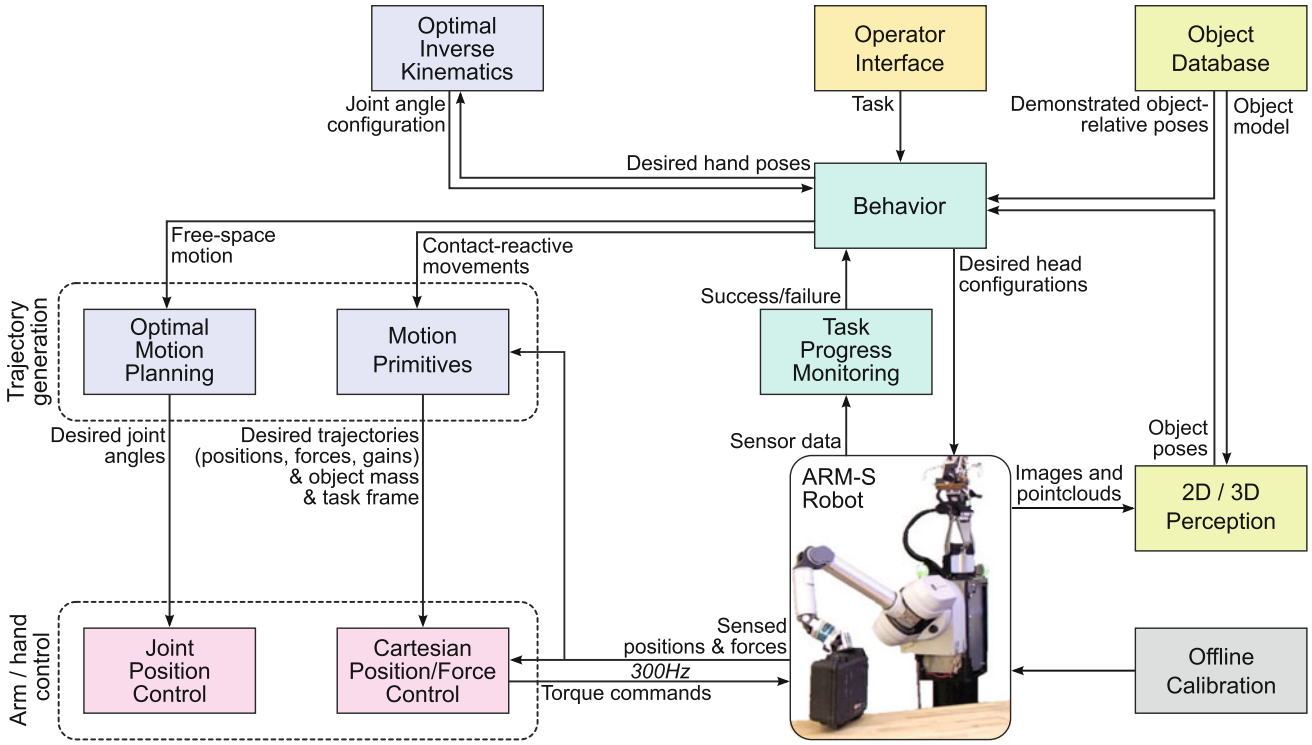


Fig. 2 Overview of the proposed system architecture for grasping and manipulation tasks: The behavior and task progress monitor (*cyan*) are specific to the particular manipulation tasks and have been designed accordingly. The tasks are selected by the human operator (*yellow*), the corresponding behavior then starts and ensures the proper coordination of the other modules. The perception module makes use of prior knowledge about the object shape (*green*) to estimate object poses. The object database (*green*) was trained with a set of corresponding object-relative pre-manipulation poses. Given this set of poses, we compute optimal joint configurations using our optimization based inverse kinematics (*blue*). Desired trajectories are either generated using optimal motion planning or by employing a sequence of contact-reactive motion primitives (*blue*). Finally, the joint space and Cartesian position/force controllers (*pink*) compute and send the torques to the robot that are required to fulfill the task. The offline calibration routines are executed once before running the grasping and manipulation tasks (Color figure online)

3.1 Software architecture

The entire system runs on a single computer with eight cores. It runs the Linux operating system patched with Xenomai (http://www.xenomai.org/) for real-time capabilities. Real-time control is very important when implementing fast control loops that directly access the robot hardware to ensure that sensors and control commands are processed in a timely manner. All the control software runs in real-time, guaranteeing good control performance with the robot. We use our custom made SL control environment (Schaal 2009). The rest of the software is not real-time (calibration, motion planning, vision) and is implemented using ROS (Quigley et al. 2009).

3.2 Calibration

Precise calibration of the sensors and the kinematic and dynamics models the robot is very important to ensure good performance during manipulation tasks. Moreover, due to the testing procedure of the ARM project, we had to provide automatic and robust calibration procedures that could be run by the independent test team. We describe in the following these calibration procedures.

3.2.1 Kinematic calibration

Precise kinematic forward models are important for robots to successfully perform dexterous grasping and manipulation tasks, especially when visual servoing is rendered unfeasible due to occlusions. Unfortunately, there are non-linearities in the kinematic chains of the ARM-S robot (shown in Fig. 2) resulting in significantly different errors for different parts of the state space. The non-linearities in the arm kinematic chain are due to cable stretch and motor-side encoders (see Sect. 2.3) and the head kinematic chain has non-linearities due to a counter balancing spring. A lot of research has been conducted to estimate geometric and non-geometric parameters of kinematic chains to minimize reconstruction errors, see Hollerbach et al. (2008) for an overview. However most approaches either do not consider such non-linearities, or consider chains that are simple enough to carefully engineer a model and estimate their non-geometric parameters. We

propose a data-driven approach that learns (local) task error models that account for such unmodeled non-linearities. We argue that in the context of grasping and manipulation, it is sufficient to achieve high accuracy in the task relevant state space. Therefore, our system is developed to generate subsequent task executions that remain similar to previous ones. That is, we enforce subsequent arm postures and reaching trajectories to be similar to those during training by employing optimization techniques when computing inverse kinematics solutions (see Sect. 3.5.1) and movement plans (see Sect. 3.5.2). It is very important to note, that this is not a limitation of the proposed approach, since the task error model considers the state space that is relevant for the task. We do not see any advantage of trying to achieve an accurate forward model for the entire state space.

The objective of calibrating the head kinematic chain, i.e. the two pan/tilt units, first is to enable our system to accurately merge 3-D point clouds of perceived objects from different view angles (see Sect. 3.3). The objective of calibrating the arm kinematic chains is to get an accurate object pose estimate in the hand frame necessary for dexterous manipulation. For both these calibration tasks we employ the same two staged approach. First, we optimize for the best transformation that minimizes the systematic pose error between the forward model and the visually perceived 6-D marker poses. Second, we learn a local non-linear correction term $\Delta \boldsymbol{x}$ that absorbs the remaining task error as a function of the current joint angle configuration $\boldsymbol{\theta}$.

Accounting for non-linearities in the head kinematic chain The training phase involves the robot moving its pan/tilt units into 60 commonly used configurations while recording perceived 6-D marker poses of the fiducials arranged in the calibration target which is lying flat on the table (see Fig. 3). Then we optimize for the 6-DOF transform between the top of the head's pan-tilt mechanism and the stereo camera that minimizes reconstruction error. Finally, we compute the remaining error and learn six Gaussian Process (GP) Regression models to predict the six correction terms, i.e. translation in x, y, z and corresponding rotations $\phi_{\text{roll}}, \phi_{\text{pitch}}, \phi_{\text{yaw}}$. Thus, the GP model provides us with a joint configuration dependent 6-DOF pose correction which we apply to the previously estimated fixed pose.

Accounting for non-linearities in the arm kinematic chain To determine the relevant task space when calibrating for non-linearities in the arm kinematic chain we process all previously stored pre-grasp and pre-manipulation poses and compute 150 IK solutions, i.e. joint configurations, for the most prominent ones. The training phase again consists of having the robot reach and retreat to each of these arm postures while recording the visually perceived 6-D marker poses of the fiducials on the cube like calibration target mounted at the endeffector of the WAM robot arm (see Fig. 3). To account for systematic error in the mounting of the Barrett arm with respect to the base frame, we again optimized a full 6-D transform to minimize reconstruction error. The remaining 6-D pose error is computed using this fixed offset and again approximated as a function of the arm joint configuration using six separate GP models.

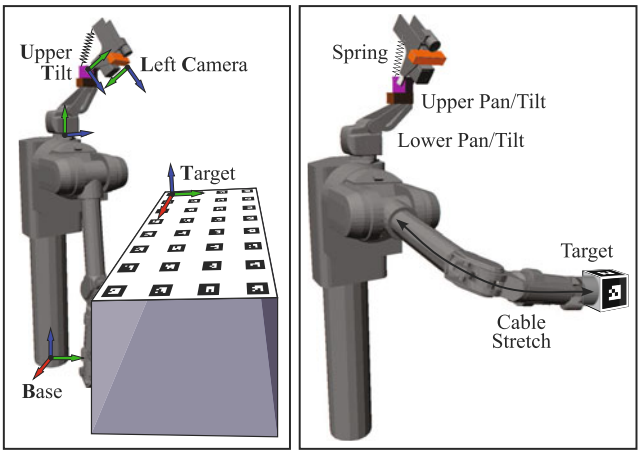


Fig. 3 Sketch of the calibration procedures of the kinematic chains. The non-linearities in the forward model of the head kinematic chain is introduced by the spring (*left*). The non-linearities in the forward model of the arm is introduced due to cable stretch and motor-side encoders (*right*)

A video showing the kinematic calibration procedure can be seen at http://youtu.be/QKKViNGhWWQ. Please refer to Pastor et al. (2013) for a more detailed description of the method as well as experimental evaluations.

3.2.2 Dynamic parameter estimation

In order to use a model based torque control approach, we need an accurate model of the dynamics for the arm. Given the little inertia of the fingers we do not model their dynamics. We do not model the dynamics of the head either since it cannot be torque controlled.

The model of the dynamics given by the CAD model of the WAM was not sufficient for good tracking performance while keeping low PD gains. Therefore we estimated the dynamic parameters of the robot using standard inertial parameter estimation algorithms (An et al. 1985). The mapping between joint torques $\boldsymbol{\tau}$ and the inertial parameters $\mathbf{m}$ of a manipulator is a linear function that depends nonlinearly on the position, velocity and acceleration of its joints

$$\mathbf{H}(\mathbf{q}, \dot{\mathbf{q}}, \ddot{\mathbf{q}})\mathbf{m} = \boldsymbol{\tau} \tag{1}$$

We use a linear regression algorithm to estimate the inertial parameters that best match the torques and observed joints motion. In order to properly observe the inertial parameters we designed trajectories composed of superimposed sines with incommensurate frequencies in order to excite most

modes of the dynamics. Our model estimates the mass, the position of the center of mass multiplied by the mass, the inertial tensor of each link as well as viscous friction (linear in joint velocity) and a Coulomb friction term (corresponding to a constant term opposite to joint velocity when the velocity is not zero).

3.2.3 Force sensor calibration

When the robot is powered on, it is necessary to compute the offsets of the six axis force sensor at the wrist. The offsets are computed knowing the initial pose of the robot and the mass of the hand (i.e. we compute the theoretical force applied by the hand on the endeffector).

During our experiments, it turned out that the force sensor had a significant level of hysteresis and the sensor values would drift significantly when the robot was moving. This effect was systematically observed on the several Barrett force sensors we used. Therefore, the procedure of zeroing the force sensor had to be repeated each time before a force controlled sequence (i.e. before interacting with objects). The hysteresis of the force sensor was a major problem, since only short force control sequences could be done before zeroing the sensor making our approach slower. Moreover, it did not allow us to perform online estimation of the inertial properties of objects grasped by the robot with enough accuracy (Atkeson et al. 1985).

3.2.4 Strain gage calibration

We calibrate the strain gages by fitting a linear model that maps sensor readings into physically consistent values (i.e. into Nm). In order to automatically do that, we use the force information at the wrist to infer the forces in each finger. At the beginning of the calibration process the hand of the robot goes above the table and the fingers are moved into an equilateral triangle configuration (for equal force distribution). Papers with low friction are placed under each finger. The robot then presses against the table with the fingers with several different vertical forces. Theoretical forces in the knuckle are computed from the vertical forces (which is direct because the fingers are in a equilateral triangle configuration). A linear regression between strain gage sensor readings and theoretical forces is used to get the sensor model. This calibration procedure was very useful to get consistent force readings across fingers and hands. The automatic procedure could be reliably run by the independent testing team.

3.3 Perception

For perception, we primarily use the Bumblebee2 stereo camera on the head of the robot. Because we know which objects will be present for each task, we develop unique strategies for each object. The basic strategies which we find most useful are:

- Iterative closest point (ICP) based template matching of templates to rigid object models.
- Template matching of 2D depth templates against the depth image
- Template matching of visual templates against color image data

For objects with enough texture to provide good depth data, we find simple ICP to work surprisingly well. To make the ICP step more robust, for each object we find the (relatively small) set of poses in which the object can rest on the table. This allows us to restrict the problem to solving for the two-dimensional translation and rotation which aligns the object model in a stable pose to the observed point cloud.

For this approach, we first cluster 3D points which are close together. We then match each 3D object template against each cluster of points using ICP, and use a combinations of number and percentage of inliers to choose the best match. When recognizing a scene with M models of objects expected to be in the scene and N observed clusters, we first perform an ICP registration of each model to each cluster. This requires running ICP $M \times N$ times. We then use brute force search to choose the correspondence between models and clusters. Although this second step requires trying a combinatorial number of possibilities, we do not need to re-execute ICP for each one, and so the running time is relatively low.

For objects with enough texture to provide depth data from the stereo cameras but with shapes that did not work well with ICP (because of local minima), we search for a 2D depth template. For this approach, a multi-level search is executed over a grid of possible x, y, and angle possibilities. First the best match is found using a course grid of possible poses; then the poses around that match are searched at a higher grid resolution to improve the pose estimate.

For some objects (such as a plain white telephone shown in Fig. 8f), there is not enough texture to give good stereo depth data. For such objects, we match visual templates to find the object. Depending on the object, we use either an RGB template, an HSV template, or a simple binary template to match the shape of the object to pixels in a background subtracted image.

For all of the template matching methods, we found it convenient to first construct a stitched together orthographic image of the table surface. Because the stereo cameras provide 3D points on the surface of the table, it is relatively easy to fit a plane to these points and project multiple camera images onto the table surface. By scaling this image such that each pixel represents a known sized patch on the surface of

the table, we avoid needing to search over all possible scales of the object in the image.

One challenge we faced when trying to find objects with a known appearance in color images is that the perceived colors vary quite a bit depending on the current lighting conditions. We mitigate this using two different approaches. First, we calibrate the white balance of the stereo cameras using the color of a known object in the scene—in this case the table. Second, we train a support vector machine (SVM) classifier for colors which we need to detect often. One example of this is finding a piece of paper. We collected examples of the perceived color of the paper under different lighting conditions, and then trained a kernel SVM.

3.4 Control system

One key element in our manipulation architecture is the use of force and torque controllers wherever possible. Our requirements for the control architecture are: high tracking performance in joint space and in Cartesian space (i.e. end-effector), variable stiffness and end-effector force control. The control is split into three independent parts: the head, the arm and the hand.

3.4.1 Head control

The head is simply controlled using the position control interface for the stepper motors.

3.4.2 Hand control

The fingers of the Barrett Hand are naturally very stiff due to the worm gear. We implemented active force controllers using the strain gages in the knuckle of the fingers in order to have more compliant fingers (e.g. when controlling zero force) and to be able to actively regulate the forces at the finger tips. We send torque commands to the fingers; the command is composed of a PD term for position and a P term for force control. The control law is

$$\tau_{finger} = K_p \left(q_{des} - q_{finger}\right) + K_d \left(\dot{q}_{des} - \dot{q}_{finger}\right) + L_p \left(F_{des} - F_{finger}\right) \quad (2)$$

where q_{finger} and q_{des} are the current and desired position of a finger, F_{finger} and F_{des} are the current and desired torque at the knuckle of the finger, K_p, K_d and L_p are positive gains. With such a setup it is possible to change the stiffness of the fingers by changing the gains of the force control (when commanding zero force). For example canceling these gains will lead to normal position control with very high stiffness, while having these gains set to their maximum will lead to an extremely compliant behavior where the position control will just act as spring and damper term. Moreover it is possible to precisely regulate the forces at the finger tips during grasping. The control architecture is summarized in Fig. 4.

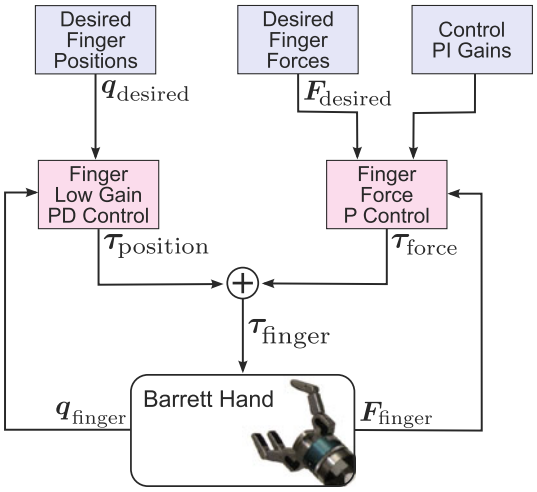


Fig. 4 Control architecture for the hand

3.4.3 Arm control

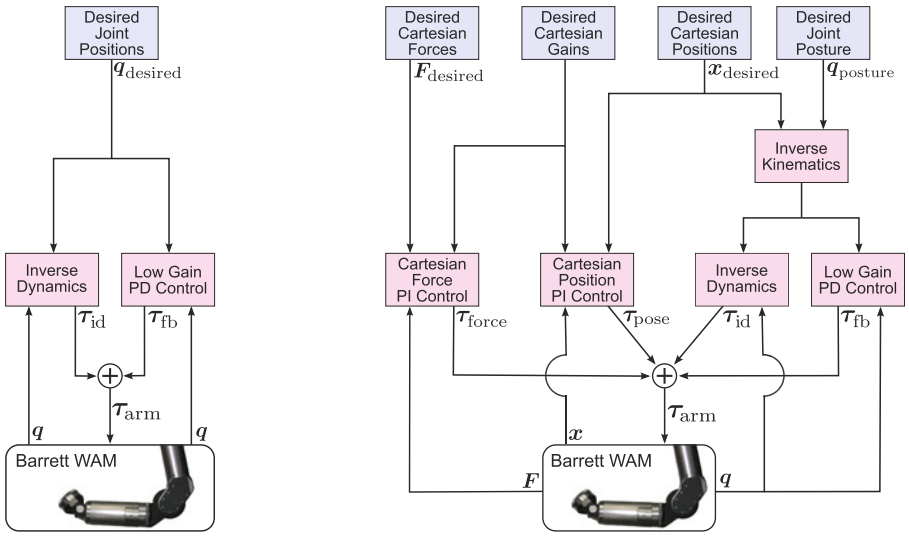
There are two modes of control for the arm, which share the same core components: joint space control and Cartesian space control (e.g. the control of the position and orientation of the hand or an object in hand like the drill bit during drilling). Both control architectures are shown in Fig. 5.

Inverse dynamics The two modes of control share an inverse dynamics controller with low gain PD control for each joint. The inverse dynamics controller is used to cancel the dynamics of the robot (i.e. feedback linearize the system) and usually accounts for more than 90 % of the torques. PD control at the joint level ensures appropriate disturbance rejection and correction of small modeling errors. The PD controllers have low gains to ensure a certain level of compliance. They are tuned to have approximately 200–300 ms rise time with a bit more than critical damping. The inverse dynamics and joint feedback torques are evaluated as follows:

$$\boldsymbol{\tau}_{id} = \mathbf{M}(\mathbf{q})\ddot{\mathbf{q}}_{des} + \mathbf{h}(\mathbf{q}, \dot{\mathbf{q}}) \quad (3)$$

$$\boldsymbol{\tau}_{fb} = \mathbf{K}_P \left(\mathbf{q}_{des} - \mathbf{q}\right) + \mathbf{K}_D \left(\dot{\mathbf{q}}_{des} - \dot{\mathbf{q}}\right) \quad (4)$$

where $\mathbf{q}$ is the vector of joint positions, $\mathbf{M}$ is the inertia matrix and $\mathbf{h}$ are the modeled forces including gravitational, Coriolis, centrifugal forces, viscous and Coulomb friction as modeled in Sect. 3.2.2. $\mathbf{K}_P$ and $\mathbf{K}_D$ are diagonal matrices that define the PD gains. We use the Featherstone (1987) formulation of the Newton–Euler algorithm to compute the inverse dynamics. We note that we never need to explicitly compute the inertia matrix or the force vector $\mathbf{h}$.



(a) Joint control diagram **(b)** Cartesian control diagram

Fig. 5 Control architectures for the arm. *On the left*, the architecture for joint space control is shown. *On the right*, the Cartesian space controller is shown. A detailed explanation of the components of the architecture is given in the text

Joint space control The joint space controller only uses the inverse dynamics block, where desired position and velocities are sent to the PD controller and the desired acceleration is used to compute the inverse dynamics torques. Very high performance tracking is realized even when moving at high speeds while the robot remains compliant.

Cartesian control Cartesian control is used when the robot is manipulating an object, therefore it is desirable to precisely control the position in certain task directions, while also regulating force interactions at contact. A simple example of such a setup arises when the robot is drilling. The drill bit needs to be placed precisely above the drilling target which requires very stiff control of the position in the horizontal plane while it is necessary to apply a large force in the vertical direction to drill.

We decided to use a mixture of joint space control (for rough positioning and cancellation of the robot dynamics) and hybrid position and force control at the end-effector (for precise control of the dynamics at the end-effector).

The desired Cartesian positions and velocities are converted into desired joint velocities using a differential IK algorithm. We compute the IK using the pseudo-inverse of the end-effector Jacobian computed with a singular value decomposition (SVD). Singularities are dealt with by damping directly the singular values that become too close to zero (Chiaverini 1997). Desired positions are computed by integrating the desired velocities, and the desired accelerations are computed through differentiation.

In order to control both stiff positions and forces in different directions, we use a hybrid force/position control approach Chiaverini and Sciavicco (1993) and Yoshikawa (2000) where two PI controllers for both position and force control are mapped into joint torques

$$\boldsymbol{\tau}_{force} = \mathbf{J}^T \left(\mathbf{K}_{FP}(\mathbf{F}_{des} - \mathbf{F}) + \mathbf{K}_{FI} \int_{\Delta T} (\mathbf{F}_{des} - \mathbf{F}) dt \right)$$

$$\boldsymbol{\tau}_{pose} = \mathbf{J}^T \left(\mathbf{K}_{PP}(\mathbf{x}_{des} - \mathbf{x}) + \mathbf{K}_{PI} \int_{\Delta T} (\mathbf{x}_{des} - \mathbf{x}) dt \right)$$

where F is the force at the endeffector, $\mathbf{x}$ is the Cartesian position and $\mathbf{K}_i$ are gain matrices. ΔT is the time window during which errors are integrated. We schedule the gains, such that directions that are controlled in position or force are varied according to the task. In practice the scheduling is often limited to turning on or off the control directions. Although not perfect from the theoretical point of view (Yoshikawa 2000) this hybrid feedback control approach mixing positions and forces worked very well in practice. Several potential problems due to the manipulator dynamics are avoided due to the cancellation of this dynamics with the inverse dynamics.

Task frame For different tasks it is desirable to perform Cartesian control in different frames. For example, when drilling it is useful to directly set the frame of control at the tip of the drill bit so we can regulate the force parallel to the drill bit while having good control of the positions orthogonal to it for drilling at the right position. Therefore, Cartesian control is implemented in a configurable task-frame that allows to change the coordinate system in which the control is to be done relative to the endeffector. It is then possible to align the directions of control with directions relevant for the task and set control gains accordingly. During drilling, for example, this frame is set automatically after perceiving the position of the drill bit relative to the hand.

Object mass compensation Grasped objects can significantly influence the dynamics of the robot and the control performance. We therefore add to the overall joint torques a feedforward object mass compensation command to cancel the forces and moments exerted at the endeffector by the grasped object

$$\boldsymbol{\tau}_{obj} = \mathbf{J}^T \begin{pmatrix} m_{obj}\mathbf{g} \\ \mathbf{x}_{obj} \times m_{obj}\mathbf{g} \end{pmatrix} \tag{5}$$

where m_{obj} is the mass of the object, $\mathbf{x}_{obj}$ the position of its center of mass relative to the endeffector and $\mathbf{g}$ is the gravity vector. Note that we treat the object as a point mass and not as a rigid body (i.e. we do not consider its moment of inertia). We use the mass of the object given by the object model and do not estimate the object mass online. Such estimation would be straightforward (Atkeson et al. 1985) if the force sensor did not have strong hysteresis problems as explained in Sect. 3.2.

Design choices due to robot limitations The control system runs at 300 Hz despite the fact that the WAM can theoretically be controlled at 500 Hz. The communication to the arm and hand is done on a single CAN bus. The bandwidth of the CAN bus proved to be the limiting factor when trying to access all the sensors and control the joints of the WAM and the hand at a high rate (i.e. in our case the full CAN capacity is used). All sensors are read at 300 Hz except for the touch sensors which are updated and thus read only at 25 Hz.

This control bandwidth limitation was problematic when trying to directly implement resolved acceleration controllers (Nakanishi et al. 2008), where feedback control is directly done in Cartesian space using Jacobian pseudo-inverses. Due to sensor noise and limited bandwidth it was not possible to sufficiently increase the control gains to obtain satisfactory performance without going unstable (the system could not be damped enough). This is the reason we implemented IK with joint space control. Feedback control of the end-effector is then done using a hybrid force/position controller using the Jacobian transpose (Yoshikawa 2000). An advantage of this controller is that it seems to have a more "natural" or human-friendly response to disturbances and was extremely stable to external perturbations. The resulting controller led to satisfactory performance for Cartesian position and force tracking.

3.5 Planning and motion primitives

In the following we detail the trajectory generation level. We used two approaches to move the robot: planned joint trajectories to move across the workspace (e.g. to move the arm close to an object or to transport an object to a target position) and sequences of motion primitives in task space to grasp and manipulate objects.

We chose to use joint space motions to move across the workspace because there exist several kinematic singularities across the workspace and it is therefore not possible to have local Cartesian motion using differential kinematics that can move in straight lines between any two arm configurations. As opposed to simply planning collision-free configurations or motion plans using randomized planners, we use optimization methods to generate smooth and predictable trajectories. Such methods allow us to define a set of desirable characteristics for configurations or trajectories, and optimize over them. The redundancy in planning collision-free motions is resolved by optimizing an objective function, which results in repeatable and predictable trajectories. This is especially useful in the case of the cable-driven Barrett WAM arm, because the kinematic calibration errors are not only configuration-dependent, but also path-dependent (see Sect. 3.2.1).

The choice of Cartesian motion in the region of manipulation was because it provides a simple and natural way to create motions of the hand relative to an object that includes force control in the frame of the task of interest. By optimizing an objective function that includes the manipulability measure and joint limits when finding the IK solution, we ensure that the robot has an optimal range of motion in Cartesian space during the manipulation task.

3.5.1 Optimal inverse kinematics

In order to move the arm in joint space towards an object or a specific target, we need to find the joint configuration of the end point. We solve this problem using differential IK similar to the one used at the control level. We generate 100 random joint configurations, and then iterate the following update on each of them until convergence:

$$\mathbf{q} \leftarrow \mathbf{q} + \mathbf{J}^{\dagger}\left(\mathbf{x}_{des} - f(\mathbf{q})\right) - \left(\mathbf{I} - \mathbf{J}^{\dagger}\mathbf{J}\right)\frac{\partial C(\mathbf{q})}{\partial \mathbf{q}} \tag{6}$$

where $\mathbf{q}$ is the joint configuration, $\mathbf{x}_{des}$ represents the desired Cartesian pose, $f(\mathbf{q})$ is the non-linear forward kinematics function of the tool control point (TCP), $\mathbf{J}(\mathbf{q})$ is the geometric Jacobian matrix containing partial derivatives of the TCP w.r.t the joints, and $\mathbf{J}^{\dagger}$ refers to the Moore–Penrose pseudoinverse of $\mathbf{J}$. Note that the forward kinematics function $f(\mathbf{q})$ also includes the non-parametric hand-eye calibration model (as described in Sect. 3.2.1). In some cases, the desired pose $\mathbf{x}_{des}$ is not fully specified in order to allow for more flexibility in the solutions. For example, when grasping a cylindrical object, due to the symmetry along the long axis, the orientation of the hand can be rotated around this axis and thus we do not specify it in the pose $\mathbf{x}_{des}$. By dropping the relevant row from the term $(\mathbf{x}_{des} - f(\mathbf{q}))$ and the Jacobian $\mathbf{J}$, we get one more degree of freedom in the null space of the task. These redundancies in the joint angles are resolved by minimizing the cost function $C(\mathbf{q})$ in the null space of the

task constraints. The cost function is a weighted linear combination of the following features:

- Obstacle clearance: the arm must be kept a certain minimum distance away from the table and other obstacles.
- Joint limit avoidance: the joints must be kept away from their limits in order to increase manipulability in the workspace.
- Manipulability measure: for efficient Cartesian motion, the arm must be kept away from singular configurations, which can be evaluated using the manipulability measure (Sciavicco and Siciliano 2000): $C_{mm}(\mathbf{q}) = \sqrt{\det(\mathbf{J}\mathbf{J}^{\mathrm{T}})}$
- Occlusions: for tasks that include visual feedback, (e.g. drilling, Sect. 4.3.2), configurations that provide a direct line of sight between the cameras and the tool being manipulated are preferred.
- Similarity to demonstrations: in order to improve the repeatability of IK solutions, we minimize the configuration-space distance to a preferred kinematic configuration.

The weights on each of these cost terms was tuned by hand in this work, although we have since shown that these weights can be learnt from demonstrations (Kalakrishnan et al. 2013). Since Eq. 6 converges to a local minimum for each of the 100 random joint configurations, we pick the one that satisfies all task constraints and achieves the lowest cost.

Some tasks involve moving between two distinct parts of the workspace, e.g. grasping and transporting an object to a different location, or picking up a drill and drilling a hole on a red dot. In such cases, we compute both IK solutions jointly, with an additional cost on the configuration space distance between the two solutions. This ensures that both configurations are similar to each other, simplifies motion planning between the two states, and minimizes extraneous motion of the arm while the object is in the hand.

3.5.2 Stochastic trajectory optimizer

In order to plan joint-space trajectories across the workspace, we use the STOMP algorithm (Kalakrishnan et al. 2011). We initialize STOMP with a straight-line trajectory in configuration space. The algorithm proceeds by iteratively sampling trajectories around the current optimum, and combining them to produce a trajectory of lower cost. This procedure does not require access to gradients of the cost function, which may not exist or may be expensive to compute. The exploration and update rules ensure that the trajectory is smooth after every iteration. No smoothing or post-processing is required before executing the trajectory on the robot. The cost function is designed as a weighted linear combination of the following features:

- Obstacle clearance: keeps the arm away from the table and obstacles. This can be computed efficiently by using the signed Euclidean Distance Transform (Ye 1988) of the 3-D obstacle voxel grid.
- Joint accelerations, jerk: ensures that trajectories are smooth in joint-space.
- End-effector velocities: ensures that the distance traveled by the endeffector is minimized. Minimizing the angular velocity component prevents unnecessary rotation or tilting of objects in the hand.

Similar to the IK objective function, the weights for the above cost terms may be learnt from demonstrations (Kalakrishnan et al. 2013), although they were tuned by hand for the purpose of this work.

3.5.3 Motion primitives

In addition to joint space motion planning, we used a library of motion primitives. These primitives consist of simple parametrizable motions that are sequenced to create more complex manipulation skills. The motion primitives that we used are:

- Cartesian motion. Generates a motion of the endeffector in Cartesian space. It is used to move the hand, or an object in hand in a specific direction or orientation.
- Cartesian force. Generates a desired force trajectory. It is used to apply a desired force in a certain direction.
- Poke. A motion that moves the hand in a specific direction until it touches an obstacle. It is usually used before a Cartesian force primitive.
- Finger motion and force. Generates a desired finger motion with a desired force trajectory. This primitive is used to grasp objects.
- Gain scheduling. Creates both position and force gain profiles in different directions of control. It is used in conjunction with the motion and force primitives.
- Change task frame. Changes the orientation of the end-effector frame, in order to align the directions of control with directions relevant for the task (e.g. the direction corresponding to an object in hand).
- Force grasp. Generates a stereotypical grasping motion with desired contact forces.

3.6 Grasp planning and execution

Our approach to grasping objects in general greatly differs from usual approaches, in the sense that we do not compute grasp points or precise finger locations on the object but we rely on force control to achieve a desired grasp. For the objects with known models, we first demonstrate a set of pre-grasp poses by kinesthetically moving the hand to

the desired locations relative to the object. The pre-grasp pose is stored as the 6-DOF pose of the hand relative to the object pose. During execution, we check the feasibility of each object-relative pre-grasp pose given the detected object location, and select the first one. Each of these pre-grasp poses is associated with a set of motion primitives that when executed achieve the desired grasp. In most cases, the primitives involve moving the hand forward until it touches the object (or table), followed by closing the fingers by controlling the desired force at each finger. Such an approach allows to create compliant grasps but also to adapt the finger stiffness when required (e.g. when grasping heavy objects). This simple approach to grasping turned out to be very efficient and robust. The exact finger positions do not need to be specified and therefore they can adapt to arbitrary shapes. Also, if the object moves in the hand, the fingers automatically adapt their position to maintain contact with it. We note that we do not use any force closure arguments in our experiments as the complexity of such algorithms is usually very high. When grasping objects such as a screwdriver or a hammer lying flat on the table, we first touch the table from above, and then use force control in the vertical direction to servo a downward force. This ensures that the fingers maintain contact with the table while closing around the object.

When dealing with objects in a known class but without a model, we simply used the model we had of the object in that class, both for perception and for grasp planning. The strategy of controlling forces in the fingers and the wrist provided sufficient robustness to the presented variations in size and shape of the object. Although not applied to the experiments presented in this paper, we have since developed a model-free grasp planning approach which does not require models, and improves its own performance through trial and error (Herzog et al. 2012, 2013). The experiments in the cited paper were conducted using an identical control architecture, with just the grasp planning module substituted with the model-free version.

3.7 Failure and success detection

In order to increase the robustness of the tasks, it is necessary to constantly assess the success or failure of the executed subtask, in order to decide if the action needs to be re-executed or if it is possible to execute the next action. For example, during stapling, the robot ensures that the paper was correctly stapled by monitoring its force sensor. A force peak that was characteristic of successful stapling could then be detected. In case stapling was not detected, the robot would re-execute the stapling task. Another example, that was generic for all grasping tasks was to detect that the object was in the hand of the robot, by detecting the mass of the object if it was heavy enough to be detected, and by detecting the position of the fingers. Since the fingers were force controlled, completely closed fingers would mean that the object was not in hand (when the object was large enough). Detection that the drill was turned on was done by looking at the frequency spectrum of the force sensor at the wrist, where a characteristic frequency would appear when the drill was on. Such failure detection and appropriate recovery behavior helped to significantly increase the task performance. However, most detection routines were engineered for each task and not automatically deduced according to the task. Further research on a more automatic approach to failure detection has been pursued in Pastor et al. (2011).

4 Experimental results

In the following, we present the grasping and manipulation tasks that were achieved using the proposed architecture as well as the experimental setup and the results in the context of the ARM project.

4.1 Independent testing

The proposed system was tested by an independent external team in the context of the ARM project. It means that all the grasping and manipulation tasks we describe in the following were tested by the official test team of the project. We uploaded our software and after running the calibration procedure and basic sanity checks to make sure the system was behaving correctly, the test team ran our software on each grasping and manipulation tasks. We had no control on the way the tests were conducted (i.e. objects are placed in new arbitrary poses and locations for each trial and we could not optimize anything based on experimental bias). We did not attend the official tests. All tests were run five times and the best four runs were taken into account.

4.2 Experimental setup

All tasks were executed with the same setup. Objects of interest would lie on the table and a 3-D model of the object would be available (expect for some grasping tasks). The objects could be placed in arbitrary positions on the table and for some grasping tasks they were also held balanced with an invisible wire, such that their pose could be arbitrary. The robot then had 5 min to execute the task and stop when it was done. No human intervention was possible during the execution of the task.

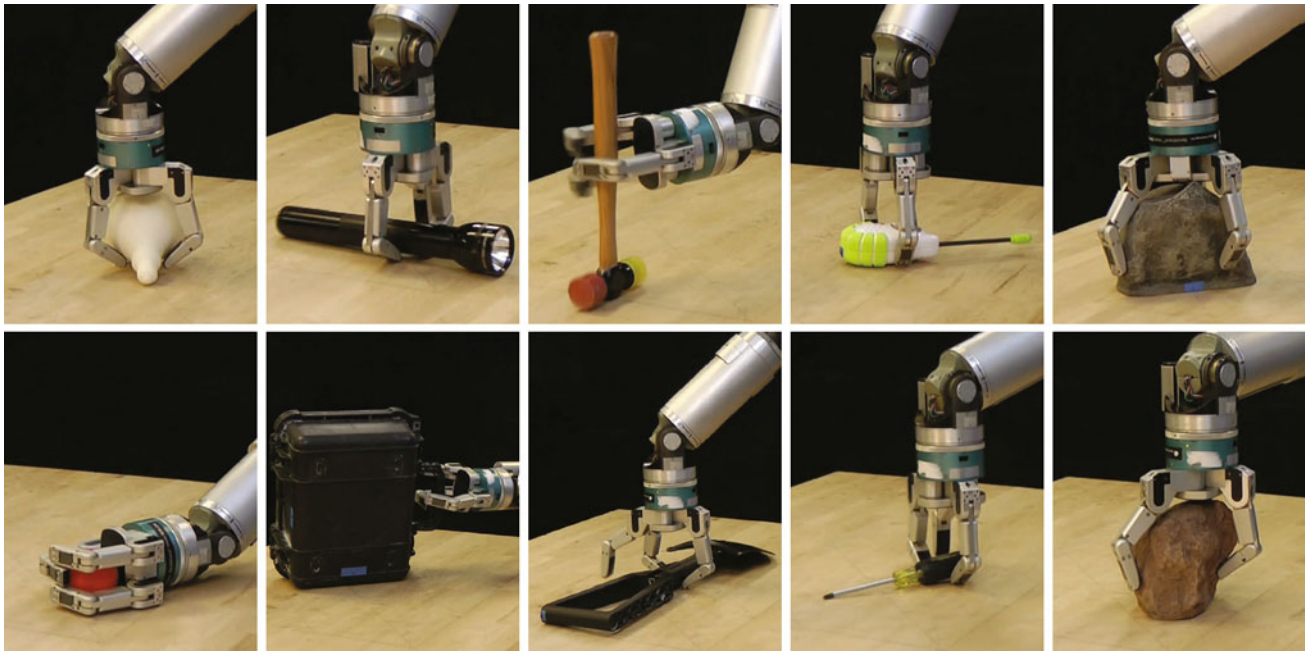


Fig. 6 The figure illustrates the objects grasped during the evaluation together with the types of grasps executed. Note that the lower pictures with the screwdriver and the rock were unknown objects (different from the object models). We notice in these pictures that the robot often touches the table, which is possible thanks to force control. It allows the robot to naturally align its hand with the table to ensure that all fingers touch the table (as in the radio picture), and to execute grasps that collision-free motion planners could not plan (such as the grasp of the ball where the wrist is lying on the table). The same grasp strategy is used for both rocks: controlling the force at each finger results in natural adaptation to their very different shapes

4.3 Grasping and manipulation challenges

The system was tested on 12 different grasping and six manipulation tasks. We describe these tasks below.

4.3.1 Grasping tasks

The objects to be grasped were: a screwdriver, a radio, a rock, a ball, a flashlight, a hammer, a case, a floodlight and a shovel (see Fig. 6). For all these objects we had a 3-D model of the objects that were used with our perception module. In addition, to demonstrate the generalization capabilities of our system, three unknown objects of known type were to be grasped: a screwdriver, a hammer and a rock. We did not have models for these objects and therefore, for the perception and grasp selection part, we used the models of the already known object of the same class. It thus demonstrates that our algorithms do not rely too much on precise object models.

4.3.2 Manipulation tasks

Figure 8 shows an example of each of the manipulation tasks. For each task, we created a high level state machine using a sequence of motion primitives, failure detectors and recovery primitives in order to achieve the task with a high probability of success. An example of the control flow for one of the tasks is shown in Fig. 7. The tasks and our approaches to solving them are briefly described below:

- Stapling: the robot had to staple a stack of paper. The papers were already placed in the stapler. After moving the hand above the stapler, we touch the sides of the stapler with the fingers to realign the hand with the stapler. We then apply a desired force to push the stapler down until success, which is detected by looking for a spike in the force sensor.
- Turn on flashlight: the robot had to turn on a flashlight by triggering its button. A sequence of motion primitives was used to close two fingers around the flashlight and push the button with the third.
- Open door: the robot had to open a door using the door handle. A few "poke" motion primitives were used to touch the door and the top and inside of the door handle, ensuring a proper grasp of the handle. The motion to turn the handle and open the door was previously recorded from a kinesthetic demonstration and replayed in Cartesian space relative to the pose of the hand after grasping.
- Unlocking a door: A key was initially placed in the hand of the robot and it had to unlock the door lock. Similar to the previous task, a set of "poke" motion primitives were used to touch the front, top and side of the handle with the key, in order to accurately localize the hand.

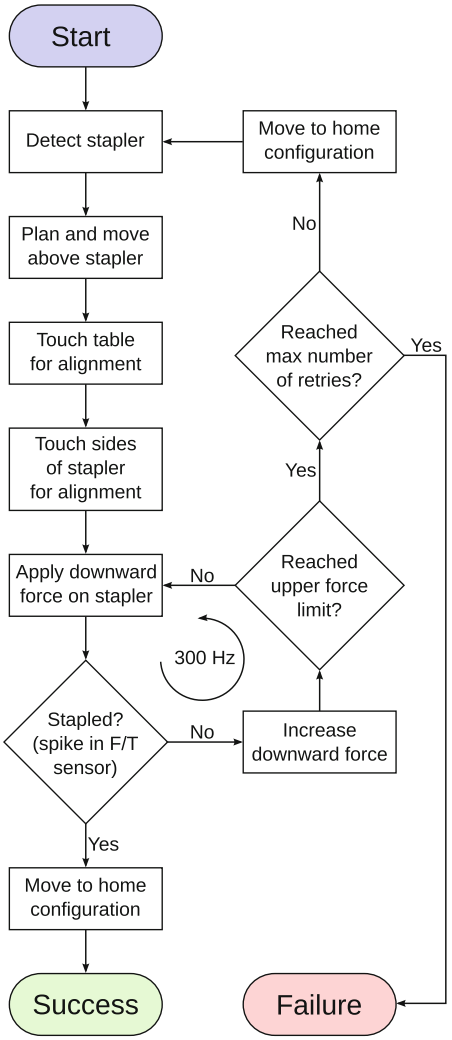


Fig. 7 A simplified control flow diagram for the stapling task

The actual key insertion was achieved by maintaining a forward force with the key on the handle and sliding it down until it slipped into the hole.

– Drilling: The robot had to pick up a drill and drill a hole in a designated red spot on a piece of wood. The piece of wood could be lying on the table or be vertical. After grasping the drill and turning it on, we detect whether it was on by analyzing the frequency spectrum of the force sensor. We estimate the pose of the drill tip by fitting a line to the 3-D points in the appropriate region of the stereo camera view. This allowed use to re-estimate the position of the drill in the hand, and provided the required accuracy for moving the drill tip to the red dot. Drilling was achieved by first touching the red dot with a "poke" primitive, followed by applying increasing downward forces until the desired drill depth was achieved.
– Hangup phone: The robot had to pick up a phone and place it on its cradle to hang it up. The cradle could be placed anywhere on the table or on the side of the door. We first touch the sides of the phone and the cradle to reorient and accurately localize the phone in the frame of the hand. After picking up the phone, we use 2-D template-matching on the dial pad to determine the orientation of the phone in the hand, and then move it in front of the cradle with the right orientation. Finally, a set of force-controlled motion primitives were executed that touch the cradle with the phone and allow it to slip into place.

4.4 Results

Results of the experiments done at the test facility are shown in Table 1. A compilation of the considered grasping and manipulation tasks can be seen at http://youtu.be/VgKoX3RuvB0 and in the video supplement.

For comparison purposes we also show the results of the five other teams against which we were competing. We can see in the table that our approach led to the best success rate (ex-aequo with Team A), with a comparable average time performance.

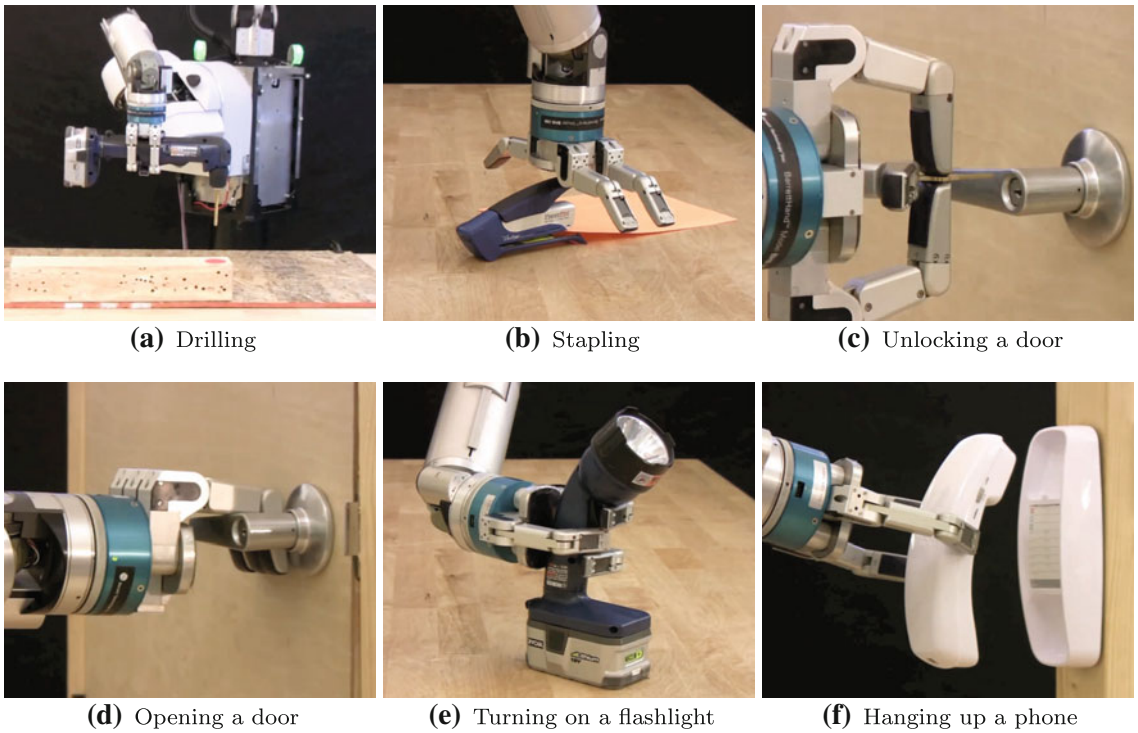
In the grasping tasks, our approach failed only once out of 48 trials which demonstrate the robustness of the grasping approach. The manipulation results have slightly lower performance, as there were four failures. Two failures occurred during drilling (when the robot was not able to hit the red spot), and the other two occurred while hanging up the phone on the vertical cradle (when the phone would slip out of the cradle because it was misplaced). All the other manipulation tasks were successful for all trials. It is worth mentioning that the manipulation tasks were far from trivial as several competing teams had low success rates on them.

It is important to take these results carefully because several uncontrollable reasons can have an impact on them, for example unexpected behaviors of the system during the competition or unseen bugs. Therefore, it is not really possible to conclude that the approaches of the other teams were in theory less good. However, we believe that these results offer a strong argument in favor of our system architecture since it was competitive with other state of the art approaches.

We only presented here the official results from the ARM tests, but it is worth noting that we had very similar performance when doing tests on our own platform. This demonstrates the robustness of our approach as well as the reproducibility of the results.

5 Discussion

In the following we discuss the advantages and drawbacks of our approach. Then we discuss important research directions to significantly improve the autonomy of our manipulation architecture.



(a) Drilling **(b)** Stapling **(c)** Unlocking a door

(d) Opening a door **(e)** Turning on a flashlight **(f)** Hanging up a phone

Fig. 8 The figure depicts the six manipulation tasks executed during the evaluations. Videos of these tasks may be found in the attached multimedia extension

Table 1 Results from testing conducted by an independent test team

Team	Successes (out of 72)	Grasping (out of 48)	Manipulation (out of 24)	Average time (s)
Team A	**67**	**47**	**20**	**75.4**
USC	**67**	**47**	**20**	80.6
Team B	64	46	18	77.5
Team C	58	**47**	11	125.7
Team D	58	41	17	170.7
Team E	49	42	7	151.8
Average (SD)	60.5 (6.95)	45 (2.76)	15.5 (5.32)	113.6 (41.76)

Our approach (labeled USC) is compared with approaches by the five other teams (teams A–E). Best performance from each category is shown in bold. The last line of the table shows the average performance across teams and the SD

5.1 Limitations of the approach

The first and most important lesson we learned in this project is that we are nowhere close to truly autonomous manipulation. While we developed modules that work very well in practice, our approach still relies too much on models of the objects to be grasped or manipulated. It is worth mentioning, however, that model-free grasp selection has recently been successfully implemented on our system in Herzog et al. (2013). In addition, we do not have a systematic method to derive a state machine for a particular task. Instead, the control flow of each manipulation task had to be engineered by hand. This means that unexpected situations which were not accounted for in the design cannot be handled. For example, when trying to turn on a flashlight lying on its side, it would first need to be placed upright before turning it on. Such situations could be handled by incorporating a higher level task planner that can automatically produce the right sequence of motion primitives to execute (for example Kaelbling and Lozano-Pérez 2011). Another missing feature of our approach is that we do not exploit the full richness of the sensory information available. For example, we made little use of the touch sensors of the fingers and all the other sensors are mainly used at the control level or for perceiving objects but are rarely used to create very reactive behaviors (i.e. to quickly re-plan when something goes wrong). Switching controllers is another aspect that we did not address completely, approaches such as Kröger and Wahl (2010) could be used to improve this aspect. Possible improvements that are related to the platform include having torque sensing at the joint level and a higher control bandwidth to implement more sophisticated controllers.

5.2 Advantages of the approach

The great advantage of our approach is the intrinsic variable compliance of the robot. The robot is not just compliant to be compliant but its stiffness was constantly varied accord-

ing to the task, in order to guarantee precision when needed. Another strength of our approach is that we exploit the force sensors available to directly regulate contact interaction with objects or the environment. With this approach, it was possible to create robust grasps without ever computing grasp points but merely a rough grasp pose. The fingers of the hand would automatically adapt to the shape of the object and it would still be possible to increase their stiffness when it was necessary. We do not claim that computing grasp points is never necessary but we think that using compliant approaches allow to significantly reduce the necessity for precision when grasping a wide range of objects.

Another interesting aspect of our approach is the use of optimization tools for motion planning and cost functions to generate motion plan that would be more convenient to execute a given task. Using such approaches guaranteed that motions were very similar between experiments and that we would find solutions that were locally optimal for the task to be executed (e.g. maximum manipulability).

The use of motion primitives was also very useful, as it was possible to quickly design manipulation tasks by combining these primitives together. We will argue that such an approach would also be interesting for higher-order reasoning to automatically synthesize manipulation tasks.

5.3 Future research directions

In the following we try to describe several on-going and future research directions we believe are key to the development of really autonomous manipulation. They all constitute complements to the proposed approach.

Automatic acquisition of motion primitives Primitives of motion have been very useful in our approach. However, their acquisition needs to be fully automated with as little engineering required as possible. Noticable progress has been achieved on learning movement primitives from demonstration (e.g. Ijspeert et al. 2003; Calinon et al. 2007) as well as on further refinining learned movements with reinforcement learning (e.g. Peters and Schaal 2008; Kalakrishnan et al. 2011). However, many open research questions remain: For example, it remains unclear how to automatically and robustly sequence several movement primitives to accomplish more complex manipulation tasks. A promising first step towards incrementally creating more complex behaviors has been proposed in Niekum et al. (2013). Nevertheless, especially in the context of manipulation, sensor feedback (e.g. haptic and visual) should not be neglected and remembered along with executed movements (Pastor et al. 2013, 2012). Therefore, how a library of movement primitives should be maintained and updated is a question of particular interest. Finally, previous work oftentimes used a rather small set of movement primitives (e.g. Pastor et al. 2012) to generate a particular manipulation behavior. Thus, applicability to a more general setting and generating manipulation behaviors from motion libraries of a significant size has yet to be demonstrated.

Truly reactive behaviors One major frustration when doing manipulation tasks is that the robot behavior lacks real reactive capabilities. Indeed, compliance and active force control can improve the robustness of the task in face of unexpected disturbances but such features can only deal with short term disturbances that do not deviate the execution of the task too far from the planned one. In other words, they can only deal with disturbances on a short time scale. On the other hand, motion planning can in principle tackle any type of disturbance by fully replanning a desired motion when an event occur. However, despite being able to generate relatively impressive motions these planners can be very slow (Hauser et al. 2008; Lengagne et al. 2010), or have never been used on real robots (Mordatch et al. 2012) and usually do not take into account the richness of the information created by the available sensors. We can think of such planners as operating on a slow time scale. In our view, we need reactive behavior capabilities such that each motion primitive is attached with local reactive capabilities that use the richness of the sensory information available to adapt online the motion plan as unexpected events occur. We demonstrated in Pastor et al. (2011) preliminary results along these lines where the reactive motion plays a role at a time scale in between the feedback controllers and the motion planners. Model predictive control could also be a way to solve such problems, however the complexity and high-dimensionality of the tasks usually make such approaches intractable. In general, we believe that the exploitation of the richness of sensory information available on robots is a crucial element for autonomous manipulation. It seems that even with the sparse sensor data available on current robots there are very few, if any, algorithms able to exploit this information to control motion. The arrival of denser sensor information (e.g. tactile sensors, artificial skin) makes this issue even more important.

Coupling and feedback between modules The approach we proposed is still very sequential, in the sense that we still have a sense-plan-act paradigm, which is coordinated by a state machine. Moreover, all the different modules essentially function in isolation from each other. We will argue that a purely sequential approach (i.e. a pipeline), while being conceptually simpler to design, is not the right path to autonomous manipulation or to complex behaviors in general. Parallel approaches where the robot constantly perceives, plans and controls on multiple time-scales are more promising. In addition, we need more communication and coupling between modules. A natural first step that we are

currently implementing in our system is to have visual servoing capabilities, where the hand and the object to manipulate are constantly tracked by the visual system and their relative positions are used as a feedback term for our controllers. A tighter integration of perception (from various sensor modalities) with the control systems is a promising future research direction. It would also be desirable to increase the communication between planning and control up to the point where the boundary between both becomes blurred. Such a view is closely related to the idea of reactive behaviors that we discussed in the previous section and seems very much in line with current theories in complex systems.

Model free As we mentioned earlier, our approach still relies too much on hand crafted models. The use of models in our case was strongly motivated by the structure of the ARM project. Its competitive nature, the need for robust performance and the availability of models were the reasons we choose to rely so much on models. However, it seems reasonable to think that in a completely autonomous setting, it is not possible to rely on such knowledge and that both model-free approaches and approaches that automatically build models are to be researched more carefully. There have been several contributions that have demonstrated grasp planning without object models (Bohg and Kragic 2010; Hsiao et al. 2010; Li and Pollard 2005). One of particular interest was successfully implemented in our system (Herzog et al. 2013, 2012), where the conjunction of our force controlled grasping and the model-free selection of grasp poses led to very high grasping performance.

Automatic high-level sequencing of actions The last direction of research is methods to automatically sequence actions to be taken in order to achieve a desired manipulation task. It is a particularly difficult task, as it requires a lot of abstract knowledge on the type of task to be achieved, its goals and how success can be measured for each of its subtask. Approaches such as Kaelbling and Lozano-Pérez (2011) or Toussaint et al. (2010) are examples of research toward this goal. We conjecture that the use of motion primitives and their associated success measures can help tremendously in this direction, for both selecting the next action and to simplify the whole high-level planning problem. Encouraging preliminary results in this direction, using our architecture, have been demonstrated in Pastor et al. (2012).

6 Conclusion

In this paper we proposed a system architecture for robotic grasping and manipulation. At the core of the approach is the use of advanced force/torque controllers and optimization methods for motion planning. Though not ideal yet, our system has demonstrated robust performance in a wide variety of manipulation tasks and has proven competitive against other high quality approaches. Based on the lessons learned from our system, we have sketched a potential research program that we believe could lead us towards truly autonomous manipulation.

Acknowledgments This research was supported in part by National Science Foundation Grants ECS-0326095, IIS-0535282, IIS-1017134, CNS-0619937, IIS-0917318, CBET-0922784, EECS-0926052, CNS-0960061, the DARPA Program on Autonomous Robotic Manipulation, the Army Research Office, the Okawa Foundation, the ATR Computational Neuroscience Laboratories, and the Max-Planck-Society. The authors would also like to thank the anonymous reviewers for their constructive comments.

References

An, C., Atkeson, C., & Hollerbach, J. (1985). Estimation of inertial parameters of rigid body links of manipulators. In *IEEE Conference on Decision and Control* (pp. 990–995).

Asfour, T., Regenstein, K., Azad, P., Schroder, J., Bierbaum, A., Vahrenkamp, N., et al. (2006). Armar-III: An integrated humanoid platform for sensory–motor control. In *IEEE-RAS International Conference on Humanoid Robots* (pp. 169–175).

Atkeson, C., An, C., & Hollerbach, J. (1985). Rigid body load identification for manipulators. In *IEEE Conference on Decision and Control* (pp. 996–1002).

Axelrod, B., & Huang, W. (2012). Improving hand-eye calibration for robotic grasping and manipulation. In *IEEE International Conference on Technologies for Practical Robot Applications* (pp. 121–126).

Bagnell, J. A. D., Cavalcanti, F., Cui, L., Galluzzo, T., Hebert, M., Kazemi, M., et al. (2012). An integrated system for autonomous robotics manipulation. In *IEEE/RSJ International Conference on Intelligent Robots and Systems* (pp. 2955–2962).

Balasubramanian, R., Xu, L., Brook, P. D., Smith, J. R., & Matsuoka, Y. (2012). Physical human interactive guidance: Identifying grasping principles from human-planned grasps. *IEEE Transactions on Robotics*, *28*(4), 899–910.

Beetz, M., Klank, U., Kresse, I., Maldonado, A., Mösenlechner, L., Pangercic, D., et al. (2011). Robotic roommates making pancakes. In *IEEE-RAS International Conference on Humanoid Robots*.

Berenson, D., Srinivasa, S. S., Ferguson, D., Collet, A., & Kuffner, J. (2009). Manipulation planning with workspace goal regions. In *IEEE International Conference on Robotics and Automation*.

Bicchi, A. (1995). On the closure-properties of robotic grasping. *The International Journal of Robotics Research*, *14*(4), 319–334.

Bicchi, A., & Kumar, V. (2000). Robotic grasping and contact: A review. In *IEEE International Conference on Robotics and Automation* (pp. 348–353).

Bohg, J., & Kragic, D. (2010). Learning grasping points with shape context. *Robotics and Autonomous Systems*, *58*, 362–377.

Bohg, J., Morales, A., Asfour, T., Kragic, D. (2013). Data-driven grasp synthesis: A survey. In *IEEE Transactions on Robotics* (in submission).

Bruyninckx, H., & De Schutter, J. (1996). Specification of force-controlled actions in the "task frame formalism": A synthesis. *IEEE Transactions on Robotics and Automation*, *12*(4), 581–589.

Calinon, S., Guenter, F., & Billard, A. (2007). On learning, representing, and generalizing a task in a humanoid robot. *IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics*, *37*(2), 286–298.

Chiaverini, S. (1997). Singularity-robust task-priority redundancy resolution for real-time kinematic control of robot manipulators. *IEEE Transactions on Robotics and Automation*, *13*(3), 398–410.

Chiaverini, S., & Sciavicco, L. (1993). The parallel approach to force/position control of robotic manipulators. *IEEE Transactions on Robotics and Automation*, *9*(4), 361–373.

Chiaverini, S., Siciliano, B., & Villani, L. (1999). A survey of robot interaction control schemes with experimental comparison. *IEEE/ASME Transactions on Mechatronics*, *4*(3), 273–285.

Chitta, S., Jones, E., Ciocarlie, M., & Hsiao, K. (2012). Perception, planning, and execution for mobile manipulation in unstructured environments. In *Special Issue on Mobile Manipulation: IEEE Robotics and Automation Magazine* (Vol. 19).

Diankov, R., Ratliff, N., Ferguson, D., Srinivasa, S., Kuffner., J., (2008). Bispace planning: Concurrent multi-space exploration. In *Robotics: Science and Systems (R:SS)*.

Featherstone, R. (1987). *Robot dynamics algorithms*. Boston: Kluwer Academic Publishers.

Hauser, K., Bretl, T., Latombe, J. C., Harada, K., & Wilcox, B. (2008). Motion planning for legged robots on varied terrain. *The International Journal of Robotics Research*, *27*, 1325–1349.

Hauser, K., & Ng-Thow-Hing, V. (2010). Fast smoothing of manipulator trajectories using optimal bounded-acceleration shortcuts. In *IEEE International Conference on Robotics and Automation*.

Herzog, A., Pastor, P., Kalakrishnan, M., Righetti, L., Asfour, T., & Schaal, S. (2012). Template-based learning of grasp selection. In *IEEE International Conference on Robotics and Automation* (pp. 2379–2384).

Herzog, A., Pastor, P., Kalakrishnan, M., Righetti, L., Bohg, J., Asfour, T., et al. (2013). Learning of grasp selection based on shape-templates. In *Autonomous Robots Journal Special Issue: Autonomous Grasping and Manipulation*. doi:10.1007/s10514-013-9366-8

Hogan, N. (1985). Impedance control: An approach to manipulation. *Journal of Dynamic Systems, Measurement, and Control (Transactions of the ASME)*, *107*(1), 1–24.

Hollerbach, J., Khalil, W., & Gautier, M. (2008). Model identification. In B. Siciliano & O. Khatib (Eds.), *Springer Handbook of Robotics* (Chap. 14, pp. 321–344). Berlin: Springer.

Hsiao, K., Chitta, S., Ciocarlie, M., & Jones, E. (2010). Contact-reactive grasping of objects with partial shape information. In *IEEE/RSJ International Conference on Intelligent Robots and Systems* (pp. 1228–1235).

Hubert, U., Stückler, J., & Behnke, S. (2012). Bayesian calibration of the hand-eye kinematics of an anthropomorphic robot. In *IEEE-RAS International Conference on Humanoid Robots*.

Hudson, N., Howard, T., Ma, J., Jain, A., Bajracharya, M., Myint, S., et al. (2012). End-to-end dexterous manipulation with deliberate interactive estimation. In *IEEE International Conference on Robotics and Automation* (pp. 2371–2378).

Ijspeert, A. J., Nakanishi, J., & Schaal, S. (2003). Learning attractor landscapes for learning motor primitives. In *Advances in Neural Information Processing Systems* (pp. 1547–1554).

Kaelbling, L. P., & Lozano-Pérez, T. (2011). Hierarchical task and motion planning in the now. In *IEEE International Conference on Robotics and Automation* (pp. 1470–1477).

Kalakrishnan, M., Chitta, S., Theodorou, E., Pastor, P., & Schaal, S. (2011). STOMP: Stochastic trajectory optimization for motion planning. In *IEEE International Conference on Robotics and Automation* (pp. 4569–4574).

Kalakrishnan, M., Pastor, P., Righetti, L., & Schaal, S. (2013). Learning objective functions for manipulation. In *IEEE International Conference on Robotics and Automation*.

Kalakrishnan, M., Righetti, L., Pastor, P., & Schaal, S. (2011). Learning force control policies for compliant manipulation. In *IEEE/RSJ International Conference on Intelligent Robots and Systems* (pp. 4639–4644).

Kazemi, M., Valois, J. S., Bagnell, J. A., Pollard, N. (2012). Robust object grasping using force compliant motion primitives. In *Proceedings of Robotics: Science and Systems*.

Kröger, T., & Wahl, F. M. (2010). Online trajectory generation: Basic concepts for instantaneous reactions to unforeseen events. *IEEE Transactions on Robotics*, *26*(1), 94–111.

Lengagne, S., Mathieu, P., Kheddar, A., & Yoshida, E. (2010). Generation of dynamic motions under continuous constraints: Efficient computation using B-Splines and Taylor polynomials. In *IEEE/RSJ International Conference on Intelligent Robots and Systems* (pp. 698–703).

Li, Y., & Pollard, N. (2005). A shape matching algorithm for synthesizing humanlike enveloping grasps. In *IEEE-RAS International Conference on Humanoid Robots* (pp. 442–449).

Majarena, A. C., Santolaria, J., Samper, D., & Aguilar, J. J. (2010). An overview of kinematic and calibration models using internal/external sensors or constraints to improve the behavior of spatial parallel mechanisms. *Sensors*, *10*(11), 10256–10297.

Miller, A., & Allen, P. (2004). Graspit!: A versatile simulator for robotic grasping. *IEEE Robotics and Automation Magazine*, *11*(4), 110–122.

Mordatch, I., Todorov, E., & Popović, Z. (2012). Discovery of complex behaviors through contact-invariant optimization. In *Eurographics/ACM SIGGRAPH Symposium on Computer Animation* (pp. 1–8).

Nakanishi, J., Cory, R., Mistry, M., Peters, J., & Schaal, S. (2008). Operational space control: A theoretical and empirical comparison. *The International Journal of Robotics Research*, *27*(6), 737–757.

Niekum, S., Chitta, S., Marthi, B., Osentoski, S., Barto, A. G., (2013). Incremental semantically grounded learning from demonstration. In *Robotics: Science and Systems (R:SS)*.

Pastor, P., Kalakrishnan, M., Binney, J., Kelly, J., Righetti, L., Sukhatme, G., et al. (2013). Learning task error models for manipulation. In *IEEE International Conference on Robotics and Automation*.

Pastor, P., Kalakrishnan, M., Chitta, S., Theodorou, E., & Schaal, S. (2011). Skill learning and task outcome prediction for manipulation. In *IEEE International Conference on Robotics and Automation* (pp. 3828–3834).

Pastor, P., Kalakrishnan, M., Meier, F., Stulp, F., Buchli, J., Theodorou, E., et al. (2013). From dynamic movement primitives to associative skill memories. *Robotics and Autonomous Systems*, *61*(4), 351–361.

Pastor, P., Kalakrishnan, M., Righetti, L., & Schaal, S. (2012). Towards associative skill memories. In *IEEE-RAS International Conference on Humanoid Robots*.

Pastor, P., Righetti, L., Kalakrishnan, M., & Schaal, S. (2011). Online movement adaptation based on previous sensor experiences. In *IEEE International Conference on Robotics and Automation* (pp. 365–371).

Peters, J., & Schaal, S. (2008). Natural actor critic. *Neurocomputing*, *71*(7–9), 1180–1190.

Pradeep, V., Konolige, K., & Berger, E. (2010). Calibrating a multi-arm multi-sensor robot: A bundle adjustment approach. In *International Symposium on Experimental Robotics*.

Quigley, M., Gerkey, B., Conley, K., Faust, J., Foote, T., Leibs, J., et al. (2009). Ros: An open-source robot operating system. In *IEEE International Conference on Robotics and Automation; Workshop on Open Source Software*.

Raibert, M., & Craig, J. (1981). Hybrid position/force control of manipulator. *Journal of Dynamics Systems, Measurement and Control*, *103*(2), 126–133.

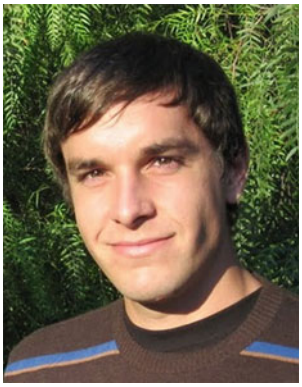
Ratliff, N., Zucker, M., Bagnell, J., & Srinivasa, S. (2009). CHOMP: Gradient optimization techniques for efficient motion planning. In: *IEEE International Conference on Robotics and Automation* (pp. 12–17).

Rusu, R. B., Şucan, I. A., Gerkey, B., Chitta, S., Beetz, M., & Kavraki, L. E. (2009). Real-time perception guided motion planning for a

personal robot. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*.
Schaal, S., (2009). *The SL simulation and real-time control software package*. Technical Report. University of Southern California. http://www-clmc.usc.edu/publications/S/schaal-TRSL.pdf
Sciavicco, L., & Siciliano, B. (2000). *Modelling and control of robot manipulators*. Berlin: Springer-Verlag.
Srinivasa, S., Ferguson, D., Helfrich, C., Berenson, D., Romea, A. C., Diankov, R., et al. (2010). HERB: A home exploring robotic butler. *Autonomous Robots*, *28*(1), 5–20.
Stückler, J., Badami, I., Droeschel, D., Gräve, K., Holz, D., McElhone, M., Nieuwenhuisen, M., Schreiber, M., Schwarz, M., Behnke, S., (2013). *Nimbro@home: Winning team of the robocup@home competition 2012*. RoboCup 2012, Robot Soccer World Cup XVI.
Toussaint, M., Plath, N., Lang, T., Jetchev, N., (2010). Integrated motor control, planning, grasping and high-level reasoning in a blocks world using probabilistic inference. In *IEEE International Conference on Robotics and Automation* (pp. 385–391).
Whitney, D. E. (1977). Force feedback control of manipulator fine motions. *Journal of Dynamic Systems, Measurement, and Control (Transactions of the ASME)*, *99*(2), 91.
Ye, Q. (1988). The signed Euclidean distance transform and its applications. In *International Conference on Pattern Recognition* (pp. 495–499).
Yoshikawa, T. (2000). Force control of robot manipulators. In *IEEE International Conference on Robotics and Automation* (pp. 220–226).

Ludovic Righetti is a group leader at the Max-Planck Institute for Intelligent Systems (Tubingen, Germany) and has been a postdoctoral fellow at the Computational Learning and Motor Control Lab (University of Southern California) between March 2009 and August 2012. He studied at the Ecole Polytechnique Federale de Lausanne where he received a Diploma in Computer Science (eq. MSc) in 2004 and a Doctorate in Science in 2008. His doctoral thesis was awarded the 2010 Georges Giralt Ph.D. Award given by the European Robotics Research Network (EURON) for the best robotics thesis in Europe. His research focuses on the generation and control of movements for autonomous robots, with a special emphasis on legged locomotion and manipulation.

Mrinal Kalakrishnan received his M.S. in Computer Science in 2008 from the University of Southern California, where he is also currently a Ph.D. candidate. His research focuses on the development of machine learning techniques to improve the performance of motion planners for autonomous robots in challenging domains such as rough terrain locomotion and autonomous grasping and manipulation. He is also interested in designing and learning controllers that use sensory and visual feedback loops to achieve compliant yet robust behavior.



Peter Pastor is currently finishing his Ph.D. at the Computational Learning and Motor Control Lab at the University of Southern California (USC). He received his Diploma in Computer Science from the Karlsruhe Institute of Technology in 2008. He also received a M.Sc. in Computer Science from USC in 2010. His research interests include robotic grasping and manipulation as well as machine learning.

Jonathan Binney receieved a B.S in Computer Science from the University of California, Los Angeles in 2006 and a Ph.D. in Computer Science from the University of Southern California in 2012. He is currently a research engineer working on planning and perception at Willow Garage.

Jonathan Kelly is an Assistant Professor at the University of Toronto Institute for Aerospace Studies. During 2011, he was a postdoctoral researcher in the Computer Science Department at the University of Southern California, working on the DARPA ARM Project. He received his M.S. in 2005 and his Ph.D. in 2011, both from the University of Southern California. At USC, he was an Annenberg Fellow. His research interests include robot navigation, mapping, and multimodal sensor fusion. He is a member of the IEEE.

Randolph C. Voorhies is a Ph.D. student in Computer Science at the University of Southern California where he also earned a Bachelors degree in Computer Science, and a Masters degree in Computer Science and Intelligent Robotics. His main interests are computer vision for mobile robotics as well as point cloud processing for localization and navigation.

Gaurav S. Sukhatme is Professor and Chairman of Computer Science (joint appointment in Electrical Engineering) at the University of Southern California (USC). He received his undergraduate education at IIT Bombay in Computer Science and Engineering, and M.S. and Ph.D. degrees in Computer Science from USC. He is the director of the USC Robotic Embedded Systems Laboratory which he founded in 2000. His research interests are in robot networks, embedded networks, aquatic robots, and robot perception. He has published extensively in these and related areas. Sukhatme has served as PI on numerous NSF, DARPA and NASA grants. He is a Co-PI on the Center for Embedded Networked Sensing (CENS), an NSF Science and Technology Center. He is a fellow of the IEEE and a recipient of the NSF CAREER award and the Okawa foundation research award.

Stefan Schaal is Professor of Computer Science, Neuroscience, and Biomedical Engineering at the University of Southern California, and a Founding Director of the Max-Planck-Institute for Intelligent Systems in Tuebingen, Germany. Before joining USC, Dr. Schaal was a postdoctoral fellow at the Department of Brain and Cognitive Sciences and the Artificial Intelligence Laboratory at MIT, and an Adjunct Assistant Professor at the Georgia Institute of Technology and at the Department of Kinesiology of the Pennsylvania State University. He was also an Invited Researcher at the ATR Computational Neuroscience Laboratory in Japan, where he held an appointment as Head of the Computational Learning Group during an international ERATO Project, the Kawato Dynamic Brain Project (ERATO/JST). Dr. Schaal's research interests include topics of statistical and machine learning, neural networks, computational neuroscience, functional brain imaging, nonlinear dynamics, nonlinear control theory, and biomimetic robotics. He applies his research to problems of artificial and biological motor control and motor learning, focusing on both theoretical investigations and experiments with human subjects and anthropomorphic robot equipment.