Auton Robot
DOI 10.1007/s10514-009-9160-9

HERB: a home exploring robotic butler

Siddhartha S. Srinivasa · Dave Ferguson · Casey J. Helfrich · Dmitry Berenson · Alvaro Collet · Rosen Diankov · Garratt Gallagher · Geoffrey Hollinger · James Kuffner · Michael Vande Weghe

Received: 29 January 2009 / Accepted: 15 October 2009

Abstract We describe the architecture, algorithms, and experiments with *HERB*, an autonomous mobile manipulator that performs useful manipulation tasks in the home. We present new algorithms for searching for objects, learning to navigate in cluttered dynamic indoor scenes, recognizing and registering objects accurately in high clutter using vision, manipulating doors and other constrained objects using caging grasps, grasp planning and execution in clutter, and manipulation on pose and torque constraint manifolds. We also present numerous severe real-world test results from the integration of these algorithms into a single mobile manipulator.

Keywords Mobile manipulation · Personal robotics · Robotic manipulation · Computer vision · Search · Navigation

S.S. Srinivasa (✉) · D. Ferguson · C.J. Helfrich
Intel Research Pittsburgh, 4720 Forbes Avenue, Suite 410, Pittsburgh, PA 15213, USA
e-mail: siddhartha.srinivasa@intel.com

C.J. Helfrich
e-mail: casey.j.helfrich@intel.com

D. Berenson · A. Collet · R. Diankov · G. Gallagher · G. Hollinger · J. Kuffner · M.V. Weghe
The Robotics Institute, Carnegie Mellon University, 5000 Forbes Avenue, Pittsburgh, PA 15213, USA

D. Berenson
e-mail: dberenso@ri.cmu.edu

A. Collet
e-mail: acollet@ri.cmu.edu

R. Diankov
e-mail: rdiankov@ri.cmu.edu

G. Gallagher
e-mail: ggallagh@ri.cmu.edu

G. Hollinger
e-mail: gholling@ri.cmu.edu

J. Kuffner
e-mail: kuffner@ri.cmu.edu

M.V. Weghe
e-mail: vandeweg@ri.cmu.edu

1 Introduction

With our aging population and the rising cost of health care, the role of assistive agents in the home is becoming more and more important. There are a variety of solutions in the market, from living agents like service dogs (Canine Companions 2009) and monkeys (Monkey Helpers 2009), to robotic agents like vacuum cleaners (IRobot 2009) and teleoperated arms (Exact Dynamics 2009). However, all of these agents only provide partial solutions. Living agents can only be trained to perform a few key tasks, and suffer from being un-anthropomorphic: the home is not structured for a dog or a monkey. Autonomous robotic agents fail to address any area of the home above the floor level. A recent survey by Ray et al. (2001) reveals that when asked what they would like a robot to do for them, the top answers were cleaning, dish washing, laundry, ironing, and moving heavy things, all of which involve the robot manipulating objects in the world, and none of which achievable by robotic agents in the market now.

This is primarily because such tasks are no easy feat for robotic systems. Manipulation in human environments involves performing several challenging subtasks, including efficient navigation and mapping, robust object recognition and pose estimation, and sophisticated trajectory planning. Moreover, these all need to be performed in an environment that is unstructured and constantly changing. Each of these

Published online: 17 November 2009

subtasks is itself an active area of research, and yet all must be accomplished simultaneously to produce a system with the competence to perform even simple manipulation chores around the house, such as moving heavy objects or cleaning.

Recently, several research groups have turned their attention towards this compelling domain. Exciting recent developments have been improved user interaction with robotic systems through intuitive laser pointer-based interfaces (Nguyen et al. 2008), learning to grasp novel objects (Saxena et al. 2008), learning the structure of common articulated objects through manipulation (Katz and Brock 2008), and performing intricate tasks such as setting tables by combining high-level logic with robust perception (Muller et al. 2007).

The focus of our group has been the development of an autonomous mobile manipulation platform that can perform sophisticated manipulation tasks in human environments at human-like speeds. Our goal is to create a robotic system that can reliably perform routine tasks within the home or office and perform these tasks quickly enough that the person who requested them is not frustrated.

To this end, we have created *HERB*, the Home Exploring Robotic Butler (Fig. 1) that can efficiently map, search, and navigate through indoor environments, recognize and localize several common household objects, and perform complex manipulation tasks (such as carrying pitchers without spilling them). In this paper, we describe the key components of Herb, from its ability to search for objects, differentiate movable and immovable elements in its environment, recognize and extract the pose of common objects, and grasp and carry constrained items. We provide a number of results from component-level analysis, as well as public demonstrations during which Herb operated for several hours on end and interacted with several hundred people.

Fig. 1 Herb: a platform for Personal Robotics developed jointly by Intel Research Pittsburgh and Carnegie Mellon University

2 System architecture

HERB is the union of several onboard and offboard components. Onboard components comprise of a Segway RMP200 mobile base, a Barrett WAM arm, several sensors(described in detail in Sect. 3), and a pair of low-power computers, all of which are powered by a custom-built power supply. Onboard components communicate over a wireless network with offboard off-the-shelf PCs.

2.1 Software modules

HERB implements a set of sensing, planning, and execution modules (Fig. 2a). A high-level script arbitrates their execution and error recovery. Almost all of our current demonstrations execute the following sense, plan, and act cycle. First the robot senses its environment and sends a static snapshot to the planning system. The planning system uses the geometry and kinematics of the robot to create a global plan that avoids obstacles and meets task specific constraints. Then the execution system attempt to follow this global plan while compensating for dynamics and runtime uncertainties. Error recovery during any part of the cycle is hand-coded.

Although recent research has concentrated on tighter integration of these components, we have taken the approach of keeping the top structure simple and building more powerful primitives within each component. By using a simple three-stage framework, we can precisely define the assumptions and outputs of each stage. For example, each planning algorithm relies on a static snapshot of the environment and employs a model of the environment that can be easily simulated. Such models typically include simulation of kinematics, geometry, dynamics models, task constraints, contact constraints, and sensor visibility. Some planners use statistical models in order to simulate environment and execution uncertainty. By relying on a snapshot of the environment and using simple simulations, a planner can isolate itself from the uncertainties present in sensing and execution, thus making the planning problem more manageable. When finished, each planner should output a global plan that specifies where the robot should move; time-critical inputs like control parameters are left to the execution stage.

When designing the system architecture, we followed two driving principles:

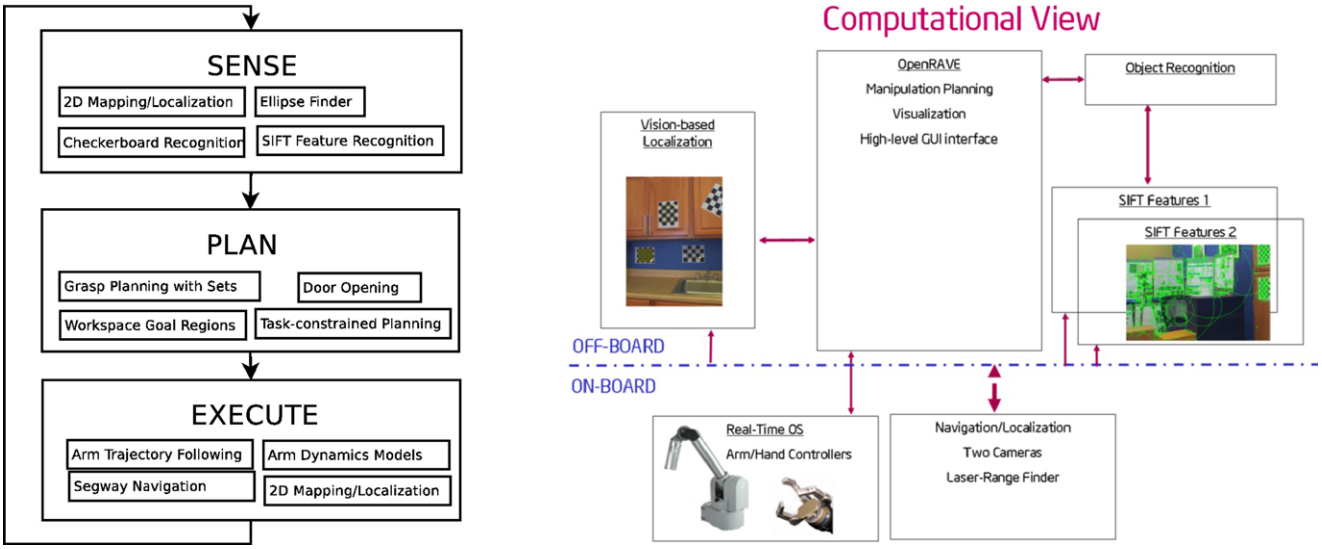

Fig. 2 (*Left*) Software architecture, listing the modules used for various demos. Crosses indicate deprecated modules. (*Right*) Distribution of computing resources. Each box represents an isolated machine

– *Unlimited computational power is available.* This allowed us to concentrate on designing more powerful algorithms with scripting languages rather than spending time on optimization and debugging. The final system involved six multi-core computers running 20–30 distributed processes communicating with each other.
– *Sensing and planning algorithms should require minimal human input*. This enables them to adapt to new environments, new conditions, or new robot systems. Employing such algorithms, as described in the later sections, helped us decrease development time because we could easily test a multitude of scenarios to find where the system succeeds and fails.

Following these principles, HERB consists of many distributed modules. For the communication infrastructure and process management, we use the *Robot Operating System* (Quigley et al. 2009) package. ROS allows us to easily transfer processes onto different computers as necessary. When deciding where and how each algorithm should be computed, we moved as much computation as possible to dedicated computers off the robot. The onboard computational power is always limited due to weight and power constraints, so it should be used for real-time tight-feedback processes only. The design space for computation is tricky because the onboard and offboard computation is separated by a wireless network and bandwidth/latency become an issue. In HERB, the execution layer lies on-board the computer because the arm movement and segway navigation require tight feedback loops greater than 10 Hz. The sensing component is divided between onboard real-time obstacle avoidance and offboard perception. The camera data is compressed and streamed offboard to construct a snapshot of the environment. The manipulation planning algorithms producing global plans are strictly offboard since each planner returns a new trajectory on the order of seconds.

2.2 Software architecture

HERB works across many computers and employs several distributed robotics packages to accomplish its tasks. The entire HERB system consists of a group of separate processes that communicate with the others through the network Fig. 2b. The lowest level components are the drivers for the laser range finder, segway, WAM arm, Barrett Hand, and the two cameras. All drivers run onboard and stream data and offer services across ROS. The navigation stack consisting of Adaptive Monte-Carlo Localization (AMCL) and Wavefront planning has exclusive control of the laser range finder and segway; its main purpose is to offer services like point-to-point movements and localization within the map.

The vision system is responsible for detecting all manipulable objects and computing a more accurate localization of the robot. In the current system implementation, we rely on 3D models and a map to predict the location of non-manipulable obstacles. Altogether the vision system spans five different computers: one onboard computer for streaming the camera data, three for recognizing objects from two cameras, and one for the localization. The main sets of features we use for object recognition are SIFT features (Lowe 2004). Every camera has a dedicated quad-core computer that computes SIFT features at 6 frames per second on a 640×480 image using the *libsiftfast* library (Zerofrog 2008). For vision-based localization, we build a map of markers using checkerboards. For detection, we use the

Fig. 3 (*Left*) Sensors on HERB and the modules that use them. (*Middle*) HERB uses automatically generated map of checkerboards for refining its localization. Entire scene is shown at the *top*, and the camera views are show at the *bottom*. (*Right*) Registering mugs with the arm camera. Image from the arm camera is inset

OpenCV checkerboard detector (Bradski and Kaehler 2008), which can run at 5 frames per second on a quad-core machine while simultaneously recognizing up to four different types of checkerboards.

Finally all sensing information is combined inside the *OpenRAVE* environment (Diankov and Kuffner 2008) to form the entire world state of the system. OpenRAVE is responsible for all the manipulation planning algorithms, providing a scripting server, and sending trajectories to the arm and hand. In order to reduce development time, we use *Octave* (Eaton 2002)/*MATLAB* to communicate with the OpenRAVE scripting server and send high-level commands. The scripting environment is responsible for managing the state of the robot, its goals, and failure conditions. It also offers a simple GUI to control the high-level execution of the robot.

3 Sensing

The following three sections describe the sensing modules on HERB, and our lessons learned from them. Sensors on HERB are added, removed, and moved based on the requirements of the task. The current set, shown in Fig. 3a, comprises of a planar laser on the Segway, a pair of cameras on the shoulder, and a camera on the arm. The SICK LMS laser on the Segway is used for localization (Sect. 4), mapping (Sect. 5) and Segway motion planning. It is placed at a fairly standard location, at ankle height. Cameras on the shoulder comprise of a Pointgray Flea above a Pointgray Dragonfly. They were placed to provide limited but accurate 1DOF panning using the shoulder. The lenses were chosen to provide a narrower field of view but greater depth of field for the Flea, and vice versa for the Dragonfly. The choice reflects their intended purpose: the Flea is used primarily for vision-based localization (Sect. 4) from variable distances, while the Dragonfly is used to recognize and register manipulable objects (Sect. 6) at a close range.

The arm camera comprises of a Logitech 9000 USB webcam in a 3D printed housing we built. We experimented with various placements of the camera: from the forearm to the palm. Placing the camera further along the kinematic chain provided greater dexterity at the cost of accumulating joint uncertainty, USB cable management, and occlusion by grasped objects and the arm. Considering all of these factors, we found the placement in Fig. 3c to be the most suitable for providing a vantage point suitable for running our now-deprecated mug detection algorithm (Sect. 6).

4 Segway localization

Localizing HERB's base involves registering its pose (x, y, θ) in the coordinate frame of the environment. Our environment comprises of two collocated maps: a planar occupancy grid generated by the laser, and 3D models of regions HERB manipulates in. Presently, the former is much larger than the latter: HERB wanders around the entire lab (about 3000 sq.ft.) but only manipulates in the kitchen (about 100 sq.ft.). Further details of our planar map representation and its treatment of dynamic obstacles are presented in Sect. 5.

We use two localization modes: a coarse mode when HERB is navigating which uses adaptive Monte Carlo localization a standard package available in Player/Stage, and a fine mode in the kitchen when HERB needs to manipulate which uses a novel vision-based localization scheme we have developed. These modes reflect our experimentally derived degrees of precision to guarantee successful navigation and manipulation in clutter.

4.1 Checkerboard localization

For an accurate robot localization system, we chose to use cameras localizing with respect to checkerboards scattered across the environment. Although we could localize with respect to the laser map used for navigation, the vision system gave errors within 5 millimeters, so was much more accurate that the adaptive Monte Carlo Localization algorithm used for navigation. To create the map of all checkerboards, we took about 20 snapshots of the kitchen environment and stitched all the observations together to form a coherent map. The stitching process involved looking at all possible permutations of checkerboards and discarding all inconsistent measurements. If a unique map could not be found with the current observations, it usually meant there was not enough constraints for where the individual checkerboards are with respect to each other; in this case, we added more images until a unique map was generated. The final maps we use for demos usually consist of 20+ checkerboards. Figure 3b shows the robot localizing using a map of 7 checkerboards along with the simulated camera and real camera views.

4.2 Lessons learned

Checkerboard localization was far more accurate than AMCL, but took often 10–30 seconds to provide an estimate. This delay was largely due to instabilities in checkerboard detection especially at a distance: the algorithm wanted until checkerboard pose stabilized before starting to process it. Furthermore, to provide an accurate estimate, the algorithm needed to see at least three checkerboards in a single image. To be able to guarantee this everywhere in the kitchen, we had to place numerous checkerboards in the scene. Needless to say, this make the kitchen look rather unnatural. We attempted to mitigate this by replacing the checkerboards with children's drawings and detecting them with our object recognition algorithm from Sect. 6 but the drawings did not have enough contrast to be visible from 3–4 m.

We managed to improve the accuracy of AMCL by adding a small perturbation to the pose, even when the robot was stationary, forcing AMCL to update its estimate. This often produced better results, but sometimes produced incorrect but extremely confident results. We eventually decided that failing occasionally to grasp an object or open a door was preferable to pasting the kitchen with checkerboards, and waiting for numerous seconds for a better estimate.

5 Navigation and mapping

Maps of indoor environments change constantly due to the rearrangement of movable objects. While online re-mapping using a SLAM algorithm might provide a solution for navigation, we envisioned an algorithm for handling changes in the environment, while maintaining semantic labels. To accomplish this, Herb uses GATMO, a Generalized Approach to Tracking Movable Objects, to maintain hypotheses of where people and other movable objects are in its environment (Gallagher et al. 2009).

5.1 Previous work

GATMO builds on several active areas of interest, including people tracking, dynamic object detection and recognition, and dynamic mapping. There has been significant work done with people tracking (Montemerlo and Thrun 2002), and Detecting and Tracking Moving Objects (DATMO) (Mendes et al. 2004; Wang and Thorpe 2002). However, these approaches either use a static map, or seek to remove moving objects from the map (Schulz et al. 2003; Hahnel et al. 2003). Conversely, recent mapping strategies (Schulz and Burgard 2001) have been able to detect when an object has changed position, but use static maps and run offline. GATMO merges work in these areas to allow dynamic object detection and mapping to be done simultaneously in an informed and online manner.

5.2 Map representation

HERB's map consists of two parts: a static (original) map, and several lists of map objects with locations and orientations on the map. There are four lists that collectively encompass all map objects. The list maintains the position and orientation of each object, as well as the history of the objects sightings. These lists are given in Table 1.

When GATMO is first given a new map, it segments the map based on connectivity to other objects, and adds all the objects to list U. In addition, objects in U are compared to each other to determine if they are the same, for example, a set of identical chairs. Over time, as objects move, the status of objects change to movable or absent, depending on whether the objects are seen in their new positions. If an object is considered movable, then immediately all matching objects are considered movable. In this way, GATMO allows the robot to make intelligent predictions about the environment, even in situations with partial observability.

Table 1 Lists maintained in a GATMO map

List Name	Object Description	Comment
A	Absent	Was in the map, but is now absent
B	Movable	Has moved, currently in the map
U	Unclassified	Default object state
Γ	Never Added	Observed, but never added

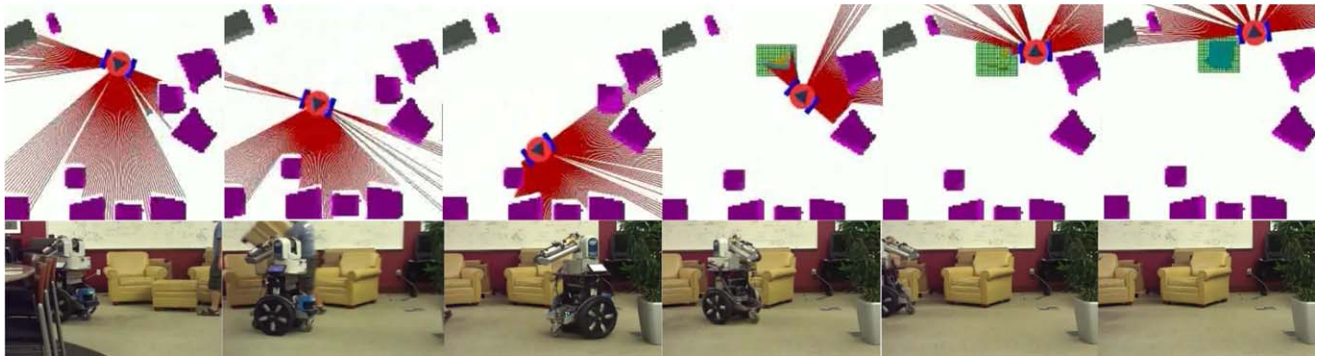

Fig. 4 (Color online) Recognizing object movement: This figure show the process of recognizing the absence of an object and re-assigning it to a new location. The robot is shown by the *circle with a triangle on it*. The *red lines* extending from the robot show the ray to the laser scan observations at that time step. The *purple blobs* are objects in the environment, and the *grey blob* is the wall, colored differently for visualization purposes

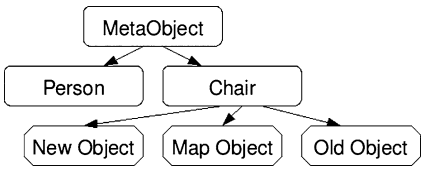

Fig. 5 Multiple hypothesis hierarchy of object classification

5.3 Object classification

In GATMO, observations are clustered and classified as observed objects in a multi-level hypothesis hierarchy. At the root of this hierarchy, the observed object is represented by a *Metaobject*. *Metaobjects* can have two hypotheses, *person* or *chair*. (See Fig. 5.) The *person* hypothesis represents objects that actively move, like people, animals, and other robots. The *person* hypothesis maintains a Kalman filter that tracks the position and velocity of the object's center. The *chair* hypothesis represents objects that seldom move. It is represented by a grid structure, which encodes the past observations similar to an occupancy grid. The *chair* hypothesis also maintains several association hypotheses. In addition to considering itself as a new object on the map, the *chair* hypothesis considers the probability that it is an object in map lists A, B or Γ. If a *chair* hypothesis predicts a match with an object in one of these lists with sufficiently high probability, it is added to the map in list B.

An example of GATMO identifying a movable object is presented in Fig. 4. In this figure, HERB sees a new object and maintains a chair hypothesis (green grid in image one through five). At the same time, HERB notices that another object is missing from the map (light purple in images two, three and four). When it decides that the chair is no longer present, it is removed from the map (image five). Finally, the chair hypothesis is matched to the absent object, and the map is updated (image 6).

5.4 Lessons learned

The interaction between GATMO and localization is mostly mutually beneficial. GATMO provides the localizer its best current map of a constantly changing world. The localizer uses this map to provide GATMO with its best current estimate of HERB's pose. Sometimes, especially when the localizer is confidently wrong, the interaction can lead to static regions being eaten away due to poor localization. This effect disappears when good localization is regained, and the mistakenly deleted regions grow back when sensed again, but it is certainly possible for a determined adversary to foil GATMO. We are exploring two approaches to mitigate this effect: decoupling localization from the planar map by using vision-based localization with ceiling markers for example, and adding other sensors, like cameras, to assist in GATMO's object classification.

6 Vision

Robust perception is a vital capability for robotic manipulation in unstructured scenes. In this section, we present an approach for real-time object recognition and full pose estimation from natural features, as shown in Collet et al. (2009). For each object, a metric 3D model is built using local descriptors from several images. Then, for every new test image, we use a novel combination of RANSAC and Mean Shift clustering to recognize and register all instances of each object in the scene. The resulting system provides markerless 6-DOF pose estimation for complex objects in cluttered scenes.

Requirements for the vision system were motivated by HERB's environment and its abilities. Our scenes were cluttered, often with multiple instances of the same object (Fig. 6). Since localization was expensive, we required pose

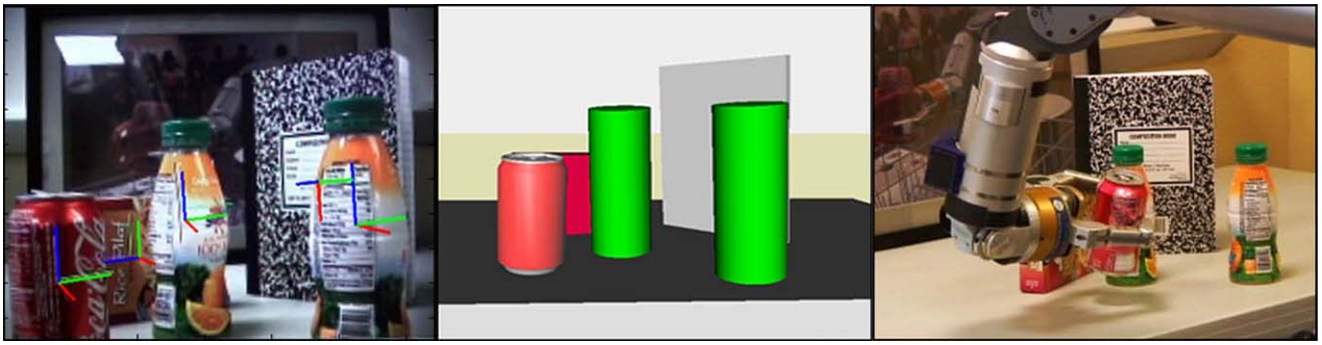

Fig. 6 Object grasping in a cluttered scene through pose estimation performed with a single image. (*Left*) Scene observed by the robot's camera, used for object recognition/pose estimation. Coordinate frames show the pose of each object. (*Middle*) Virtual environment reconstructed after running pose estimation algorithm. Each object is represented using a simple geometry. (*Right*) Our robot platform in the process of grasping an object, using only the pose information from this algorithm

estimates from a single image. We estimated an accuracy requirement of 1 cm at a distance of 1 m for reliable grasp success in high clutter. Finally, we set a time limit of 1 second for the entire pipeline, from image acquisition to pose estimation.

Our pipeline consists of image extraction, onboard compression, wireless transfer, decompression, and pose extraction. Pose estimation accuracy, grasp success, and timing numbers are detailed in the results.

6.1 Related work

Reliable object recognition, pose estimation and tracking are critical tasks in robotic manipulation (Taylor and Kleeman 2003; Ekvall et al. 2005; Zickler and Veloso 2006; Mittrapiyanuruk et al. 2004) and augmented reality applications (Vacchetti et al. 2004; Gordon and Lowe 2006). In robotics research, many approaches try to solve a simplified pose estimation problem, in which the object is usually assumed to be lying on a planar surface, hence restricting the search to a position $[x\ y]$ plus an angular orientation (Walter and Arnrich 2000; Zhang et al. 1999). Several methods are available to estimate the 6-DOF pose of objects: Ekvall et al. (2005), use color co-occurrence histograms and geometric modeling; Mittrapiyanuruk et al. (2004) use Active Appearance Models (AAMs) for the registration process, among others.

6.2 Ellipse finder

Our object recognition algorithm was motivated by the failures of a previous algorithm we had developed to detect the circular tops of mugs. The ellipse finder made numerous simplifying assumptions: that all mugs were of a known height, uniform color, standing upright on a uniform background. Under these assumptions, the tops of mugs appeared as ellipses in an image which we extracted using a fast symmetry detector (Loy and Zelinsky 2003). Orientation of the handle was detected by searching an annulus around the detected ellipse (Fig. 3c).

Our goal was to go beyond this: to detect textured freeformed objects in clutter against arbitrary backgrounds.

6.3 Modelling objects using natural features

The first task towards the creation of our automated recognition and registration system is the training stage. Our system uses natural features of the object to create a 3D metric model. Reliable local descriptors are extracted from natural features (i.e. features present in an object, not artificially added) using SIFT (Lowe 2004). Matching between SIFT descriptors is performed using the Best Bin First (Beis and Lowe 1997) algorithm. Using structure from motion on the matched SIFT keypoints, we merge the information from each training image into a sparse 3D model. Finally, proper alignment and scale for each model are optimized to match the real object dimensions. Some examples are shown in Fig. 7.

6.4 Automatic object recognition and pose estimation

The on-line stage of this system is a fully automated object recognition and pose estimation algorithm from a single image. Using the information from each sparse 3D model, this algorithm is able to detect several objects and several instances of each object by combining Levenberg-Marquardt optimization with clustering and robust matching. The output information is the object types and their transformations R_{est}, t_{est} with respect to the camera frame. If the camera has been extrinsically calibrated, all objects can be accurately positioned in any virtual environment we wish to use. Each object type is processed independently in each image when using this algorithm. For each object type, the recognition/pose estimation algorithm is executed as follows.

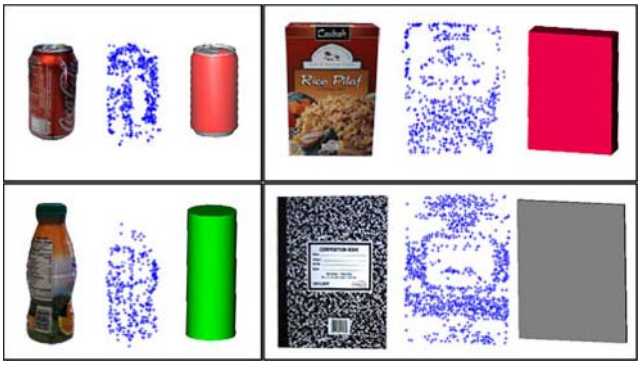

Fig. 7 Learned visual models for soda can, rice box, juice bottle and notebook. Each box contains an image of the object, a projection of its corresponding sparse 3D model, and its virtual representation in HERB's workspace

1. Cluster SIFT features' 2D locations p using Mean Shift algorithm. Each cluster contains a subset of points p^k.
2. For each cluster of points p^k, choose a subset of n points and estimate a hypothesis with the best pose according to those points. If the amount of points consistent with the hypothesis is higher than a threshold ϵ, create a new object instance and refine the estimated pose using all consistent points in the optimization. Keep repeating this procedure until the amount of unallocated points is lower than a threshold, or the maximum number of iterations has been exceeded.
3. Merge all instances from different clusters whose estimated pose is similar. Those instances with the most consistent points survive.

6.5 Results

In order to prove our pose estimation algorithm's suitability for robotic manipulation, two sets of experiments have been conducted, both using the 4 objects depicted in Fig. 7. The first set evaluates our algorithm's accuracy in estimating the position and orientation of objects in images. To evaluate the position accuracy of our algorithm, each object is placed alone in an image and moved in 10 cm intervals, from 30 to 90 cm away from the camera and from 0 to 20 cm laterally. To evaluate the rotation accuracy, in-plane (i.e. parallel to the image plane) and out-of-plane rotations are examined independently. Each object is placed alone in an image at 50 cm away from the camera and rotated from -45 to 45 degrees, in 15-degree intervals along the desired test axis. Given the randomized nature of our implementation, 10 images are taken at each test position and their results averaged. The average translation error is 0.67 cm, the average in-plane rotation error is 1.23 deg and the average out-of-plane rotation error is 3.81 deg.

The second set uses the full pose estimation algorithm in the context of our robotic system to grasp objects in cluttered scenes. 25 grasping attempts were executed for each object. The grasping tests were performed by placing a single object on a table within the robot's reachable space. Prior to each grasping attempt, the object is placed in a new arbitrary position and orientation (standing up, sideways and upside down). A grasp is considered successful if HERB grasps the object and is able to lift it 5 cm off the table. The grasping success rate in this test is 91%, thus confirming the statement that our pose estimation algorithm is accurate enough to enable robotic manipulation of the detected objects. The system runs on average at 4 fps, although it depends heavily on the analyzed scenes (from 1 fps with 12 objects to 6 fps with one or two objects).

6.6 Lessons learned

Computer vision literature is filled with local descriptors and object recognition algorithms. Laying out a set of requirements early in the process—automatically estimating the pose of objects in clutter accurately—enabled us to quickly focus on a select few and, to our surprise, discover inadequacies in many current approaches, and provide a robust solution.

Our algorithm performs extremely well on textured objects, which are rich in SIFT features. Home environments, however, contain numerous UN-textured objects. We believe that the fusion of local geometric descriptors, as well as other sensing modalities, like stereo and laser data, will be required to detect and register these objects accurately.

7 Planning

The following two sections describe the planning modules on HERB and our lessons learned from them. Using our planners, we are able to robustly execute common household tasks, like manipulating doors, cabinets and handles, grasping free-form objects in high clutter, and manipulating objects with task and weight constraints.

The planners take as input the description of a robot and parts of the environment we do not want the robot to collide with. The type of description is fairly flexible: we have used triangular meshes, shape primitives, and voxel grids. The description of the robot usually includes its kinematics and joint and torque limits. They also take as input the kinematics of the environment. These include the presence and description of constraints like hinges, as well as the absence of constraints like between an object and the tabletop. Most object IDs are populated automatically by the vision system. Some object IDs, like those of doors are populated manually.

The description of goals for the planners are general, for example:

– "Open the fridge door > 90 degrees"

– "Clear all objects from table"
– "Retrieve juice from fridge without tipping"

The planners produce as output kinematically feasible paths for the robot. Note that the robot description could include both the arm and the base for a mobile manipulator. Failure is returned if a path is not found within a time limit. Kinematically feasible plans are retimed to satisfy velocity and acceleration constraints, for execution on the robot.

Our planners share the following features:

– *Generality*: If a robot and an environment fit our input description, the planners will produce an output.
– *No physics*: Our planners are purely kinematic. We do not use any models of dynamics or friction. Our experiments with physics simulation for manipulation were unsatisfactory: we found them both computationally expensive and inaccurate.
– *Uncertainty*: We address kinematic uncertainty by ensuring our plans work in spite of pose error. This is accomplished in practice by jittering the plan and ensuring it is still kinematically feasible.

While these features have enabled us to produce plans in a few seconds, they are by no means set in stone, merely a list of what has worked for us. It is easy to find situations where our planners are not applicable, like underactuated joints, uncertain environments, dynamic manipulation or pushing, to name a few.

8 Opening doors by planning with caging grasps

An autonomous home robot needs to open and close doors, drawers, cabinets, and turn handles with human-level performance and speed. Because motions for these objects are constrained to one or two degrees of freedom, the free configuration space of robot motions manipulating these objects is greatly reduced. This coupled with the fact that small execution errors in the robot can produce large counter forces from the object means that an autonomous robot has to be able to reason using compliance strategies with prehensile grasps (Prats et al. 2008), strategies with non-prehensile grasps like pushing or hooking (Pereira et al. 2002), and visual servoing strategies combined with force control (Jain and Kemp 2008).

Herb is designed to open and close doors, drawers, cabinets, and turn handles using a formulation of planning using caging grasps (Diankov et al. 2008) that relaxes the task constraints imposed on the robot (Fig. 8). An advantage to considering caging grasps is that the directions of compliance are automatically computed using simple contact analysis, which results in an automated process to build a model of how a robot can manipulate doors. In contrast to compliance control research, this method of planning does not require any extra human knowledge in specifying which degrees of freedom are constrained and which the robot can tolerate execution errors in. Furthermore, relaxing task constraints through caging grasps can enable low degree-of-freedom robots to achieve a constrained task that could not be possible with previous formulations.

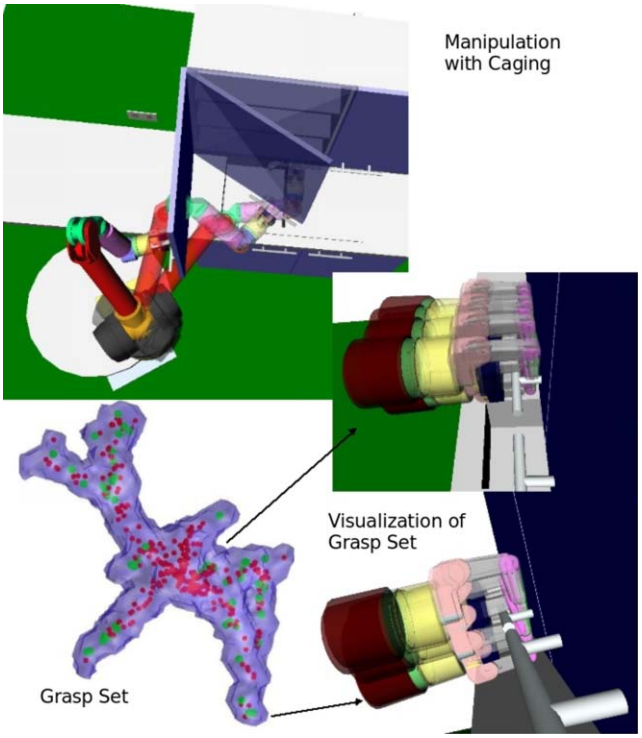

Fig. 8 (Color online) A robot hand opening a cupboard by caging the handle. Space of all possible caging grasps (*blue*) is sampled (*red*) along with the contact grasps (*green*)

8.1 Caging grasp formulation

We formulate the problem using the configuration space of the robot $q \in \mathcal{Q}$, the configuration space of the end-effector $g \in \mathcal{G}$, and the configuration of the constrained target object $\rho \in \mathcal{R}$ (Fig. 9). Each of these spaces is endowed with its corresponding distance metric $d : \mathcal{X} \times \mathcal{X} \rightarrow \mathbb{R}$. In the relaxed task constraint formulation, each target object is endowed with a task frame which is rigidly attached to it, and a set of grasps G represented in that task frame.

A transform T_ρ relates the task frame at an object configuration ρ to the world reference frame. Because all the grasps in G are in the task frame, it allows us to compute and cache G offline, thereby improving the efficiency of the online search. At any configuration ρ, we denote the grasp set in the world frame by

$$T_\rho G = \{T_\rho g \mid g \in G\}. \quad (1)$$

We assume while planning that the end-effector of any configuration of the robot always lies within the grasp set

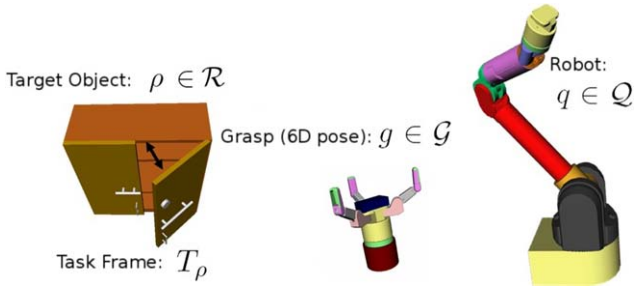

Fig. 9 The configuration spaces used for planning. The task frame T_ρ and the caging grasps $\mathcal{G}$ are used to combine the target space $\mathcal{R}$ and robot spaces $\mathcal{Q}$

G with respect to the task frame. Because this couples the motion of both the object and the robot during manipulation, their configurations need to be considered simultaneously when planning. We define the relaxed configuration space $\mathcal{C}$ as

$$\mathcal{C} = \{(\rho, q) \mid \rho \in \mathcal{R}, q \in \mathcal{Q}, FK(q) \in T_\rho G\}. \quad (2)$$

The free configuration space $\mathcal{C}_{\text{free}} \subseteq \mathcal{C}$ contains all states not in collision with the environment, the robot, or the object. In order to plan in this high configuration space, we use Randomized A* (Diankov and Kuffner 2007), which operates in a similar fashion to A* except that it generates a random set of actions from each state visited instead of using a fixed set. Randomized A* is well suited to our current problem because

- The distance of the target configuration to its goal can be reflected in the A* heuristic.
- There is a guarantee that each state is visited at most once.
- Does not need to generate all the IK solutions for a given grasp.
- Can return failure after exhausting the entire search space.

8.2 Results

To show the improved caging grasps have in the feasibility of task, we randomly sampled positions for the robot and computed the success rate for each position. For example, for a 6 degree-of freedom arm, the feasibility regions increase by %500 (Fig. 10). Furthermore, because the algorithm is extendable to different grasping modalities like pushing, Herb can open a refrigerator by first hooking it and then pushing it (Fig. 11).

8.3 Lessons learned

Opening doors and cabinets was, by far, hardest task for HERB. Admittedly, this is a qualitative statement, but the visual impact of HERB finding creative ways both to succeed and to fail to open doors was tremendous. Perhaps it was the realization that HERB could no longer treat the world like a game of pickup sticks, carefully avoiding everything but the object it had to grasp.

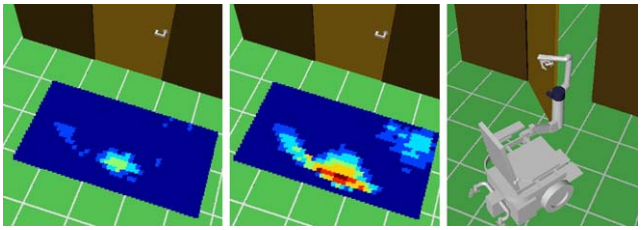

Fig. 10 Comparison of fixed feasibility regions (*left*) and relaxed feasibility regions (*right*) for each scene

Fig. 11 Herb autonomously opening the refrigerator door by first pulling on the handle and then pushing the door from the inside

Door opening also challenged our disregard for physics: HERB's fat fingers would often jam in the handle trig erring a stall in the arm controller. We believe that an intelligent combination of the position freedoms afforded by caging and force control is required for robust execution.

9 Manipulation planning

Our approach to the manipulation planning problem for HERB centers around exploiting the freedom allowed by loose task specifications and, for objects that require it, constraining the robot's motion so that constraints on the object are met.

9.1 Planning with workspace goal regions

HERB is designed to perform manipulation tasks in the home. Many such manipulation tasks afford a large amount of freedom in the choice of grasps, arm configurations, and object goal locations. For example, when we pick up a coffee mug and place it in the sink, we can choose from a wide range of hand configurations to grasp the mug securely, as well as a wide range of goal locations in the sink to place the mug. However, a manipulation planning algorithm that exploits this freedom in the task specification must also maintain efficiency and desirable completeness properties.

To handle loose task specifications we introduced the concept of Workspace Goal Regions (WGRs) (Berenson

Fig. 12 Snapshots from HERB executing a trajectory for lifting a pitcher (*left* to *right*). The pitcher is constrained to not tip forward or to the side. This constrained trajectory took less than 5 seconds to plan using the CBiRRT planner. After the trajectory is complete, the robot turns and waits for the user to pull on the pitcher before releasing it

et al. 2009a), which allows us to specify continuous regions in the six-dimensional workspace of the robot's end effector as goals for a planner. A given task can entail any number of WGRs, each of which encompass a subspace of any dimension less than or equal to six. These volumes are particularly useful for manipulation tasks such as reaching to grasp an object or placing an object onto some 2D surface or into some 3D volume. WGRs are also intuitive to specify, can be efficiently sampled, and the distance to a WGR can be evaluated very quickly. WGRs can be integrated into most sampling-based planners by trading off exploitation of the WGRs and exploration of the configuration space. In the case of an RRT, we trade off sampling from the WGRs to get goal configurations and growing the search tree. WGRs enable our RRT-based planner to quickly reach and grasp simple objects such as cans, bottles, and boxes for which we can easily define regions of acceptable grasping poses. To grasp complex objects we use the methods described in Berenson et al. (2007) and Berenson and Srinivasa (2008). Once an object is grasped, its goal pose can easily be defined as a WGR for many tasks such as throwing the object away or handing it to a person. Several examples of HERB executing trajectories planned by using WGRs are shown in Fig. 13.

Fig. 13 Snapshots from three runs of the planner on the WAM arm. *Top Row*: Grasping and throwing a way a box of rice. *Middle Row*: Grasping and throwing away a juice bottle. *Bottom Row*: Grasping and throwing away a soda can

9.2 Planning with constraints

Though the start and goal locations of objects afford a large degree of freedom in robot configuration, many objects cannot be moved arbitrarily. For instance, if a mug is full of coffee, we should not tilt the mug while moving it. Such constraints on object pose limit the allowable configurations of the robot while moving and thus complicate the path-planning problem.

Creating manipulation planning algorithms to perform constrained tasks also involves computing motions that are subject to multiple simultaneous task constraints. For example, a robotic manipulator lifting a heavy milk jug while keeping it upright involves a constraint on the pose of the jug as well as constraints on the arm configuration due to the weight of the jug. In general, a robot cannot assume arbitrary joint configurations when performing constrained motions. Instead, the robot must move within some manifold embedded in its configuration space that satisfies both the constraints of the task and the limits of the mechanism. To create plans for such constrained tasks, we have developed the Constrained BiDirectional Rapidly-exploring Random Tree (CBiRRT) (Berenson et al. 2009b) motion planning algorithm, which uses Jacobian-based projection methods as well as efficient constraint-checking to explore constraint manifolds in the robots configuration space. The CBiRRT can solve many problems that standard sampling-based planners, such as RRT or Probabilistic Roadmaps (PRM), cannot. Our framework for handling constraints allows us to plan for manipulation tasks such as sliding and lifting heavy objects, and maintaining pose constraints while moving objects (see Fig. 12).

9.3 Lessons learned

Fast, feasible manipulation planning in high clutter is HERB's strength. HERB is extremely good at snaking into a cluttered environment and retrieving the desired object. There are, however, numerous possible improvements: producing plans that provide worst-case guarantees under pose uncertainty, incorporating simple physics like planar pushing during manipulation, and legible, repeatable arm motion.

10 Demonstrations

In addition to testing the accuracy of each of our system components, we are strong advocates of running live, public demonstrations. These demos involve real objects, user interaction, and often uncontrolled lighting and dynamic obstacles. As our system capabilities have improved, we have also increase the complexity of our demonstration tasks. The following sections describe four of our demos, their goals, software modules used, and our lessons learned.

10.1 R@I: Moving coffee mugs

In June 2008, our robot performed a day-long public demonstration of recognizing and registering coffee mugs and loading them into a dishwasher rack at the Research at Intel Day event (Fig. 15a). The arm was mounted on a fixed waist-high platform, and flanked by three circular white pedestals. The audience placed black plastic mugs upright in random locations on either of the two side pedestals. HERB searched for a mug on either pedestal and placed it in a dishwasher rack located on the central pedestal.

The software modules used were the ellipse finder, grasp planning with sets, and the arm and hand controllers. The Segway was not used for this demonstration due to space limitations on our booth imposed by the demo staff.

Error recovery comprised of grasp failure detection if the mug was pulled away before the hand closed and compliance if the arm collided with an unmodeled obstacle. In both these cases, the arm returned to its prespecified home position and retried. HERB also requested for the dish rack to be emptied if it was no longer able to place a mug in it.

During the course of the day the system loaded over 400 mugs into the rack. Multiple mugs were placed by visitors in whatever upright position and rotation they chose, on either pedestal. Planning and execution for locating, grasping and loading a single mug in a cluttered scene took 15 seconds, on average. Only 6 times during the day did the robot fail to successfully load the intended mug. All of the failures occurred during grasping when the mug slipped out of the robot's hands. We believe that this is due to errors in extrinsic calibration of the camera due to inaccuracies in the robot's cable-driven joints which, albeit small, can produce large ± 3 cm. errors in estimating the pose of a mug located about 1 m. away from the camera.

10.2 IDF: Collecting coffee mugs

In August 2008, we demonstrated HERB during a brief on-stage performance at the Keynote of Justin Rattner, CTO Intel, at the Intel Developer's Forum, during which we instructed the robot to autonomously drive out on-stage, use an arm-mounted webcam to locate a pair of plastic coffee mugs placed on a pedestal, pick them up and load them one at a time into an onboard storage bin, and retreat offstage (Fig. 15b).

We added Segway localization and Wavefront navigation to the software modules. A speech synthesis module was added to interact with users. Additional error recovery in the form of a joystick override for the Segway was mandated by the demo staff.

The only shortcoming during execution was an inability for the Segway to reach its goal in front of the pedestal with enough precision for the vision and arm systems to work, thus requiring a brief human intervention to joystick the Segway a little closer to the pedestal. The entire sequence of driving out, interacting, recognizing, picking up and storing two mugs, and driving back took approximately 3 minutes. The demonstration was at the Moscone Center in San Francisco, with over 5000 people in the audience.

10.3 Search: combining search and action

In August 2008, we examined the task of searching for an object and then performing some action with that object. This is of particular interest because it lays the groundwork for worthwhile tasks around the home and office (e.g., fetching coffee, washing dirty dishes, etc.). A complete description of this work, with detailed theoretical and implementation results, is available in Hollinger et al. (2009).

Figure 14 shows images of Herb searching for a coffee mug and returning it to the sink. In this demo, Herb utilizes the Adaptive Monte Carlo Localization system with a laser scanner to localize itself while it moves between possible mug locations. Upon reaching a possible mug location, it uses the vision system to identify the mug and the grasp tables to pick it up.

We are interested in extending our search/action framework for ordering and solving multiple queries. For instance, a robot assistant may need to divide its time across a range of tasks. This would require constant management of search and action. Solving this problem would enable Herb to provide continuous assistance for daily living.

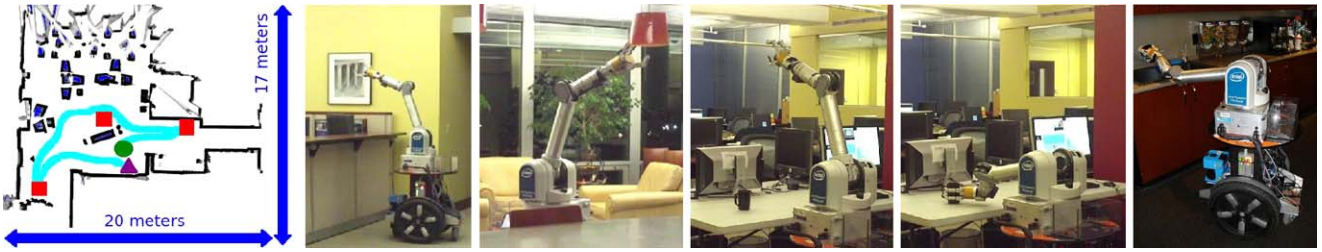

Fig. 14 (Color online) Map of the kitchen environment used for fetch tasks (*left*). Herb starts at the *green circle* and generates a path to look for a coffee mug at the locations marked by *red squares*. After finding the mug, Herb takes the mug back to the kitchen sink (*purple triangle*). Snapshots of Herb looking for the mug and performing the fetch task are also shown from *left* to *right*

(a) Research at Intel Day (b) Intel Developer Forum Keynote (c) Intel Pittsburgh Open House

Fig. 15 Public demonstrations of Herb

10.4 IOH: retrieval in a kitchen

In October 2008 we demonstrated HERB performing simple retrieval tasks in its own domestic kitchen at the Intel Research Pittsburgh Open House (Fig. 15c). During the day-long event visitors used a GUI to instruct HERB to autonomously move around the kitchen, open a cabinet or refrigerator door, pick up user-selected objects, and either bring the objects to the user or place them in a nearby recycling bin. The objects were previously known to the robot, and included a water pitcher, a juice bottle, a box of rice, and a prepackaged soup bowl. When handing the requested object to the user, the robot waited until it sensed an externally-applied force on the arm joints before releasing its grip, creating an intuitive robot-human handoff.

This demonstration showcased all of our latest modules: object recognition, caging grasps, task-constrained planning, and workspace goal regions.

The grasping portion of the kitchen task differed little from previous work, and was the most reliable part of the system. The system struggled with opening both the cabinet and refrigerator doors, frequently failing due to contention between the calculated arm trajectory and the natural trajectory of the door handle. The navigation module managed to move the Segway with very few collisions, but often stopped a little short of the ultimate goal. And while the object recognition and localization system was very successful at correctly identifying objects, it occasionally made small errors in estimating the pose of objects in high clutter, resulting in the arm bumping into surrounding objects.

On a typical run, the robot would spend 25 seconds driving from point to point in the kitchen, 15–30 seconds to relocalize itself, 45 seconds to open a door, and 30 seconds to pick up an object.

10.5 Lessons learned

Although we are often surprised at the new ways the system can fail to do what we expect, most of the failures can be traced to one of a few root causes, including:

1. *Inaccuracy in the robot joints*: The cable-driven WAM arm has the benefit of being naturally compliant and proprioceptive. Unfortunately, the price paid is accuracy: the joints are limited to an accuracy of ± 0.5 degrees. This error effects not only the positioning of the hand during grasping, but also any of our object localization algorithms which rely on robot-mounted cameras.
2. *Robot localization errors*: Since the camera-based localization used to calculate the robot's pose is subject to error, the arm will occasionally bump into obstacles which the planner believed were further away.
3. *Segway positioning accuracy*: Because our Segway motion planner requires a generous goal region, the Segway occasionally ends up in a position that leaves the arm trajectory planner unable to find a solution for reaching the desired object.

Fig. 16 A typical unexpected collision

11 Conclusions and future work

Building an end-to-end system of Herb's complexity has been a great learning experience. We have learned the most from repeated and severe testing, not just of each component in isolation, but all of the components together. In spite of building robust algorithms, errors do occur. In our experience with a live audience, we have observed that detecting that an error has occurred, and not repeating it is often as important as error recovery: the worst errors are those where the robot hits an obstacle, backs up, and then hits it again.

Making sense of unexpected collisions, which constitute a majority of our errors as Herb operates in tight cluttered spaces, is surprisingly hard. A typical such case is illustrated in Fig. 16. Here, the arm controller has just returned a stall because of an unexpected collision with the real cupboard handle. The planner, however, thinks there is no collision. How does it plan its next move? The cause of the collision could be poor localization, poor arm calibration, or an unmodeled or poorly modeled obstacle. We believe that collision recovery is a relevant and exciting area for future work.

We are also interested in building environment and object models online, something that we do not do with Herb. To that end, we are outfitting Herb with a spinning Hokuyo TOP-URG laser on the camera mast on the shoulder. We are excited about the prospects of entering a completely unmodeled kitchen and performing useful tasks in it.

Our manipulation planning algorithms are designed to be general, as described earlier. We believe that they could as well work on El-E (Nguyen et al. 2008), PRII (PRI 2008), STAIR (Saxena et al. 2008), or any other mobile manipulator. In the future, we are looking forward to sharing our algorithms with other groups to help build a Personal Robotics ecosystem.

Acknowledgements This material is based upon work partially supported by the National Science Foundation under Grant No. EEC-0540865. Berenson, Collet, Diankov, Gallagher, and Hollinger were partially supported by the Intel Summer Fellowship 2008 awarded by Intel Research Pittsburgh. Vande Weghe is supported by Intel Research Pittsburgh. Collet is partially supported by the La Caixa Fellowship. Gallagher is partially supported by a National Science Foundation Graduate Research Fellowship.

References

Beis, J. S., & Lowe, D. G. (1997). Shape indexing using approximate nearest-neighbour search in high-dimensional spaces. In *Proc. IEEE conf. comp. vision patt. recog.* (pp. 1000–1006).

Berenson, D., & Srinivasa, S. (2008). Grasp synthesis in cluttered environments for dexterous hands. In *IEEE-RAS international conference on humanoid robots*.

Berenson, D., Diankov, R., Nishiwaki, K., Kagami, S., & Kuffner, J. (2007). Grasp planning in complex scenes. In *IEEE-RAS international conference on humanoid robots (Humanoids07)*.

Berenson, D., Srinivasa, S., Ferguson, D., Collet, A., & Kuffner, J. (2009a). Manipulation planning with workspace goal regions. In *IEEE international conference on robotics and automation*.

Berenson, D., Srinivasa, S., Ferguson, D., & Kuffner, J. (2009b). Manipulation planning on constraint manifolds. In *IEEE international conference on robotics and automation*.

Bradski, G., & Kaehler, A. (2008). *Learning OpenCV: computer vision with the OpenCV library*. New York: O'Reilly.

Canine Companions (2009). Canine companions for independence. http://www.cci.org.

Collet, A., Berenson, D., Srinivasa, S. S., & Ferguson, D. (2009). Object recognition and full pose registration from a single image for robotic manipulation. In *IEEE international conference on robotics and automation*.

Diankov, R., & Kuffner, J. (2007). Randomized statistical path planning. In *IEEE/RSJ international conference on intelligent robots and systems*.

Diankov, R., & Kuffner, J. (2008). *Openrave: A planning architecture for autonomous robotics*. Tech. Rep. CMU-RI-TR-08-34, Robotics Institute, Carnegie Mellon University.

Diankov, R., Srinivasa, S., Ferguson, D., & Kuffner, J. (2008). Manipulation planning with caging grasps. In *IEEE-RAS international conference on humanoid robots*.

Eaton, J. W. (2002). *GNU Octave Manual*. Network Theory Limited.

Ekvall, S., Kragic, D., & Hoffmann, F. (2005). Object recognition and pose estimation using color cooccurrence histograms and geometric modeling. *Image Vision Comput*, *23*(11), 943–955.

Exact Dynamics (2009). Arm: Assistive robotic manipulator. http://www.exactdynamics.nl.

Gallagher, G., Srinivasa, S., Bagnell, D., & Ferguson, D. (2009). An online approach to object detection, modeling and mapping for mobile robots. In *Proc. int'l conf. robotics and automation*.

Gordon, I., & Lowe, D. G. (2006). What and where: 3d object recognition with accurate pose. In *Toward category-level object recognition* (pp. 67–82).

Hahnel, D., Triebel, R., Burgard, W., & Thrun, S. (2003). Map building with mobile robots in dynamic environments. In *Proc. int'l conf. robotics and automation*.

Hollinger, G., Ferguson, D., Srinivasa, S., & Singh, S. (2009). Combining search and action for mobile robots. In *Proc. int'l conf. robotics and automation*.

IRobot (2009). Roomba. http://www.irobot.com.

Jain, A., & Kemp, C. C. (2008). Behaviors for robust door opening and doorway traversal with a force-sensing mobile manipulator. In *Proceedings of the manipulation workshop in robotics science and systems*.

Katz, D., & Brock, O. (2008). Manipulating articulated objects with interactive perception. In *IEEE international conference on robotics and automation*.

Lowe, D. G. (2004). Distinctive image features from scale-invariant keypoints. *International Journal of Computer Vision*, *60*, 91–110.
Loy, G., & Zelinsky, A. (2003). Fast radial symmetry for detecting points of interest. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *25*(8), 959–973.
Mendes, A., Bento, X., & Nunes, L. (2004). Multi-target detection and tracking with a laser scanner. In *Intelligent vehicles symposium*.
Mittrapiyanuruk, P., DeSouza, G. N., & Kak, A. C. (2004). Calculating the 3d-pose of rigid-objects using active appearance models. In *IEEE international conference on robotics and automation* (pp. 5147–5152). New York: IEEE.
Monkey Helpers (2009). Helping hands: Monkey helpers for the disabled. http://www.monkeyhelpers.org.
Montemerlo, M., & Thrun, S. (2002). Conditional particle filters for simultaneous mobile robot localization and people-tracking (slap). In *Proc. int'l conf. robotics and automation*.
Muller, A., Kirsch, A., & Beetz, M. (2007). Transformational planning for everyday activity. In *17th international conference on automated planning and scheduling* (pp. 248–255).
Nguyen, H., Anderson, C., Trevor, A., Jain, A., Xu, Z., & Kemp, C. (2008). El-e: an assistive robot that fetches objects from flat surfaces. In *Proc. human robot interaction, the robotics helpers workshop*.
Pereira, G. A. S., Kumar, V., & Campos, M. F. M. (2002). Decentralized algorithms for multirobot manipulation via caging. In *Proceedings of the workshop on the algorithmic foundations of robotics*.
Prats, M., Sanz, P. J., & del Pobil, A. P. (2008). A sensor-based approach for physical interaction based on hand, grasp and task frames. In *Proceedings of the manipulation workshop in robotics science and systems*.
Quigley, M., Gerkey, B., Conley, K., Faust, J., Foote, T., Leibs, J., Berger, E., Wheeler, R., & Ng, A. (2009). (ros): an open-source robot operating system. In *ICRA workshop on open source software in robotics*.
Ray, C., Mondada, F., & Siegwart, R. (2001). What do people expect from robots? In *Intelligent robots and systems, 2008 proceedings 2008 IEEE/RSJ international conference* (pp. 3816–3821).
Saxena, A., Driemeyer, J., & Ng, A. (2008). Robotic grasping of novel objects using vision. *The International Journal of Robotics Research*, *27*(2), 157–173.
Schulz, D., & Burgard, W. (2001). People tracking with a mobile robot using sample-based joint probabilistic data association filters. *Robotics and Autonomous Systems*, *34*(2–3), 107–115.
Schulz, D., Burgard, W., Fox, D., & Cremers, A. (2003). People tracking with a mobile robot using sample-based joint probabilistic data association filters. *The International Journal of Robotics Research*, *22*(2), 99–116.
Taylor, G., & Kleeman, L. (2003). Fusion of multimodal visual cues for modelbased object tracking. In *Australasian conference on robotics and automation (ACRA2003)*, Brisbane, Australia.
The personal robotics project (2008). http://pr.willowgarage.com.
Vacchetti, L., Lepetit, V., & Fua, P. (2004). Stable real-time 3d tracking using online and offline information. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *26*(10), 1385–1391. doi:10.1109/TPAMI.2004.92.
Walter, J., & Arnrich, B. (2000). Gabor filters for object localization and robot grasping. In *IEEE proceedings of the international conference on pattern recognition* (p. 4124). Washington: IEEE Computer Society.
Wang, C. C., & Thorpe, C. (2002). Simultaneous localization and mapping with detection and tracking of moving objects. In *Proc. int'l conf. robotics and automation*.
Zerofrog (2008). Libsiftfast. http://sourceforge.net/projects/libsift.
Zhang, J., Schmidt, R., & Knoll, A. (1999). Appearance-based visual learning in a neuro-fuzzy model for fine-positioning of manipulators. In *IEEE international conference on robotics and automation* (p. 1164).
Zickler, S., & Veloso, M. (2006). Detection and localization of multiple objects. In *Humanoid robots, 2006 6th IEEE-RAS international conference* (pp. 20–25). doi:10.1109/ICHR.2006.321358.

Siddhartha S. Srinivasa is a Research Scientist with Intel Research Pittsburgh. He also holds an Adjunct Faculty position at the Robotics Institute at Carnegie Mellon University. He is a co-PI of the Personal Robotics project, in which an anthropomorphic robotic arm and a mobile robot coordinate to accomplish useful manipulation tasks in populated indoor environments. His research focuses on enabling robots to interact faster, better, and smoother with the real world. He received his Ph.D. from the Robotics Institute at Carnegie Mellon University where he developed robust controllers for robotic manipulation. He also has a B. Tech in Mechanical Engineering from the Indian Institute of Technology Madras.

Dave Ferguson is a Researcher at Two Sigma Investments. He received his Ph.D. in Robotics from the Robotics Institute at Carnegie Mellon University in 2006. His robotics research focused on planning and coordination for single agents and multi-agent teams. His algorithms have been used on a number of real-world autonomous systems including the Mars Exploration Rovers, subterranean mine mapping robots, and Tartan Racing's Urban Challenge winning vehicle "Boss". Most recently, he was the co-lead of the Personal Robotics project at Intel Research, where he helped lead the development of "Herb", an autonomous mobile manipulation robot for indoor assistance.

Casey J. Helfrich is a Research Engineer at the Intel Research Lab in Pittsburgh. He received a Bachelor's degree in Physics from Carnegie Mellon Univeristy in 2001 and an additional B.S. in Computer Science from Carnegie Mellon University in 2002. He joined the Pittsburgh lab in November 2001 when the walls were still being put up, and helped design and build the IT infrastructure for Intel Research. Casey has contributed to several Intel Research Pittsburgh Projects including Internet Suspend Resume, Diamond, Dynamic Physical Rendering and Personal Robotics.

Dmitry Berenson is currently a Ph.D. student at the Robotics Institute at Carnegie Mellon University working on the Intel Personal Robotics project sponsored by the Quality of Life Technology Institute. He graduated from Cornell University in 2005 with a B.S. in Electrical and Computer Engineering. His interests include manipulation, planning algorithms, grasping, mobile manipulation, and humanoid robotics.

Alvaro Collet is a M.Sc. student in The Robotics Institute at Carnegie Mellon University. He is the primary computer vision researcher on the Personal Robotics project at Intel Research Pittsburgh. His interests include vision for manipulation, active sensing, object recognition, and sensor fusion. Alvaro graduated from Universitat Ramon Llull in 2005 with a B.S. and M.S. in Electrical and Computer Engineering.

Rosen Diankov graduated from University of California Berkeley in 2006 with Electrical Engineering and Computer Science, and Applied Math degrees. At the moment he is a Ph.D. graduate student at the Robotics Institute at Carnegie Melon University. Rosen's main research focus is to solve the robotics problem: perception, planning, visualization, and control into one coherent framework. Up until now he has worked on several vision and planning systems involving autonomous robots in everyday scenarios. One of his contributions to the robotics field is an open-source planning framework called OpenRAVE, which is helping to serve as a repository for planning algorithms. At the moment Rosen is working on the Personal Robotics project at Intel Research Pittsburgh.

Garratt Gallagher is currently pursuing a Masters of Robotics at Carnegie Mellon University. His research at CMU has focused on perception, mapping and planning in dynamic indoor environments. His research goal is to develop an indoor helper robot that can assist people in their everyday lives. He works with the Personal Robotics group at Intel Research Pittsburgh, where he helped create HERB, a robot that is designed to help people perform household tasks. In addition, he is part of the Quality of Life Technology center, a NSF funded project to develop technology that can aid the elderly and disabled.

Geoffrey Hollinger is a Ph.D. student at Carnegie Mellon University in the Robotics Institute. He is actively involved in the EMBER project to develop autonomous robots that assist first responders in disaster scenarios. His work focuses on designing scalable, near-optimal algorithms for searching unstructured environments with teams of agents. The applications of his work also include urban search and rescue, aged care, and personal robotics. He received his M.S. in Robotics from Carnegie Mellon University, and he holds a B.S. in General Engineering and a B.A. in Philosophy from Swarthmore College.

James Kuffner is an Associate Professor at the Robotics Institute, Carnegie Mellon University. He received a B.S. and M.S. in Computer Science from Stanford University in 1993 and 1995, and a Ph.D. from the Stanford University Dept. of Computer Science Robotics Laboratory in 1999. He spent two years as a Japan Society for the Promotion of Science (JSPS) Postdoctoral Research Fellow at the University of Tokyo working on software and planning algorithms for humanoid robots. He joined the faculty at Carnegie Mellon University's Robotics Institute in 2002. He has published over 100 technical papers and received the Okawa Foundation Award for Young Researchers in 2007.

Michael Vande Weghe has worked for the past ten years in mechanical, electrical, and software development at Carnegie Mellon University, where he is presently a Senior Research Engineer at the Robotics Institute. Mike is responsible for hardware development and robot control on the Intel Research Personal Robotics project. Before coming to CMU, Mike worked for Parlance Corporation and BBN on real-time computer speech recognition, and for Lutron Electronics on high-frequency switching power systems. Mike has an S.B. in Electrical Engineering from MIT, and a M.S. in Robotics from CMU.