2012 IEEE International Conference on Robotics and Automation
RiverCentre, Saint Paul, Minnesota, USA
May 14-18, 2012

End-to-End Dexterous Manipulation with Deliberate Interactive Estimation

Nicolas Hudson, Thomas Howard, Jeremy Ma, Abhinandan Jain,
Max Bajracharya, Steven Myint, Calvin Kuo, Larry Matthies, Paul Backes[1]
Paul Hebert, Thomas Fuchs, Joel Burdick[2]
[1]**Jet Propulsion Laboratory, Pasadena, CA**
[2]**California Institute of Technology, CA**

Abstract— **This paper presents a model based approach to autonomous dexterous manipulation, developed as part of the DARPA Autonomous Robotic Manipulation (ARM) program. The developed autonomy system uses robot, object, and environment models to identify and localize objects, and well as plan and execute required manipulation tasks. Deliberate interaction with objects and the environment increases system knowledge about the combined robot and environmental state, enabling high precision tasks such as key insertion to be performed in a consistent framework. This approach has been demonstrated across a wide range of manipulation tasks, and in independent DARPA testing achived the most successfully completed tasks with the fastest average task execution of any evaluated team.**

I. INTRODUCTION

Autonomous Robotic Manipulators have the potential to increase manufacturing efficiency, provide in home care, and reduce the risk to humans in hazardous situations. Today's fielded hardware systems consist either of precise devices working in known environments, such as manufacturing, which incur laborious set up costs to execute a single repetitive task, or are low precision teleoperated or scripted systems (EOD robots, Robotic Explorers) which are highly constrained by power, weight and computation.

The current challenge in autonomous robotic manipulation is to approach the capabilities of dedicated one-off manipulators in known environments with versatile, inexpensive and ubiquitous manipulator systems which can operate in a range of environments with only high level human input.

We adopt a sensor driven, model-based approach to autonomous robotic manipulation, which continually uses environmental interactions to update the system (robot and environment) state, and then use these state estimates for planning and control.

The developed approach is motivated specifically by the DARPA Autonomous Robotic Manipulation (ARM) program [1], [2], in which teams are competing to create general manipulation software which can complete a range of grasping and manipulation tasks. Developed software is run by an independent test team on the ARM robot (Figure 1), and must be capable of completing tasks regardless of how or where objects are placed in the environment.

Required tasks include grasping objects such as a screwdriver, a shovel, a water-bottle, and a briefcase, and a range of manipulation tasks including drilling a hole, unlocking and opening a door, and hanging up a phone. This paper describes an end-to-end autonomy system which is capable of achieving all of these tasks with a consistent and rational approach.

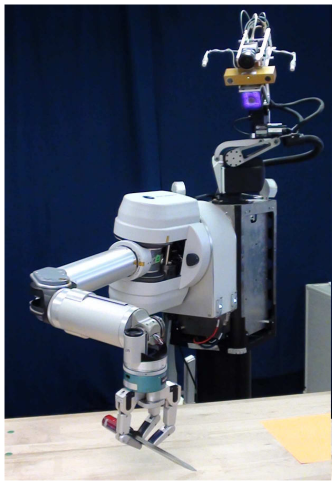


Fig. 1. The ARM Robot consists of a 7-DOF Barrett WAM Arm, a 4-DOF Barrett hand and a 4 DOF neck. Sensor include four (finger and palm) tactile arrays, finger strain gauges, a 6-DOF wrist force-torque sensor, and a head mounted high resolution camera (AVT GC2450C Prosilica), stereo pair (Point Grey Bumblebee2), and depth camera (Swiss Ranger 4000)

Figure 1 and 2 depict picking up a screwdriver and unlocking a door. Traditional approaches to manipulation require high system accuracy to complete these tasks. Inserting a key into a lock and grasping a screwdriver (with minor axis dimensions of 2 cm) from a table requires accuracy of approximately 1mm and 1cm respectively. The ARM robot is not capable of achieving these tasks using open loop task execution. Specifically, if ARM robot manipulator was commanded to move to a point as seen in the camera frame, the end effector error would likely be between 1-3 cm. This large error is due to component error stacking in the system, and is likely to be representative of fieldable systems. In the case of the ARM robot, errors accumulate from low sensor resolution, (320x240 depth sensors), unmodeled deformation of the neck under gravity, and the lack of output encoders on the WAM arm which is tendon driven. Other difficult aspects to the required manipulation tasks are reliable object localization, as objects such as a water-bottle are difficult to

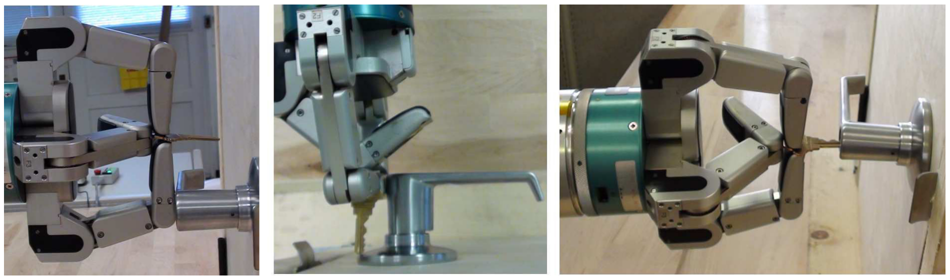


Fig. 2. Inserting a key and unlocking a door: First the finger knuckles move into contact the with the door handle on both the front (left image) and side face (center image), to provide relative hand to handle localization. The key is then inserted using force control and dither motions.

detect, and movement of large objects such as a shovel or briefcase, where collision free motion is required.

This paper describes several advancements in demonstrated manipulation autonomy. First, we utilize a single end-to-end autonomy architecture capable of a diverse range of manipulation tasks. All of the system components including perception, modeling, planning, estimation and control are general, and are not changed for each task. The only human input, or system difference is the specification of a xml task description such as 'pick up the screwdriver', or 'open the door'. Second, a consistent manipulation strategy (Section III) of deliberate interaction with the environment to update system state is described. Third, this approach has been demonstrated in competitive, independent DARPA ARM testing (Section IV), where both the fastest task execution times, and the highest task reliability were achieved.

II. RELATED WORK

There are currently several examples in which component technologies have been combined to demonstrate an end-to-end autonomous manipulation capability. HERB [3] uses the generalized approach to tracking movable objects (GATMO) [4] during navigation. It then uses known appearance models of previously seen objects to recognize and localize them in indoor environments [5]. It plans force closure grasps of simple objects [6], such as mugs and bottles, and caging grasps for the specific case of door handles [7]. It then executes these actions open-loop, which is the source of most persistent errors. The STAIR project [8] has demonstrated grasping novel objects by learning visual features that correspond to grasp points [9], but since the features are local, the overall geometry or mechanics of the object are not considered. The arm motions are executed open loop, although optical proximity sensors have been used to adjust the finger and wrist positions [10]. However this does not address actual contact and relies on a good initial grasp pose. The STAIR project has also investigated mobile manipulation applied to door opening with a single, known door model [11]. El-E [12] uses a parallel jaw gripper to grasp objects designated by a laser pointer from above a flat surface by visually segmenting the object, and has simple behaviors to reposition and re-grasp an object based on range and pressure sensors. Behaviors for opening a door with force feedback have also been demonstrated [13], but they are programmed and tuned for the specific task. Willow Garage has demonstrated door opening and inserting a plug into a socket [14], but the behaviors are again very specific, including dedicated detection algorithms for known and engineered objects (orange sockets and checkerboard plugs). CMU has similarly addressed mobile manipulation of an assembly task with known objects tagged with a 2D bar-code and specific controller gains and thresholds [15].

III. APPROACH

Unlike most of the above work, we do not use a traditional sense-plan-act paradigm, where first the robot localizes objects, then planning and control are executed sequentially. This system instead uses continuous estimation of the entire system state to update and then re-plan actions, and also deliberately executes actions which increase system state knowledge. This is done not only for a specific task, but across all tasks.

In general the autonomy approach presented here conforms to standard system decompositions: Objects and the environment are first segmented, classified and localized using vision (*Perception* Sec. III-A). The robot state and environment models are used to produce optimal grasp sets, manipulation strategies and collision free trajectories (*Planning* Sec. III-D). As environmental interaction occurs, more sensors such as tactile, force, and strain sensors can be fused with visual sensing to provide updates of system state (*Estimation* Sec. III-B). Real-time execution of task objectives, using feedback from sensors and state estimation drives system actuators (*Control* Sec. III-E). Subsequent sections describe unique aspects of each system component, First however, a general manipulation strategy used for all ARM tasks is described.

For all ARM tasks, including drilling, unlocking, opening, actuating and grasping of various objects, a single strategy is utilized. Two diverse examples of picking up a screwdriver and unlocking a door are described in the general manipulation framework:

- **Non-Contact Perception**, in which only visual sensors are used to segment, classify and localize objects in the scene, and determine their pose relative to environmental constraints (such as a table or wall/door plane).
- **Approach**, where using the initial system state estimates, optimal arm, neck and finger trajectories are planned, to bring the manipulator near the object in

a manipulable configuration. In the case of the screwdriver on a table, the grasp set will include a caging grasp, where widely spaced fingers and the table prevent object escape. For key insertion, the hand is brought near the door handle.

- **Initial Contact / Re-localization**. When the manipulator is in the same field of view as the relevant objects, estimation of the arm position (arm-tracking) allows the control to provide more precise relative hand to object motions. For all tasks, a general contact strategy, where parts of the manipulator are moved into contact with the environment, is used to provide further relative localization. In the case of the screwdriver, moving to touch the table with outstretched fingers both provides vertical localization and contains the objects within a caging grasp. Where more precise localization is required, several motions into contact used. For key insertion (Figure 2), first the finger knuckles localize the door handle in depth, before moving (or sliding) to the top and side surfaces of the handle provide height and lateral localization. By recording a map of the contacts, the door handle model is used to precisely localize the key infront of the door handle.
- **Grasping or Manipulation**, once sufficient relative object localization has been achieved, controlled execution of the primary specified task objective (drilling, grasping, compression, actuation, etc.) is conducted. For grasping the screwdriver, fingers are closed while wrist force control keeps the fingers against the table surface, enveloping and then power-grasping the screwdriver. For key insertion, a dithering forward motion with force and torque control is used.

Thorough the manipulation process, object models are both updated and used: Each part of the manipulation process is planned using the most recent state information; Localization of objects utilizes 3-D geometric mesh representations for both iterative closes point matching and contour matching; The mass, estimated by the force sensor, and dimensions of objects are used for feedforward control. Unique aspects of each individual system component are now explained.

A. Perception

The perception system is required to segment, classify and localize the 6-DOF pose of objects in the environment using all the available visual sensors of the ARM robot. This includes both task manipulation objects, as well as obstacles or constraints (e.g. a door). We restrict discussion to operations in semi-known environments, for which 3-D geometric CAD models of objects in the environment are known a-priori. That is, we allow the pose and properties (mass, color, etc) of objects to be unknown and the environment (lighting, location) subject to change. However, the proposed approach is intended to provide a basis for manipulation with novel objects and environments. Because all ARM tasks are conducted in front of a table, the first step in perception is table plane estimation.

1) Table Plane Estimation & Map Generation: A standard RANSAC-based approach is used to estimate the dominant table plane from collected 3D point clouds. Once the plane is estimated, the 3D point clouds are used to populate a 2.5D elevation map which accumulates mean elevation and 3D point statistics into each grid-spaced cell. Once the map is populated with enough grid cells covering the workspace of the robot, segmentation of out-of-plane objects into 3D "blobs" follows using accumulated stereo and lidar points at various scan angles.

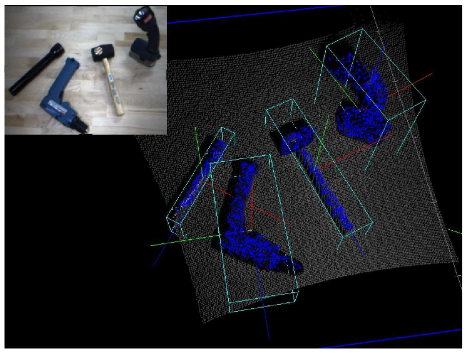


Fig. 3. Volume-Based Segmentation: The upper left image shows a view of the workspace from the left stereo camera, the gray points represent the lidar returns from the SwissRanger Lidar, the blue points represent out-of-plane points, and the bounding-boxes in blue are assigned using PCA about each segmented and clustered point cloud.

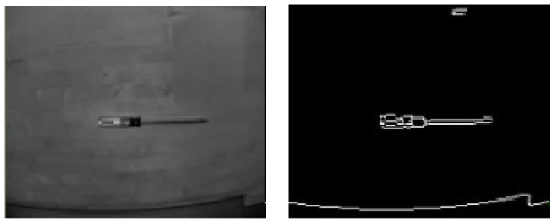


Fig. 4. Contour Segmentation: A screwdriver is shown in the flash-lidar intensity image (left) and the extracted contour image (right)

2) 3D Object Segmentation: The purpose of 3D object segmentation is to isolate a set of candidate 3D blobs in the robot workspace that can then later be classified and fitted against a 3D model for subsequent manipulation.

The range of objects specified in the ARM tasks requires the use of multiple segmentation algorithms. This is due to both: 1) the operating properties of the sensors themselves creating limitations of what can properly be sensed (multi-path effects in lidar can lead to misreadings; low resolution and poor lighting in the stereo image leads to low range accuracy); and 2) the appearance properties of the objects lend themselves to be easily detectable in one sensor and nearly undetectable in the other (e.g. a transparent water bottle is invisible in low resolution stereo imagery, though reliably detectable in the intensity image of the flash lidar).

The following segmentation routines are called in a ranked priority order – i.e. if one method of segmentation fails to extract any blobs, a second or even third method can be called as a fallback routine to ensure all viable blobs are segmented:

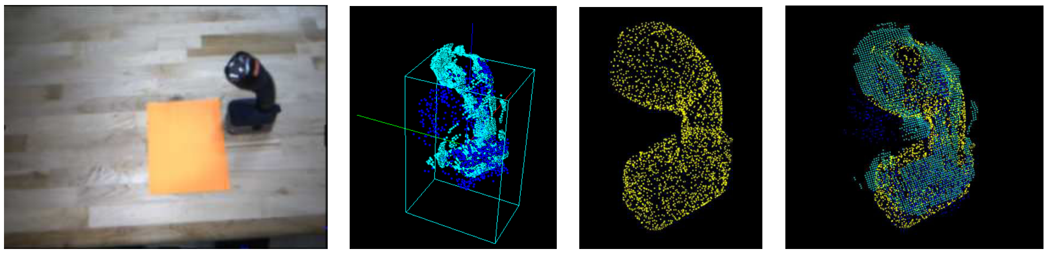


Fig. 5. (left-to-right): a snapshot of the scene as taken from the left stereo image, showing a blue flashlight; the extracted point clouds from stereo (cyan) and lidar (blue) with the PCA bounding box fitted around the points; the provided a-priori mesh model of the object to be manipulated shown in yellow dots; the resulting matched 6-DOF pose of the object using the Iterative Closest Point method

- **Volume-Based Segmentation** uses the generated 2.5D elevation map and cluster regions of large mean elevation into map cell segments. Each clustered segment carries cell statistics which are then used with PCA analysis to fit the correctly sized bounding box. Figure 3 shows the results of this method of object segmentation applied to a cluttered scene of multiple objects.
- **Contour-Based Segmentation** is useful for objects that have no discernable volume (e.g. a transparent water-bottle, a sideways lying screwdriver with a low profile). Edges are extracted from the collected lidar intensity images of each sensor scan (taken during an initial sensor coverage sequence) and are merged into contours, defined in the intensity image space. The collection of 3D points from stereo and lidar are reprojected into the lidar intensity image frame and checked against each extracted contour. All 3D points falling in the interior of a contour are associated with it and a final PCA analysis step determines the bounding box around each contour. Figure 4 illustrates the extracted contours from a test image involving a screwdriver that failed to be detected as a valid blob from the Volume-Based Segmentation approach, but was detected using the Contour-Based approach.
- **Color-Based Segmentation** is used for objects that have a strong color signature (such as a red ball or a white phone) yet lack good 3D point cloud coverage, i.e. textureless objects of a uniform color. This method maintains a 2.5D color map which stores an average RGB value for each cell from 3D colored stereo points and lidar points[1]. Segments in the color map that match the color of the desired object are isolated and used as a bitmask against the 2.5D elevation map to garner cell statistics and produce 3D blobs.

3) Classification and 6-DOF Pose Estimation: Classification is used to associate the desired manipulation object to the segmented 3d blobs in the workspace. Selection of the most likely candidate blob is done based primarily on a score of dimension matching; i.e. the blob whose bounding box dimensions most closely matches the model. Secondary metrics including color similarity can be used when the information is available.

[1]lidar points are colored by applying a simple projection mapping into the stereo camera frame and painting each pt with a mapped color pixel

The 6-DOF pose of an object is assigned by applying a refined Iterative-Closest Point (ICP) algorithm process to match the extracted point cloud of the blob against the known mesh model of the object. This process fixes a position and attitude to the object by minimizing the summed distance error between measurement points and model points. Figure 5 illustrates the basic premise of how ICP is implemented in our system.

B. Estimation

To provide continually improving estimates of the system state, and to take advantage of information gained from physical object contact, a state estimator is used to fuse multiple afferent sensory signals.

A significant decision in the state estimator is the choice of state. Our state is chosen to be $X = \{G_{PO}, G_{KV}\}$, where:

- G_{PO}, is the pose of the object (O) relative to the palm (P) frame of the manipulator,
- G_{KV}, is the pose correction of the palm/tool frame in kinematics, to the location of frame in the visual frame (V).

Conditioning on the palm-to-object state allows simple likelihood functions which incorporate hand sensors such as the force torque sensor, tactile pads and finger positions and strains. The kinematic to visual transform allows estimation of system error and bias including kinematics (K), sag, and joint biases for both the manipulator and neck. The SO3 state rotations are represented using angle-axis errors, giving $X \in R^{12}$.

The model based nature of the manipulation approach is again highlighted for state estimation. Here, the state distribution is always conditioned on a specific model $\mathscr{M}$ parameterization:

$$P(G_{PO}, G_{KV} | \mathscr{M}) \quad (1)$$

Ongoing work is addressing the issue of updating the model $\mathscr{M}$ online based observations, although this work is not addressed here. The object is modeled using a standard polygonal mesh. The manipulator is likewise represented, with each rigid link having a separate mesh.

Our algorithm makes use of an unscented kalman filter that fuses these sources together. The state prediction step is simplified due to the state choice, as once the object is grasped inside the hand, the state transition becomes identity and essentially a discrete random walk model.

$$\begin{bmatrix} G_{PO} \\ G_{KV} \end{bmatrix}_{i+1} = \begin{bmatrix} I & \mathbf{0} \\ \mathbf{0} & I \end{bmatrix} \begin{bmatrix} G_{PO} \\ G_{KV} \end{bmatrix}_{i} \quad (2)$$

Measurements that directly impact the object's state come from three sources: visual depth information (shape), silhouette information, and tactile information. Filtered point clouds are associated with manipulation objects, and a KD-tree is used to match to nearest points (vertices) on model. The innovation between the nearest model points and the sensed data points forms the likelihood in our measurement update and is similar to one step of the iterated closest point (ICP) algorithm.

In addition to visual information, tactile information can be utilized using the same update framework. The tactile arrays on the fingers and palm are thresholded to produce a binary contact measurement and combined with finger kinematics to produce 3D point measurements. The measurements, similar to 3D visual depth information, are correlated to point on the object mesh. An additional measurement includes the surface normals of the pads and object. The innovation for this measurement is to find the vertex normal corresponding to the closest point in the object's mesh as above, and compare it to the normal of the associated tactel.

Updates to the palm-pose correction G_{KV}, are referred to as *arm tracking*, as this transform can be directly observed from visual sensors. Observations from the high resolution prosilica camera, the lidar, and stereo camera are fused to create a robust arm tracker. Three separate likelihood functions are used. First the arm model is rendered in OpenGL and projected into a virtual monocular camera. A numerical optimizer is used to maximize the correlation to the silhouettes obtained in the observed prosilica image (Figure 6). Second, the flash lidar produces 3D point clouds for which we use a modified articulated iterated closest point (AICP), to match both the model and the sensed data points. Third, the stereo camera is used for template matching in an approach similar to [16], in which templates are located in both left and right cameras and then triangulated to obtain a measurement of the palm location.

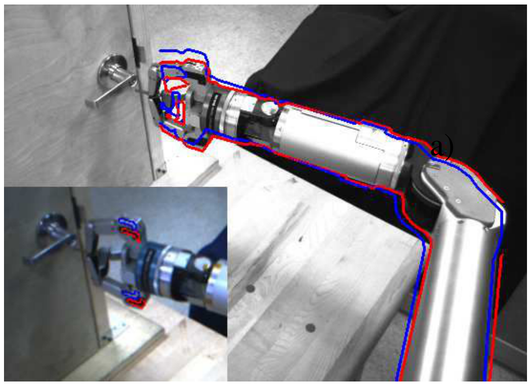


Fig. 6. a) Contour matching results showing the kinematic model outline (blue) with the corrected arm location (red). b) Template matching showing kinematic outline (blue) and corrected location (red).

For efficiency, applicable update steps of the UKF are serialized using the method of McManus et al. [17]. More details of the estimation framework can be found in [18].

C. World Modeling

A common world model with accompanying computational mechanics algorithms is used to support the planning, control, estimation, perception and calibration modules. This layer includes generic frame kinematics computations used by all the modules, as well as functions tailored to their specific needs. Beyond, the objective of consolidating the mechanics computations, the world modeling layer also allows refinements in model parameters to be updated and propagated consistently to all the computations used by the different modules. The computational mechanics layer is based on the *spatial operator algebra (SOA)* computatinoal dynamics architecture [19] which defines a family of operators and their combinations in recursive algorithms to efficiently carry out a broad family of robotics computations. For the ARM project, we have adapted the DARTS[2] software that is a general purpose implementation of the SOA architecture.

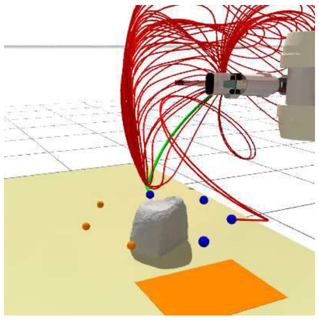
D. Manipulation Planning

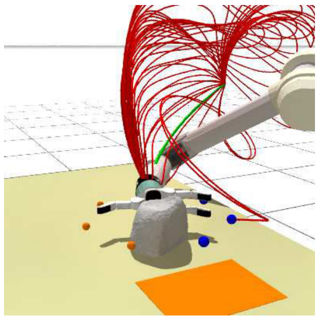
Manipulation planning generally divides into three problems: grasp planning, view planning, and motion planning. Grasp and view planning produce a set of terminal constraints for the degrees of freedom of the manipulator. Motion planning is the two-point boundary value problem of generating a collision-free trajectory that satisfies the constraint set and the differential functions that represent the robot's dynamics.

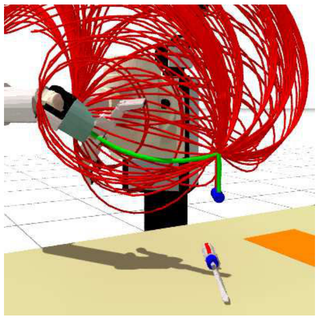
1) Grasp Set Generation: Contemporary hand planners typically form a set of potential grasps based on object geometry. GraspIt [20] achieves this efficiently by decomposing objects into individual shape primitives. Recent work has applied reinforcement learning with probabilistic distributions of object pose to determine grasps that are resistant to state estimation errors [21]. Our approach to grasp planning separates free-space grasping and table grasping (Figure 7). Our free-space grasp planner uses object-relative grasp sets produced off-line in a manner similar to GraspIt. Our table grasp planner alternatively leverages caging concepts to place the fingers in a configuration where a reactive controller can exploit the geometry of the table to grasp the object. Both methods produce a set of candidate wrist and finger configurations to grasp an object.

Each candidate grasp set is culled based on a lack of valid inverse kinematics solutions and collision with environmental objects. The rock and the screwdriver grasp sets shown in Figure 7 have 120 valid manipulator configurations at four wrist positions and 238 valid manipulator configurations at three manipulator positions respectively. Valid configurations are ranked based on a optimality and diversity metrics to compute an *N*-best manipulator constraint set for view and motion planning.

[2]Dynamics Algorithms for Real-Time Simulation (DARTS) (`http://dartslab.jpl.nasa.gov/`)


(a) free-space grasp set generation for a rock (pre-motion)


(b) free-space grasp set generation for a rock (post-motion)


(c) table grasp set generation for a screwdriver (pre-motion)

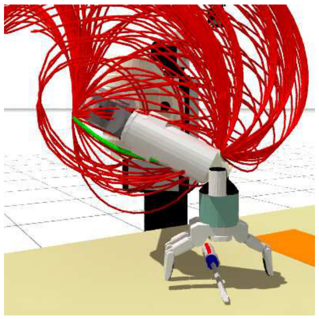

(d) table grasp for a screwdriver (post-motion)

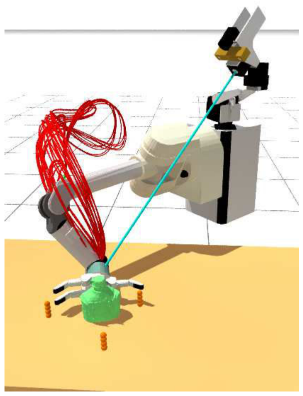
Fig. 7. Examples of grasp sets for two different objects. Free-space grasp sets use the object geometry to identify candidate manipulator configurations. Table grasp sets alternatively use the table and the object geometry to determine possible wrist configurations. Collision-free wrist poses with valid inverse kinematics solutions for the arm are shown as blue spheres. Invalid wrist poses are shown as orange spheres.

2) View Planning: The second half of constraint generation involves finding terminal configurations for the two pan-tilt units (PTU) that control the posture of the sensor head. Sensor head constraints are generally determined by optimizing a function that represents a non-occluded view of the wrist frame. A set of candidate PTU angles is found by sampling the two-dimensional space of lower PTU angles and solving for the upper PTU angles that center the camera frame on the wrist.

A neck constraint set representing maximum visibility of the wrist is determined by evaluating a cost function that penalizes occlusions with the manipulator. The neck constraints are added to the manipulator constraints to form a terminal constraint $\mathbf{x_c}$ composed of seven joint angles, four hand angles, and four neck angles. This is done for each manipulator constraint to form a terminal constraint set $(\mathbf{X_C}(t_f))$ for the motion planner (Equation 3).

$$\mathbf{X_C}(t_f) = \begin{bmatrix} \mathbf{x_{C,0}}(t_f) & \mathbf{x_{C,1}}(t_f) & \ldots & \mathbf{x_{C,N-1}}(t_f) \end{bmatrix}^T \qquad (3)$$
$$where \quad \mathbf{x_{C,i}}(t_f) = \begin{bmatrix} \theta_{\mathbf{arm}} & \theta_{\mathbf{hand}} & \theta_{\mathbf{neck}} \end{bmatrix}^T$$

3) Motion Planning: Motion planning for high degree-of-freedom systems generally divides into continuum optimization and graph-based approaches. CHOMP [22] uses covariant gradient descent to produce locally optimal motions by minimizing an objective function based on the canonical shape of the trajectory and the computed obstacle potential. STOMP [23] similarly refines an initial motion through iterative updates based on a weighted sum of perturbed trajectory parameters as opposed to an analytic or numeric gradient. Variations of the Rapidly-Exploring Random Tree (RRT) technique [24] are commonly used to find a collision-free path by randomly sampling in the action or state space towards the goal or goal region. Our approach extends state-space sampling techniques [25] for robotic torsos by solving for parameterized actions that satisfy intermediate and terminal joint constraints using the technique described in [26]. Extra degrees of freedom in the system are resolved by adding constraints to the maximum velocity of any single joint velocity and requiring that all parameterized functions start and end at the same time. This sampling-based approach differs from other contemporary work because it directly searches the parameterized space of desirable manipulator trajectories. Figure 9 shows a trajectory set generated for grasping a briefcase. Each trajectory in that set is composed of joint velocities defined by a sequence of two second-order or third-order parameterized splines. This parameterization is desirable because it is smooth and differentiable and naturally accelerates and decelerates to and from zero-velocity state constraints. The example in Figure 9 took 0.73 seconds to compute the constraint set and 2.08 seconds to generate 108 collision-free trajectories from 204 terminal state constraints on an Intel Quad-Core Xeon CPU with 6 GB of memory.


(a) a neck pose where the wrist is visible from the perception head

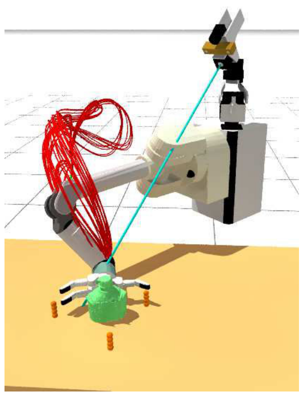

(b) a neck pose where the wrist is occluded from the perception head

Fig. 8. View planning for a 4-DOF neck. The view planner computes an optimal neck configuration for each manipulator constraint $\mathbf{x_{C,i}}(t_f)$ in the valid set by sampling in the two extra degrees of freedom.

E. Control

The control system is comprised of two parts: Task Frame Control, and Joint Level Control (Figure 10). The primary function of the control system is to execute task sub-behaviors, such as grasping a screwdriver from the table (Figure 1, 7) or turning a key in a lock.

To explain the task frame controller, we first define a behavior, an action, a control primitive, and an end condition.

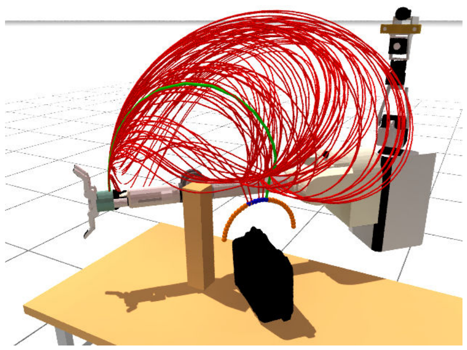


Fig. 9. Feasible, collision-free trajectories generated by the state-space sampling approach for briefcase grasping. The parameterized action used to generate all candidate trajectories is a composite of a joint-space and Cartesian-space motion.

We define a behavior as a sequence of actions. For instance, the unlock behavior (Figure 2) is comprised of the following actions: first a *move to touch* action is presses a finger knuckle into the door handle; second a *slide* action moves the key to the edge of the handle; eventually, as more information about the relative hand-to-handle pose is gathered the key can be positioned directly infront of the lock and an *insert* action can be performed.

Each action contains set of parametrized control primitives and end conditions. The 'move to touch' action is comprised of a Cartesian trajectory follower, a visual servoing controller which uses the output of the arm-tracker, and a force limit end condition.

The Generalized Compliant Motion (GCM) framework [27], is used to specify and merge control objectives. GCM is a task space controller which enables abstracted task definitions independent of system dynamics. In essence, GCM models transforms each control objective into the task frame, and then trades off control objectives by modeling each as a second order system. This essentially creates a system of spring-like responses centered at each control set-point, which 'pull' on the end effector. All manipulation actions for the ARM project are comprised of only four control primitives: force control, visual servoing, Cartesian tracking, and dithering. The output of GCM specifies the desired end-effector pose trajectory based on sensory feedback.

The desired end effector pose is then transformed into joint level torques based on a feedforward (inverse dynamics) torque, and distributed joint level PID tracking of the commanded position trajectory.

Object mass, interia and control objectives (such as force) create feedforward terms in addition to provided feedforward gravity compensation of the Barrett WAM arm.

Redundancy resolution for the 7-DOF arm optimizes an objective function which minimize the distance between planned and actual joint angles, and maximizes the distance of joints from their hard stops and singularities. This optimization occurs by searching over 200 nearby shoulder yaw angles every control loop (using inverse kinematics) and directly evaluating the desired objective function.

The GCM task frame controller is run at 100 Hz on a Linux computer, and produces feedforward torques and joint trajectories via inverse kinematics. The feedforward terms are added to the output of a PID position tracking loop which runs at 300Hz on a real-time operating system directly controlling the WAM arm. The maximum loop rates are limited by sensory data collection rates and WAM CAN bus bandwidth.

IV. EXPERIMENTAL RESULTS

As part of the DARPA Autonomous Robotic Manipulation (ARM) Software program, six teams were given identical robots (Figure 1) and an array of test objects. The results described here are from the programmatic end-of-phase tests, which evaluated 12 grasping and 6 manipulation tasks. Each team created algorithms and software, such as the approach described in this paper, and uploaded software to the ARM test facility, where independent test team evaluators ran each test with no developer input. Each of the grasping and manipulation tasks were run five times, each time with the object in a novel pose. Poses were chosen to test both manipulation and perception capabilities. For each task, the primary metric used for evaluation was completion (yes or no) with a secondary metric of completion time.

The grasping tasks were considered successful if the robot could pick an object clear off a table, and then place it on a specific target location, while maintaining a secure grasp. The 12 grasping objects were a: ball, shovel, maglite, floodlight, radio, pelican case, rock A, rock B, screwdriver A, screwdriver B, hammer A and hammer B. Objects suffixed with 'B', had not previously been seen by each team. In this later case, a canonical model was used which was not specific to the object.

The manipulation tasks were: turn on a flashlight, staple paper, hang up a phone, open a door, unlock a door with a key, and drill a block of wood at a specific target location.

TABLE I
DARPA ARM-S PHASE 1 TEST RESULTS.

Team	Sucesses (of 72)	Grasping (of 48)	Manipulation (of 24)	Average Time (s)
JPL	67	47	20	75.4
B	67	47	20	80.6
C	64	46	18	77.5
D	58	47	11	125.7
E	58	41	17	170
F	49	42	7	151.8

The JPL (Jet Propulsion Laboratory) team, using the approach described in this paper, was ranked the highest in independent DARPA testing using the specified completion metrics. Table 1 above shows the final evaluation results for each team. By request, the other teams have been kept anonymous. The scores reported by DARPA are from the 4 best runs of each task with the worst run thrown out.

In general the approach described in this paper provided robust results, even for grasping of novel objects such unknown screwdrivers or rocks, and difficult manipulation tasks such as key insertion. JPL achieved 4/4 on all tasks except grasping a floodlight (3/4), turning on the flashlight

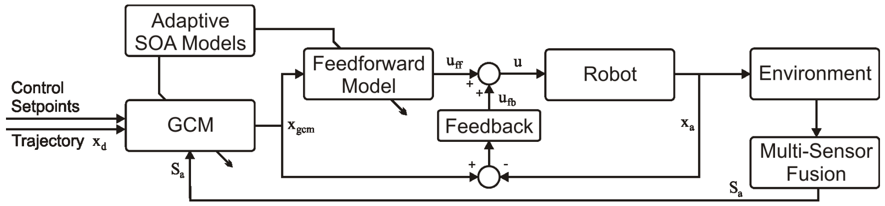


Fig. 10. The Generalized Compliant Motion (GCM) framework is used to enable robust execution of a wide variety of tasks. GCM produces task-frame motion by merging multiple task-space controllers with feedback from the system state (S_a). Task frame motion is translated into joint level position trajectories (x_{gcm}) using inverse kinematics, which are tracked by PD feedback loops (u_{fb}) on the WAM arm. PD output is combined with feedforward torques (u_{ff}) computed using inverse dynamics and gravity compensation models.

(3/4) and drilling (1/4). The floodlight (made out of white plastic for safety) was difficult to localize using vision, with poor stereo returns due to a monotone color, and poor lidar returns because of material properties. Furthermore, the lightweight floodlight would roll on the table with even a slight touch, making tactile feedback hard. Drilling was difficult as it required hitting a small red target on a block of wood, with no kinesthetic feedback or confirmation (as the target was only specified by color). In addition the drill was near the maximum payload of the WAM arm.

All teams found grasping objects significantly easier than the manipulation tasks. One common strategy used by all teams was to 'table' grasp objects (Figure 7d), where the finger tips first touch the table, and then scrape or slide along until object contact is made. For these table grasps, not only is the object physically constrained or caged against the table, but unique to the approach described here, the additional kinesthetic information can be used to update the object pose for future planning or control.

ACKNOWLEDGMENTS

The research described in this publication was carried out at the Jet Propulsion Laboratory, California Institute of Technology, with funding from the DARPA Autonomous Robotic Manipulation Software Track (ARM-S) program through an agreement with NASA.

REFERENCES

[1] Defense Advanced Research Projects Agency (DARPA), , "Autonomous Robotic Manipulation (ARM)," 2010. [Online]. Available: http://www.darpa.mil/Our_Work/DSO/Programs/Autonomous_Robotic_Manipulation_(ARM).aspx
[2] DARPA, "The ARM robot," 2010. [Online]. Available: http://thearmrobot.com/
[3] S. Srinivasa, D. Ferguson, C. Helfrich, D. Berenson, A. C. Romea, R. Diankov, G. Gallagher, G. Hollinger, J. Kuffner, , and J. M. Vandeweghe, "HERB: a home exploring robotic butler," *Autonomous Robots*, vol. 28, p. 520, January 2010.
[4] G. Gallagher, S. S. Srinivasa, J. A. Bagnell, , and D. Ferguson, "GATMO: a generalized approach to tracking movable objects." in *IEEE Int. Con. on Robotics and Automation*, May 2009.
[5] A. Romea, D. Berenson, S. Srinivasa, , and D. Ferguson, "Object recognition and full pose registration from a single image for robotic manipulation," in *IEEE International Conference on Robotics and Automation*, May 2009.
[6] D. Berenson and S. Srinivasa, "Grasp synthesis in cluttered environments for dexterous hands," in *IEEE-RAS International Conference on Humanoid Robots*, December 2008.
[7] R. Diankov, S. Srinivasa, D. Ferguson, and J. Kuffner, "Manipulation planning with caging grasps," 2008.
[8] M. Quigley, E. Berger, and A. Ng, "Stair: Hardware and software architecture," in *AAAI 2007 Robotics Workshop*, 2007.
[9] A. Saxena, J. Driemeyer, and A. Y. Ng, "Robotic grasping of novel objects using vision," *International Journal of Robotics Research (IJRR)*, vol. 27, p. 157173, Feb 2008.
[10] K. Hsiao, P. Nangeroni, M. Huber, A. Saxena, , and A. Y. Ng, "Reactive grasping using optical proximity sensors," in *IEEE International Conference on Robotics and Automation*, May 2009, p. 20982105.
[11] A. Petrovskaya and A. Y. Ng, "Probabilistic mobile manipulation in dynamic environments, with application to opening doors," in *International Joint Conference on Artificial Intelligence (IJCAI)*, 2007.
[12] A. Jain and C. C. Kemp, "EL-E: an assistive mobile manipulator that autonomously fetches objects from flat surfaces," *Autonomous Robots*, 2010.
[13] ——, "Behaviors for robust door opening and doorway traversal with a force-sensing mobile manipulator," in *RSS Manipulation Workshop: Intelligence in Human Environments*, 2008.
[14] W. Meeussen, M. Wise, S. Glaser, S. Chitta, C. McGann, P. Mihelich, E. Marder-Eppstein, M. Muja, V. Eruhimov, T. Foote, J. Hsu, R. Rusu, B. Marthi, G. Bradski, K. Konolige, B. Gerkey, and E. Berger., "Autonomous door opening and plugging in with a personal robot," in *International Conference on Robotics and Automation (ICRA)*, 2010.
[15] B. Hamner, S. Koterba, J. Shi, R. Simmons, and S. Singh, "An autonomous mobile manipulator for assembly tasks," *Autonomous Robots*, vol. 28, January 2010.
[16] M. Bajracharya, M. DiCicco, P. Backes, and K. Nickels, "Visual end-effector position error compensation for planetary robotics," *Journal of Field Robotics*, vol. 24, no. 5, pp. 399–420, 2007.
[17] C. McManus and T. Barfoot, "A serial approach to handling high-dimensional measurements in the sigma-point kalman filter," in *Proc. of Robotics: Science and Systems*, Los Angeles, CA, USA, June 2011.
[18] P. Hebert, N. Hudson, J. Ma, T. Howard, T. Fuchs, and J. Burdick, "Combined shape, appearance and silhouette for object manipulation," May 2012.
[19] A. Jain, *Robot and Multibody Dynamics: Analysis and Algorithms*. Springer, New York, 2010.
[20] A. Miller, S. Knoop, H. Christensen, and P. Allen, "Automatic grasp planning using shape primitives," 2003.
[21] F. Stulp, E. Theodorou, J. Buchli, and S. Schaal, "Learning to grasp under uncertainty," in *Proceedings of the 2011 International Conference on Robotics and Automation*, May 2011.
[22] N. Ratliff, M. Zucker, J. Bagnell, and S. Srinivasa, "CHOMP: gradient optimization techniques for efficient motion planning," in *Int. Con. on Robotics and Automation*, May 2009.
[23] M. Kalakrishnan, S. Chitta, E. Theodorou, P. Pastor, and S. Schaal, "STOMP: stochastic trajectory optimization for motion planning," in *International Conference on Robotics and Automation*, May 2011.
[24] D. Berenson, S. Srinivasa, D. Ferguson, and J. Kuffner, "Manipulation planning on constraint manifolds," in *Proceedings of the IEEE Int. Con. on Robotics and Automation*, May 2009.
[25] T. Howard, C. Green, A. Kelly, and D. Ferguson, "State space sampling of feasible motions for high-performance mobile robot navigation in complex environments," *Journal of Field Robotics*, vol. 25, no. 6-7, pp. 325–345, 2008.
[26] T. Howard and A. Kelly, "Rough terrain trajectory generation for wheeled mobile robots," *International Journal of Robotics Research*, vol. 26, no. 2, pp. 141–166, February 2007.
[27] P. G. Backes, "Dual-arm supervisory and shared control task description and execution," *Robotics and Autonomous Systems*, vol. 12, pp. 29–54, 1994.