HERB 2.0: Lessons Learned from Developing a Mobile Manipulator for the Home

Siddhartha S. Srinivasa*, Dmitry Berenson†, Maya Cakmak‡, Alvaro Collet*, Mehmet R. Dogar*,
Anca D. Dragan*, Ross A. Knepper*, Tim Niemueller[x], Kyle Strabala*, Mike Vande Weghe*, Julius Ziegler[¶]
*The Robotics Institute, Carnegie Mellon University
†EECS Department, University of California Berkeley
‡School of Interactive Computing, Georgia Institute of Technology
[x]Knowledge-based Systems Group, RWTH Aachen University
[¶]Department of Measurement and Control Systems, Karlsruhe Institute of Technology
{siddh, acollet, adragan, mdogar, rak, kstrabal, vandeweg}@cs.cmu.edu, berenson@eecs.berkeley.edu,
maya@cc.gatech.edu, niemueller@kbsg.rwth-aachen.de, ziegler@kit.edu

***Abstract*—We present the hardware design, software architecture, and core algorithms of HERB 2.0, a bimanual mobile manipulator developed at the Personal Robotics Lab at Carnegie Mellon University. We have developed HERB 2.0 to perform useful tasks for and with people in human environments. We exploit two key paradigms in human environments, that they have structure that a robot can learn, adapt and exploit, and that they demand general-purpose capability in robotic systems. In this paper, we reveal some of the structure present in everyday environments that we have been able to harness for manipulation and interaction, comment on the particular challenges on working in human spaces, and describe some of our lessons learned from extensive testing in kitchens and offices with our integrated platform.**

I. INTRODUCTION

Robots perform remarkable dexterous tasks routinely in factories. They assemble a wide range of products, from cars to microprocessors, often with super-human precision and speed. As a consequence, factories are filled with robots. The lack of abundance of robots in our homes might seem puzzling at first: surely, something capable of assembling a car, a task that few of us can claim to be capable of performing, should find the task of clearing a dining table after a meal, a task that almost all of us can claim to be capable of performing, trivial.

In trying to explain this paradox, researchers often claim that factories are *structured* whereas our homes, with their clutter and messiness, are *unstructured*. But that is perhaps oversimplifying: a bin of nuts and bolts in a car assembly plant or the inside of a chassis are perhaps as cluttered (if not more) than the average kitchen in our homes. Perhaps a deeper difference is that factories are *structured for robots* whereas our homes are *structured for humans*. One might argue that we might be as confused in a car factory, as a factory robot is in our kitchen. But, given time, we adapt to the structure that is presented to us. Likewise, we would like robots in out homes to understand, adapt to, and eventually utilize the structure that is present in our homes.

Another key difference is *generality*. A car factory is massive, with hundreds of robots spread over hundreds of square feet, each performing a few specific tasks. In contrast, a car mechanic's workshop is small, with a few skilled mechanics spread over a few square feet, each performing a multitude of tasks. The confines of a human environment, built for a general-purpose manipulator like the human, compel robots in such environments to also strive to be general purpose: there just isn't enough space for hundreds of specific-purpose robots.

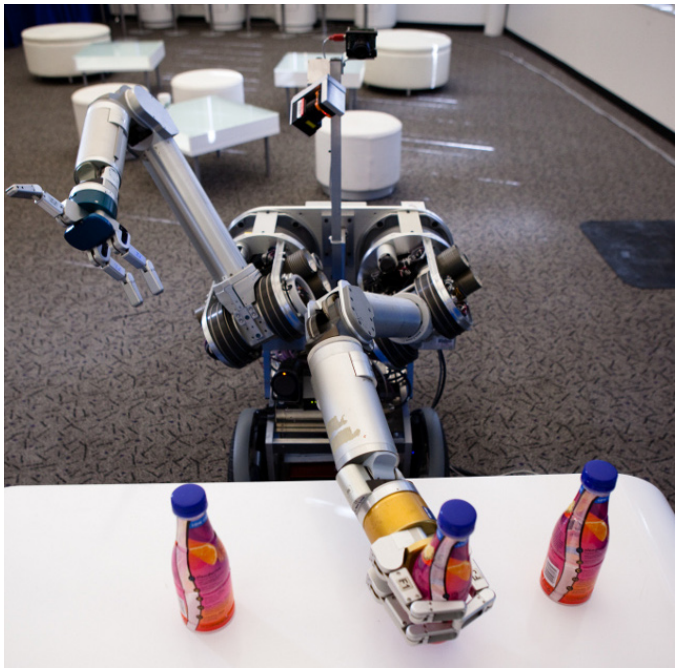

Fig. 1. HERB 2.0: A bimanual mobile manipulator developed at the Personal Robotics Lab at Carnegie Mellon University

At the Personal Robotics Lab at Carnegie Mellon University, we are developing algorithms to enable robots to perform useful tasks for and with people in human environments. To this end, we have designed and built a series of increasingly capable mobile manipulators starting from the BUSBOY [1]: a mobile base coordinating with a fixed arm, HERB [2]: an integrated mobile base and arm, to the current version HERB 2.0: a bimanual mobile manipulator.

The two paradigms, of structure and generality, resonate through all of our decisions, from the design of the hardware

(Section III) and software architecture (Section IV) to the algorithms for cognition (Section V), planning (Section VI, VII, VIII), perception (Section X), navigation (Section XI), and interaction (Section IX).

In the sections that follow, we will reveal the structure present in everyday environments that we have been able to harness for manipulation and interaction, comment on the particular challenges on working in human spaces, and describe some of our lessons learned from extensive testing in kitchens and offices with our integrated platform.

II. A Brief History of Mobile Manipulation

We build upon a long history of integrated mobile manipulation systems that combine navigation, perception, motion planning, and learning. In this section, we will briefly trace out some of that history. The sections that follow will each have their own background work specific to their subtopic.

SHAKEY [3] was possibly the first (well-documented) mobile manipulator. Developed by SRI from 1966–72, SHAKEY was equipped with an onboard SDS-940 computer, a TV camera, and other sensors atop a wheeled base. Developed primarily as a testbed for AI planning, SHAKEY navigated autonomously through corridors and pushed large boxes that were in its way using its base as a manipulator.

FREDDY II [4] was being developed at the University of Edinburgh at about the same time. It was made up of a large robot arm with fixed to an overhead gantry, with a binocular vision system for perception. While SHAKEY manipulated without an arm, FREDDY II navigated without a wheeled base. Its world consisted of a table that could be moved in two directions, giving FREDDY II the impression of motion. This was a truly remarkable robot, able to assemble wooden models using vision to identify and locate the parts – given a jumbled heap of toy wooden car and boat pieces it could assemble both in about 16 hours using a parallel gripper and single camera. FREDDY II asked and often answered some of the fundamental questions that still plague modern mobile manipulators: of perception and manipulation in clutter, and of force-controlled assembly.

HANDEY [5], [6] was developed in the 80s at MIT with the goal of performing general pick-and-place tasks over a very broad range of objects. Unlike previous robots, that posed manipulation as a blocks-world style AI problem, HANDEY posed manipulation as geometric search, striving to find collision-free paths in the robot's configuration space. HANDEY produced numerous advances, including geometry-based object detection, grasp tables, regrasping, and the concept of configuration spaces as a means to abstract and generalize algorithms across robot morphology and kinematics, a concept upon which our motion planning algorithms are based on.

Several other platforms, like the JPL CART [7], ROMEO AND JULIET [8], and HILARE 2BIS [9], populate the history of mobile manipulation.

Humanoid robots, with manipulators atop legged bases, provide a drastic design departure from the classic wheeled-base mobile manipulators. While they provide greater rough-terrain locomotion dexterity, humanoids often have to deal with several additional constraints like balance, walking, and power density. Honda's ASIMO robots [10], starting with E0 in 1986, and the H6 [11], H7 [12], and the HRP series of robots [13]–[15] developed in Japan, have demonstrated remarkable one- and two-armed mobile manipulation.

HERB 2.0 joins a modern list of mobile manipulators, including the PR2 [16], JUSTIN [17], TUMROSIE [18], EL-E [19], the STAIR project [20], among many others. The design of these robots still echoes their illustrious predecessors: wheeled bases with capable manipulators atop them. The advances in sensor and hardware designs, computing capability, and algorithms, some of which we shall describe in the following sections, have enabled these robots to often produce super-human capability in mobile manipulation.

Like most histories, this one is also both personal and incomplete. Mobile manipulators have a rich history, from the 60s, and have demonstrated great capability and promise.

III. Hardware Design

At the Personal Robotics Lab we require a platform that can operate synergistically with humans to perform tasks in the home environment. The design of our robot HERB 2.0, therefore, reflects our research interest in human-aware two-arm manipulation of unstructured environments. HERB 2.0's hardware allows it to navigate indoors for hours at a time, sense its surroundings, and manipulate objects of interest for or with human partners, with minimal reliance on supporting infrastructure.

A. Actuation

HERB 2.0's base is comprised of a Segway RMP that operates in "tractor" mode, with a rear caster installed for passive balancing. The placement of components on HERB 2.0 was carefully chosen to shift the center of gravity rearward so that weight is maintained on the caster even when suddenly stopping at the software-limited maximum deceleration. The Segway was chosen because of its high payload capacity, smooth velocity control, reliable operation, low noise, convenient USB interface, and open-source host interface software.

HERB 2.0 manipulates its environment with a pair of Barrett 7-DOF WAM arms and Barrett hands. The WAM arms have proven themselves to be great choices for working alongside humans, due to their comparatively low mass, backdriveability, and hardware-implemented safety system. Additionally, we have further enhanced their safety by sensing and reacting to position disturbances that indicate physical collisions, and by keeping the operating speeds to a reasonable level so that humans are not surprised by sudden motions. Because the arms are commanded with an open protocol that requires no proprietary drivers, we have been able to write our own host-based closed-loop control software that can switch control laws according to application and sensory input.

The configuration of the two arms with respect to the base was chosen after careful consideration of a number of alternatives. With the two arm bases mounted side-by-side facing forward, HERB 2.0 maximizes the workspace in which both hands can reach the same point while still

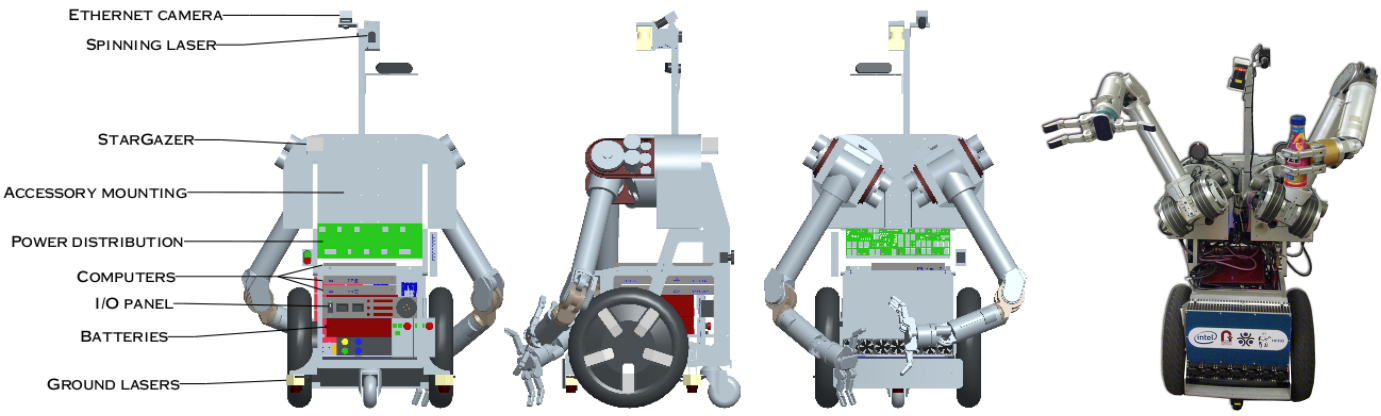

Fig. 2. Rear, side, and frontal renderings of the HERB 2.0 design, and the completed robot.

remaining narrow enough to fit through a 28" wide doorway. The mounting height allows HERB 2.0 to grasp objects from both the floor and from high shelves in overhead cabinets. Finally, by making a simple modification to the first joint on each arm, we were able to locate the largest dead zone of the configuration space to an area which would least affect our primary task, the manipulation of tabletop objects.

B. Sensors

An array of three miniature laser rangefinders mounted just above ground level gives HERB 2.0 360-degree perception of obstacles that pose a hazard to navigation, as well as provide readings for a planned self-docking capability to be developed in the future. A fourth higher-power laser is mounted on a custom-built spinning base to produce full 3D point clouds of the surrounding environment. By directing the spin axis towards the primary workspace, the laser generates a point cloud with a higher density of points right in front of HERB 2.0, thereby maximizing the utility of the data for our most common tasks. The spinning mechanism allows for variable spin rates in order to control the overall cloud resolution, and features dedicated hardware to capture the precise rotation angle for each laser scan so that the 3D points can be accurately assembled. The resulting point clouds are used primarly by the arm-motion planning software (Section VI) to avoid collisions with unexpected obstacles in the work area, such as nearby people and moveable furniture.

HERB 2.0 features three camera-based sensors for object recognition and localization. HERB 2.0 has a monochrome gigabit-ethernet camera for recognizing household objects in the workspace (Section X). This particular camera model was chosen for its high sensitivity, so that even when restricted to the lighting conditions present in typical indoor environments we can take short exposures in order to minimize motion blur. The initial design of HERB 2.0 fixed the camera to a stationary mast so that it was pointed forward and down, but such a configuration requires HERB 2.0 to move its base in order to pan to areas outside of its field of view. The next version of HERB 2.0 will include a pan/tilt head so that HERB 2.0 can attain a larger perception range while the base remains stationary. HERB 2.0 also has an upward-facing infrared camera for localizing the base with respect to ceiling-mounted retro-reflective markers. The self-contained unit includes infrared LEDS for lighting the markers, and features an overall resolution of 2cm position and 10 degree rotation. Finally, HERB 2.0 uses a popular RGB-D gaming camera to track humans in the environment. The RGB-D camera is located offboard to get a good view of the scene.

Additionally, HERB 2.0 makes extensive use of the sensors built into the Segway base and the WAM arms and hands. The Segway odometry is synthesized with the StarGazer output in order to provide a pose estimate that outperforms either sensor working alone. The WAM is equipped with both joint position sensors and an end-effector force/torque sensor that detect disturbances applied by the environment, while the fingers on the Barrett hands feature position, strain, and tactile sensors that produce critical feedback while manipulating objects.

C. Computing and Networking

HERB 2.0's computing configuration allows it to perform all essential processing onboard, reducing its dependence on the wireless network or offboard computers for the bulk of its tasks. Three high-performance mobile workstations ("laptop" form factor) with hyperthreaded quad-core processors make up the primary compute capacity. Each machine includes a high-performance GPU that is capable of running our vision-processing system (Section X), and a large solid-state drive for fast throughput with low power consumption.

Although the laptop configuration brings with it unnecessary peripherals (we make no use of the LCD screens, keyboards, or media drives), a comparison against both desktop system boards and industrial embedded-system processor boards yielded laptops the clear winner. Laptop systems typically have a much higher ratio of computing performance to consumed power, feature heatpipe cooling systems that cool both the CPU and GPU in a single compact unit, and often have a wider range of peripheral interfaces (such as IEEE-1394) integrated into the motherboard. We have adapted our laptops to HERB 2.0's needs by adding remote power buttons to the rear panel, so that the laptop lids need never be opened, and by removing the internal lithium-ion batteries so that we have more direct control over the power drawn from HERB 2.0's DC supply.

HERB 2.0 also uses an ARM-based embedded computer to interface with low-level hardware onboard. The unit features a

Xilinx FPGA along with a generous assortment of analog and digital I/O lines and a few H-bridge motor controllers. We currently use it to control the spinning laser unit, and have plans to use it to perform on/off control of the onboard power loads and to drive indicator lights to communicate high-level intent to the user.

The three laptops, the embedded computer, and the ethernet camera are all tied together with a gigabit-ethernet switch for a fast onboard network, with a connection to a wireless bridge for communication with offboard hosts. The use of a wireless bridge (as opposed to a wireless router) makes all hosts, both onboard and offboard, visible on the same network without any network address translation, which greatly simplifies development and testing.

D. Power

HERB 2.0's onboard power system allows it to operate untethered for hours at a time, even when fully active. The system produces 48, 24, 19.5, 12, 7.4, and 5 volts using Vicor DC/DC converters supplied by a pair of rechargeable ModEnergy lithium-ion battery packs. The converters are mounted to a heatsink whose size was chosen so that convective airflow produces enough passive cooling during normal operation, while temperature-triggered variable-speed fans augment the cooling as required during peak consumption periods. Recharging occurs at twice the discharge rate, so HERB 2.0 can operate indefinitely at a 67% duty cycle. The battery packs offer a serial interface for monitoring charge level and current flow, and 7-segment LED displays mounted on the rear panel give the user a quick indication of battery voltage and current draw. A bank of software-controlled relays provide several contacts for each voltage level, so that individual loads can be turned on or off through software control. The power system was designed for hot-plug recharging, which is currently provided by a manually-connected tether and plug but which will be replaced by a docking system for autonomous recharging.

E. Lessons learned

After designing and using three instantiations of HERB, we can offer some guidelines for building a reliable platform for productive robotics research. First, make the hardware interface easy for non-experts to understand, so that everyone from regular users to visiting researchers feel comfortable using the robot and are not likely to damage it through ignorance. That means extending power switches and peripheral connections so that components do not need to be physically removed to make connections, and documenting how to start and stop the various onboard systems. Second, minimize the need for future upgrades but still be prepared for them. Each components should be carefully selected and proven offboard before committing to it, so that its utility won't be outgrown quickly. Because of the nature of robotics research, however, expect that better hardware (sensors in particular) will be available down the road, and make provisions for mounting and powering accessories beyond the initial set. Finally, make an effort at every turn to keep power consumption low. Lower consumption results in reduced conversion losses, reduced cooling requirements, and reduced battery requirements, thereby eliminating excess bulk and shortening recharge times.

IV. SYSTEM ARCHITECTURE

A key challenge for robot systems in the home is to produce safe goal-driven behavior in a changing, uncertain and dynamic environment. A complex system like HERB 2.0, that has a host of sensors, algorithms, and actuators, must address issues ranging from software robustness (sensors failing, processes dying, communication failure) to problems that emerge from inaccurate or unknown models of the physical world (collisions, phantom objects, sensor uncertainty). To address this challenge, HERB 2.0 uses a software architecture loosely based on the sense-plan-act model to provide safe and rich interactions with humans and the world.

A. Abstract components

Figure 3 shows the interaction between the different components of our architecture. We gather information about the world which is composed of fixed and dynamic objects, agents like humans and other robots, and semantics like HERB 2.0's location in the home. We gather data from a wide range of sensors including high-definition cameras, the Microsoft Kinect, lasers scanners, a localization system (StarGazer), Segway encoders, as well as HERB 2.0's joint encoders and force sensors. Analysis of the sensor data over time allows us to perceive and model both objects and actions in the world.

HERB 2.0 has three classes of plan components that can make decisions: safety, agent, and teleop. Safety components ensure that HERB 2.0 does not harm humans, the environment, or itself. Some examples of safety components include the limitation of forces that the arm is allowed to exert and the limitation of joint velocities. Safety limits cannot be overridden by other components. Agent components try to accomplish goals based on HERB 2.0's perception of the world. These goals include manipulating objects in the world, expressing via gestures, and physically interacting with humans. Much of the agent programming deals with edge cases where the robot tries to correct for perception errors or system failures (Section V). The teleop components are human interfaces where the human can explicitly tell the robot what to do, both at a low level like moving single joints or at a high level like grasping an object. Teleop components override agents but cannot override safety components.

Finally, the act components perform actions to accomplish the tasks commanded by the plan components. This includes planning trajectories and base motion, servoing the motors, making sounds, and interacting with other computer systems. In addition, each act component advertises to the user interfaces the subset of actions it is capable of performing at that given time. For example, if HERB 2.0 is holding an object and moves close to a recycling bin, the action planner advertises the recycle action.

There are some interesting implications to using this system architecture. First, because all actions must go through the

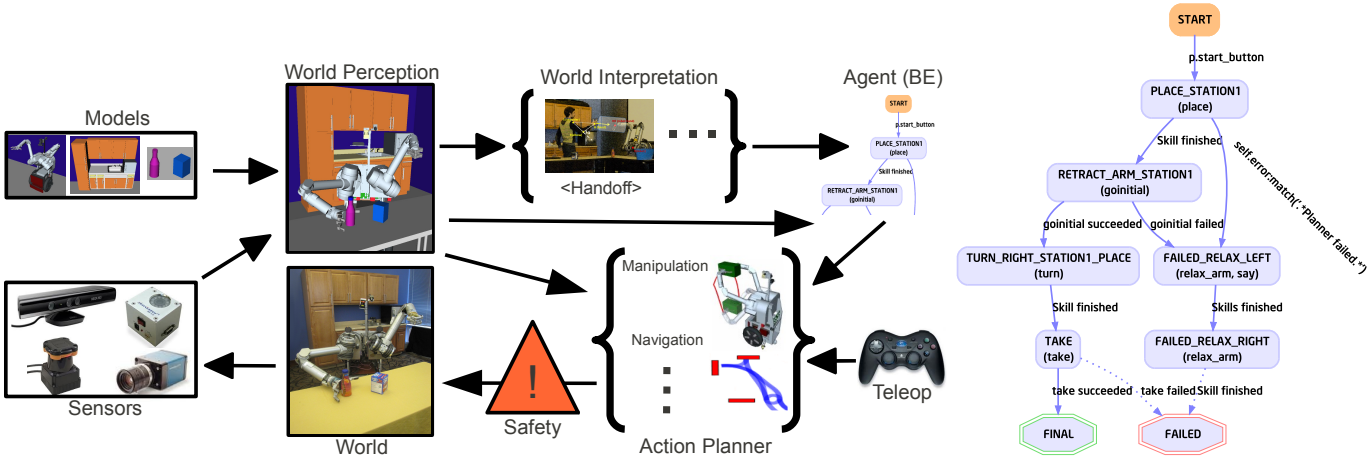

Fig. 3. (Left) HERB 2.0 system architecture. (Right) Excerpt from the Intel Open House 2010 state machine.

safety components before and during execution, it is impossible for higher level components to execute unsafe actions. Second, because of the modularity, the designers can decide at which level the decisions are made. For example, suppose the agent commands the action planner to pick up a full coffee mug. Should the agent also tell the action planner not to tip the mug, should the planner know to not tip the mug from its own information, or should the modeled mug object have a property of not being tippable as identified by the perception system? In this example, HERB 2.0's action planner decides to keep the mug upright based on its own knowledge that the mug may have liquid in it, however, this decision could easily be made by the agent or the perception system without loss of functionality. Third, due to the parallel and modular properties of this system architecture, the designer has control over what kind and how much information each component shares with the other components. One can see the parallels between these system properties and those of software architecture. It would be interesting to see if the same design principles used in software architecture can be used in the design of robot systems and their interactions.

B. Implementation

HERB 2.0's software consists of several nodes that operate in parallel and communicate to one another using the Robot Operating System, ROS [21]. ROS provides a framework for inter-node communication along with an online database of open source nodes. HERB 2.0's nodes are launched over ssh and VNC from an offboard computer. ROS offers modularity and extensibility and our nodes are written in C++, Python, Lua, and Lisp. ROS has become more stable over the years and some of our nodes, in particular the sensor drivers and visualization tools, are from community-created open-source ROS repositories. The core of HERB 2.0's intelligence — perception, planning, control, behavior engine, and navigation — was developed in our lab. The following sections discuss each of these nodes in greater detail

C. Lessons learned

Robot systems, and software systems in particular, are in a state of constant flux. We have added and moved around sensors, added an arm, reconfigured lasers, and are constantly updating the software. Core software engineering concepts, of modularity and extensibility, become all the more important when dealing with a system that physically interacts with the world. We have also found the ability to quickly prototype nodes and swap them in at runtime to be of great use.

V. BEHAVIOR ENGINE

One of the challenges of performing complex tasks in the home is to implement a system that allows for the description of behavior the robot should perform in an easy and extensible way. Robot behavior specific languages like Colbert [22] and XABSL [23] have been proposed in the past which allow a concise description of robot behavior. But these languages often impose restrictions in terms of expressiveness. To accomplish both, brevity and clarity in the robot behavior description on the one hand and providing a full programming environment that avoids restrictions in expressiveness and allows for flexibility on the other hand, we proposed the *Behavior Engine (BE)*. It uses the formalism of hybrid state machines (HSM) to model robot behavior and separates the overall behavior in three layers. It is implemented in the interpreted programming language Lua [24]. In the following we will briefly introduce how behavior is modeled and separated into layers, and how we overcome typical problems of state machines. We describe its implementation on HERB 2.0, the particular challenges encountered, and how it fits into the ROS ecosystem. We conclude with the description of an all-day demonstration of the robot, showing the reliability of the system and the lessons learned in this process

A. Modeling Robot Behavior

HSMs are directed graphs where the nodes represent states, and edges are transitions among the states. One particular state is the currently active state. A state represents, for example, the execution of an action, perception monitoring, waiting for an event, or a combination of many of these. The transitions are defined as three-tuples of originating state, target state, and a jump condition, which is a boolean function or predicate. The jump conditions of outgoing transitions of the active state are evaluated at about 30 Hz. If and when a

jump condition evaluates to true, i.e. it fires, the transition is followed. Transitions are used, for example, to react to outcomes of actions, possibly specific to the kind of an error that occurred, or to react to certain perceptions or events. If multiple jump conditions fire, only the first transition in the order of definition is followed. More elaborate conflict resolution methods could be implemented if the need arose. Another valuable property of HSMs is that they define a set of variables that can be used to store arbitrary information not only intrinsic to a state, but global for the state machine.

The overall behavior of the robot is separated into three layers. At the lowest level are real-time, hardware driving, actuator instructing, and perception processing components. At the top is the agent program that makes strategic decisions and composes smaller behavior entities to achieve a set of goals. In between these layers is the Behavior Engine, a system for developing, executing, and monitoring reactive behavior entities called skills. It serves as a plumbing layer to connect the agent with the low-level system and to prune peculiarities to keep the action interface simple for the agent.

Each skill is encapsulated in an evaluation function with parameters, which are assigned to the internal variables when the HSM is started. For example, `goto{place="kitchen"}` could invoke a skill which composes a proper series of commands for the low-level components to move the robot to the kitchen. To support non-blocking operation, execution functions are defined for interleaved invocation. Initially, the execution function will return `RUNNING` as its status. After some time, the return value will switch to either `FINAL` or `FAILED`, depending on whether the skill execution succeeded or failed. The interface between the agent and the reactive layer is through a set of invocation channels. For each channel, a skill string composed of execution functions is evaluated and the result value returned in each cycle.

Two key aspects allow for overcoming the typical problem of a vastly increasing number of states for a state machine based approach. First, the *separation of the behavior into layers* guides a separation of the overall behavior into smaller entities. From the top-level agent, the complexity of the HSMs of the individual skills is hidden, because the execution functions are the only interface. Therefore, these state machines need not be a part of the overall behavior state machine. Additionally, this fosters reuse of skills and composition into more elaborate behaviors. Second, the *set of variables* allows for keeping arbitrary data and additional state information outside of the classical notion of state machines which keeps this intrinsic to a state. This pragmatic approach therefore introduces specific side effects which can lead to a drastic reduction of the required state space.

B. Implementation on HERB 2.0

The Behavior Engine was originally used for humanoid and wheeled soccer robots and based on the Fawkes robot software framework [25]. For HERB 2.0, the BE has been ported to the ROS robot software system [21], which especially meant providing a Lua client library for ROS and coping with a different middleware. We have developed *roslua*, a client library written in Lua for ROS[1]. A client library allows communicating and interacting with other ROS nodes and is specific to a programming language.

Several advantages of Lua made it the programming language of choice for the BE. In particular, its powerful central table data structure allows to emphasize the description of behavior over programming it. The table constructor syntax was inspired by BibTeX [26], which allows to provide a syntax for the definition of behavior that goes without a large number of explicit function calls as required, for example, in SMACH [27]. This way we reach close to the clarity of domain-specific languages, but without sacrificing the abilities a complete programming environment provides.

To invoke basic actions, ROS' actionlib is used. It uses topic communication to invoke, monitor, and possibly preempt actions. In the BE, this is used for communication between the mid-level reactive system and the high-level agent as well as to invoke lower-level actions. We implemented actionlib_lua to interact with other nodes from Lua[2].

The BE was adapted to the specific challenges for domestic service robots in general and HERB in particular, for example dealing with two-armed manipulation and action concurrency. We have also started working on improving the behavior robustness by providing fail-over mechanisms whenever possible, and determining situations when human assistance was necessary.

C. Lessons Learned

In September 2010 we demonstrated HERB 2.0 at the Intel Labs Pittsburgh Open House. The robot's task was to pickup a full bottle from a counter, take it to the audience, and hand it over to a human. If no human would take the bottle after a little while it was placed on a nearby table. The robot then requested empty bottles to take back. It then drove back to the counter. If it got handed a bottle, it weighed it to determine if the bottle was actually empty or if it was full. Empty bottles were dropped into a recycling bin, while full bottles were put back on the counter. The robot continuously performed this task for about six hours. Human assistance was surprisingly infrequent (about 2-3 times an hour) and almost always requested for by the robot because it realized that it was in an unexpected or dangerous situation. During the operation, visitors would frequently pass the way of the robot while driving, or the arm could be hold back to demonstrate its compliant motion. Some software components were under active development, and therefore recovery from certain conditions required some effort. For example, there could be a discrepancy between the robot's belief and the actual arm poses. This could lead to the assumption that the robot was in collision with an object. Time constraints during the development of the task made it infeasible to come up with a recovery strategy for such a case.

These uncertainties, problems, and current limitations needed to be reflected in the robot's behavior. In the worst case, for example if posture belief indicated a collision, the robot would go into a dialog mode, in which it would request

[1] Source code and documentation at http://www.ros.org/wiki/roslua

[2] BE, actionlib_lua and demo at http://www.ros.org/wiki/behavior_engine

help from a human operator. The arm was set into a gravity compensation mode in which the operator could move it freely. Once the collision was resolved, a button on the robot was pressed and it would start over. Nowadays, much more fine grained recovery strategies have been implemented. For example, if the pose estimation of objects is slightly off, the robot can now resolve many collision situations by itself.

An excerpt from the state machine is shown in Fig.3(Right). It covers the placing of the bottle on the table if it was not taken, and on success turning back to the audience and taking an empty bottle. In case moving the arm to its initial position fails, it relaxes both arms and requests help. The upper right transition shows how specific errors can be determined by analyzing an error string.

We started off implementing the behavior by creating the state machines for the individual skills, the basic abilities of the robot. For certain classes of actions like manipulation, which required interaction with a specific submodule in a unified way, we created an auto-generator which would create the state machines on the fly. Others were created manually. The skills were then individually tested and debugged. Eventually the agent state machine was created, composing the skills to accomplish the overall task. Some time was required to define proper fail-over behaviors in the case other behaviors failed, and to get the interaction of skills right.

D. Outlook

We have demonstrated that the system is capable of producing robust and stable robot behavior for an extended period of time. By analyzing the system state, we can request operator assistance if required. Although we can recover from an increasing number of failures with the existing tools, we strive for a more general description of error conditions, recovery, and resuming of the original task. We have yet to see how far we can scale the system in terms of comprehensibility of the state graph. We assume that we can create more complex skills and to have a stronger emphasis on hierarchically structured state machine (which is already supported but not used, yet) to slow down the growth of the behavior description complexity.

VI. Manipulation Planning

Motion planning for a mobile manipulator like HERB 2.0 is difficult because of the large configuration space of the arms and the constraints imposed on the robot's motion. Some of these constraints arise from the torque limits of the robot or the necessity of avoiding collision with the environment. However, some of the most common constraints in manipulation planning involve the pose of a robot's end-effector. These constraints arise in tasks such as reaching to grasp an object, carrying a cup of coffee, or opening a door. As a result, researchers have developed several algorithms capable of planning with end-effector pose constraints [30]–[35]. Though often able to solve the problem at hand, these algorithms can be either inefficient [30], probabilistically incomplete [31]–[33], or rely on pose constraint representations that are difficult to generalize [34], [35].

We have developed a manipulation planning framework [28] that allows robots to plan in the presence of constraints on end-effector pose, as well as others. Our framework has three main components: constraint representation, constraint-satisfaction strategies, and a sampling-based approach to planning. These three components come together to create an efficient and probabilistically-complete manipulation planning algorithm called the Constrained BiDirectional RRT (CBiRRT2). The underpinning of our framework for pose-related constraints is our Task Space Regions (TSRs) representation.

TSRs describe volumes of permissible end-effector pose for a given task. For instance, for a reaching-to-grasp task TSRs can be used to define the set of end-effector poses that result in stable grasps. For picking up a cup of water, TSRs can define the set of poses in which the water does not spill. TSRs are intuitive to specify, can be efficiently sampled, and the distance to a TSR can be evaluated very quickly, making them ideal for sampling-based planning. Most importantly, TSRs are a general representation of pose constraints that can fully describe many practical tasks. For more complex tasks, such as manipulating articulated linkages like doors, TSRs can be chained together to create more complex end-effector pose constraints [36]. TSRs can also be used to construct plans that are guaranteed to succeed despite uncertainty in the pose of an object [37].

Our constrained manipulation planning framework also allows planning with multiple simultaneous constraints. For instance, collision and torque constraints can be included along with multiple constraints on end-effector pose [38]. Closed-chain kinematics constraints can also be included as a relation between end-effector pose constraints without requiring specialized projection operators [39] or sampling algorithms [40].

We have applied our framework to a wide range of problems, both in simulation and in the real world (Figure 4). These problems include grasping in cluttered environments, lifting heavy objects, two-armed manipulation, and opening doors, to name a few.

These examples demonstrate our framework's practicality, but it is also important to understand the theoretical properties of manipulation planning. Specifically, we would like to understand whether various sampling methods are able to fully explore the set of feasible configurations. To this end, we provided a proof for the probabilistic completeness of our planning method when planning with constraints on end-effector pose [41]. The proof shows that, given enough time, no part of the constraint manifold corresponding to a pose constraint will be left unexplored, regardless of the dimensionality of the pose constraint. This proof applies to CBiRRT2 as well as other approaches [30], [42], whose probabilistic completeness was previously undetermined.

A. Lessons learned

One criticism of TSRs is that the constraint representation may not be sufficiently rich. For instance, some modifications to TSR Chains are necessary to accommodate constraints where degrees of freedom are coupled (as with screw constraints). Indeed, TSRs and TSR Chains cannot capture every

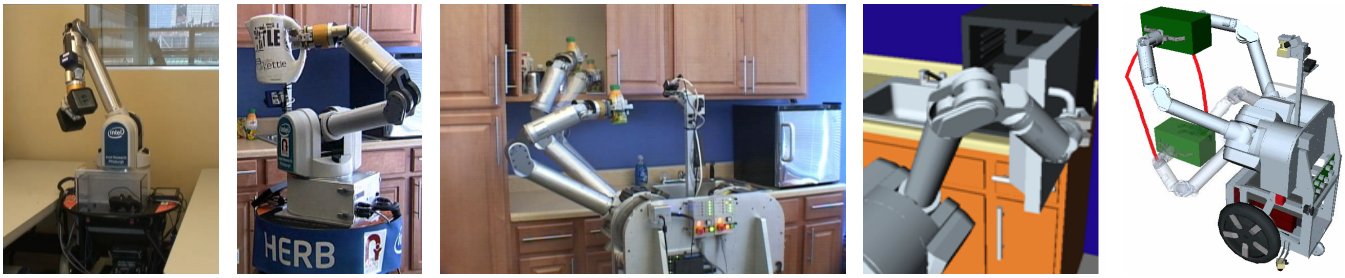

Fig. 4. HERB 2.0 and HERB 2.0 2.0 executing paths planned by our constrained manipulation planning framework. Images taken from [28] and [29].

conceivable constraint, nor are they intended to. Instead, these representations attempt to straddle the trade-off between practicality and expressiveness. TSRs have proven sufficient for solving a wide range of real-world manipulation problems while still remaining relatively simple and efficient to use in a sampling-based planner. While a more expressive representation is surely possible, we have yet to find one that is as straightforward to specify and as convenient for sampling-based planning.

We also found that, while it was fairly straightforward to generate TSRs for many tasks, the process became quite tedious. Thus it would be interesting to develop a system that could automatically generate the constraints corresponding to a given task. We have investigated automatically forming TSRs for grasping tasks by sampling over the space of stable grasps, clustering the grasps, and fitting TSRs to these clusters [43]. To generalize to object placement tasks, we would need to develop a different method. For instance, could a robot look at a scene and determine all the areas where a given object can be placed? Such a task would require understanding where the object *could* be placed (through grounding the idea of placing geometrically) and also taking into account user preferences for where objects *should* be placed.

VII. PLANNING UNDER CLUTTER AND UNCERTAINTY

Robotic manipulation systems suffer from two main problems in unstructured human environments: *uncertainty* and *clutter*. Consider the task of cleaning a dining table. In such a task the robot needs to detect the objects on the table, figure out where they are, move its arm to reach the goal object, and grasp it to move it away. If there is significant sensor uncertainty, the hand could miss the goal object, or worse, collide with it in an uncontrolled way. Clutter multiplies this problem. Even with perfect sensing, it might be impossible for the hand to wrap around the object for a good grasp. With both clutter and uncertainty, the options for a direct grasp are even more restricted, and often impossible.

We address the problems for manipulation in such a context. Two approaches we have taken are:

- Using the mechanics of pushing to provably funnel an object into a stable grasp, despite high uncertainty and clutter. We call this *push-grasping*.
- Rearranging clutter around the primary task with the use of motion primitives such as pushing, sliding, sweeping, and picking up.

A. Push-grasping

A push-grasp aims to grasp an object by executing a pushing action and then closing the fingers [44]. We present an example push-grasp in Fig.5(Top). Here, the robot sweeps a region over the table during which the bottle rolls into its hand, before closing the fingers. The large swept area ensures that the bottle is grasped even if its position is estimated with some error. The push also moves the bottle away from the nearby box, making it possible to wrap the hand around it, which would not have been possible in its original location.

The robot must predict the consequences of the physical interaction to find the right parameters of a push-grasp in a given scene. For this purpose, we introduce the concept of a *capture region*, the set of object poses such that a push-grasp successfully grasps it. The concept of capture region is similar to the *preimages* in the preimage backchaining approach [46]. We compute capture regions for push-grasps using a quasi-static analysis of the mechanics of pushing [47] and a simulation based on this analysis. We show how such a precomputed capture region can be used to efficiently and accurately find the minimum pushing distance needed to grasp an object at a certain pose. Then, given a scene, we use this formalization to search over different parametrizations of a push-grasp, to find collision-free plans.

Our key contribution is the integration of a planning system based on task mechanics to the geometric planners traditionally used in grasping. We enhance the geometric planners by enabling the robot to interact with the world according to physical laws, when needed. Our planner is able to adapt to different levels of uncertainty and clutter, producing direct grasps when the uncertainty and clutter are below a certain level.

B. Rearranging clutter

Tasks is human environments may require rearrangement. Imagine reaching into the fridge to pull out the milk jug. It is buried at the back of the fridge. You immediately start rearranging content — you push the large heavy casserole out of the way, you carefully pick up the fragile crate of eggs and move it to a different rack, but along the way you push the box of leftovers to the corner with your elbow.

We developed an open-loop planner that rearranges the clutter around a goal object [45]. This requires manipulating multiple objects in the scene. The planner decides which objects to move and the order to move them, decides where to move them, chooses the manipulation actions to use on these

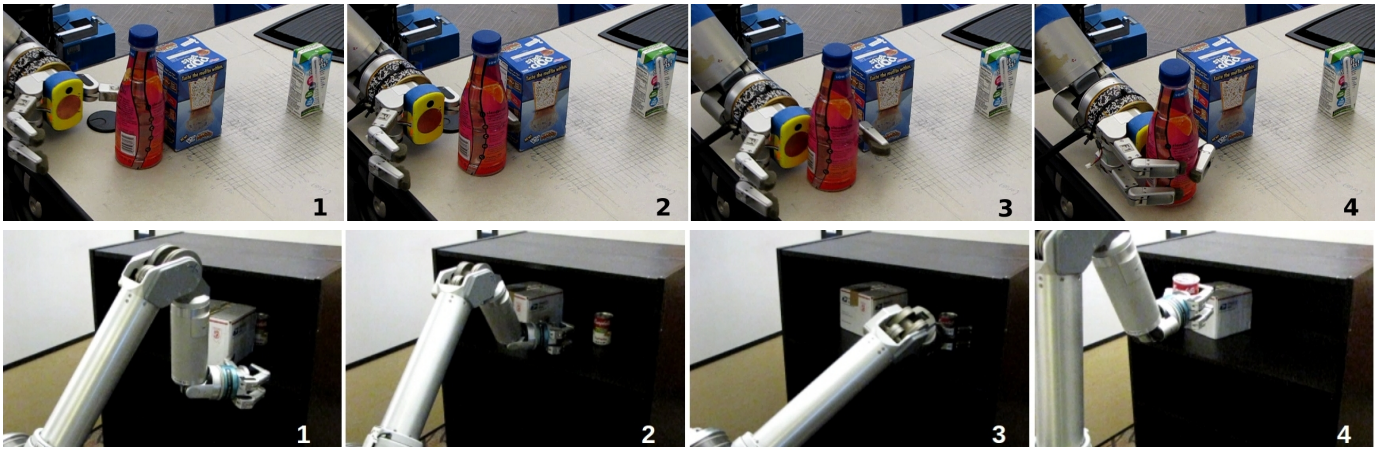

Fig. 5. (Top) An example push-grasp of an object in contact with the surrounding clutter. Image taken from [44]. (Bottom) An example rearrangement plan. The robot pushes the large ungraspable box out of the way before retrieving the goal object. Image taken from [45].

objects, and accounts for the uncertainty in the environment all through this process. One example scene is presented in Fig.5(Bottom). In this scene the robot's primary task is to grasp the can buried inside the shelf. The planner pushes the large ungraspable box to the side. This creates the space it then uses to grasp the primary goal object.

Our planner uses different non-prehensile manipulation primitives such as pushing, sliding, sweeping. The consequences of actions are derived from the mechanics of pushing and are provably conservative. Since our planner uses non-prehensile actions, it generates plans where an ordinary pick-and-place planner cannot. This enables HERB 2.0 to perform manipulation tasks even if there are large, heavy ungraspable objects in the environment, or when there is a large uncertainty about object poses.

Non-prehensile actions can decrease or increase object pose uncertainty in an environment. To account for that, our planner represents the object pose uncertainty explicitly and conservatively.

The planner plans backwards starting from the primary goal object and identifying the volume of space required to manipulate it. This space is given by the volume swept by the robot links and by the manipulated object, with the object pose uncertainty taken into account. Then, if any other object is blocking this space, the planner plans an action to move the blocking object out of the way. This recursive process continues until all the planned actions are feasible.

C. Lessons learned

The planners we have for push-grasping and for rearranging clutter are open-loop planners. On the one hand this is good because the robot does not depend on a specific sensor input that may be noisy or unavailable at times. But on the other hand the open-loop planners need to be very conservative to guarantee success. For push-grasping this can sometimes result in unnecessarily long pushes. For the rearrangement planner this can result in a quick consumption of planning space the robot can use. The lesson is: build a system that can work open-loop, but that can also integrate sensor feedback when it is available. With this line of thinking we are currently working on integrating visual and tactile sensory feedback into our system.

Clutter is not only a problem for robot actions, but it is also a problem for robot perception. If an object is hidden behind another object on a cluttered fridge shelf there is little chance that the camera on the robot's head will be able to see it. This problem motivates us to use the rearrangement planning framework also to move objects with the goal of making the spaces behind them visible to the robot sensors.

VIII. TRAJECTORY OPTIMIZATION AND LEARNING

A vital requirement for a personal robot like HERB 2.0 is the ability to work with and around people, moving in their workspaces. While the Rapidly-Exploring Random Tree algorithm from Section VI is very good at producing feasible, albeit random motion, we need to start thinking towards predictability and optimality of the paths the robot executes. With these criteria in mind, motion planning becomes a trajectory optimization problem. However, since manipulation tasks induce a very high-dimensional space, optimizers often struggle with high-cost local minima corresponding to dangerous and sometimes even infeasible paths. In this section, we present two ways of alleviating this issue: one is to improve the optimizer itself (Section VIII-A), which widens the basin of attraction of low-cost solutions; the other is to learn to initialize the optimizer in the basin of attraction of these low-cost solutions (Section VIII-B), thus ensuring convergence to a good trajectory.

A. Improving the Optimizer: Extending to Goal Sets

Most manipulation tasks, such as reaching for an object, placing it on a table or handing it off to a person, are described by an entire region of goals rather than one particular goal configuration that the robot must be in at the end of the trajectory. This section extends a recent trajectory optimizer, CHOMP [48], to take advantage of this goal region by changing the end point of the trajectory during optimization [49]. Doing so enables more initial guesses converge to low-cost trajectories.

Our algorithm, Goal Set CHOMP, optimizes a functional that trades off between a smoothness and an obstacle cost:

$$U[\xi] = \lambda f_{prior}[\xi] + f_{obs}[\xi] \quad \text{s.t. } h(\xi) = 0 \qquad (1)$$

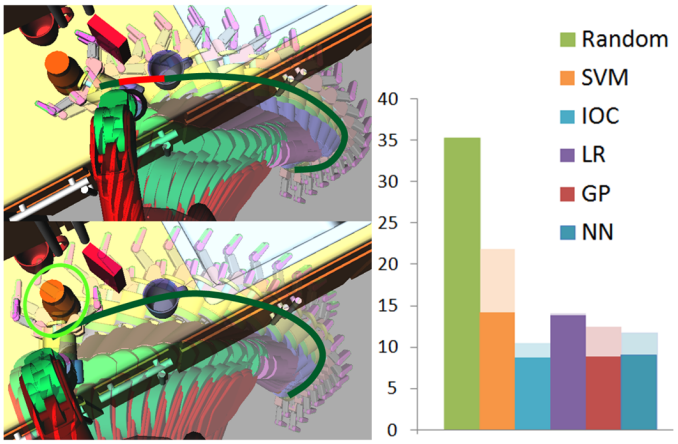

Fig. 6. Left: a comparison between a single goal optimizer that cannot find a collision-free path and the goal-set aware optimizer that shifts the trajectory end-point to avoid collisions [49]. Right: a comparison of five learning algorithms trained to predict the best goal from a goal set, against the baseline of selecting a goal randomly; we initialize the optimizer with a trajectory ending at the predicted goal, and compute the percent loss (Y axis) over the cost obtained by the best goal for 108 different reaching tasks [50].

with the prior measuring a notion of smoothness such as sum squared velocities or accelerations along the trajectory ξ, the obstacle cost pushing all parts of the robot away from collision, and h capturing constraints on the trajectory.

While CHOMP has an implicit fixed goal constraint, $\xi[n] = q_{\text{goal}}$, our extension relaxes this assumption and replaces the constraint by $h_n(\xi[n]) = 0$: the goal is restricted to an entire region rather than a single configuration.

In [49], we have shown that this extension improves upon CHOMP for a wide range of day-to-day tasks that HERB 2.0 encounters. Fig.6(left) shows a prime example of this: when using the single-goal version of CHOMP, we obtain a trajectory that does not fully avoid collisions, whereas with Goal Set CHOMP the optimizer converges to a better goal and obtains a collision-free solution.

B. Improving the Initialization: Learning to Choose Goals

With Goal Set CHOMP, we widened the basins of attraction of low-cost solutions, making initialization in such a basin more likely. In this section, we will focus on improving the initialization based on prior experiences by learning to predict the goal at which the trajectory should end.

Because of local minima, the goal choice still has a great influence on result of Goal Set CHOMP. Fig.7 shows the final cost of the trajectory as a function of what goal in the goal set the initial trajectory chooses: the difference between the best and worst performances is drastic. However, by collecting training data from such scenes, we can learn to predict an initial goal that will place the final cost within only 8% of the minimum [50]. Fig.6(right) compares the loss over this minimum cost for five different learners against the baseline of selecting a goal at random. Here, SVM is a Support Vector Machine classifier that attempts to predict whether a goal will be the best. IOC is a Maximum Margin Planning [51] algorithm that attempts to find a cost under which the best cost will be minimum. The Linear Regression (LR), Gaussian

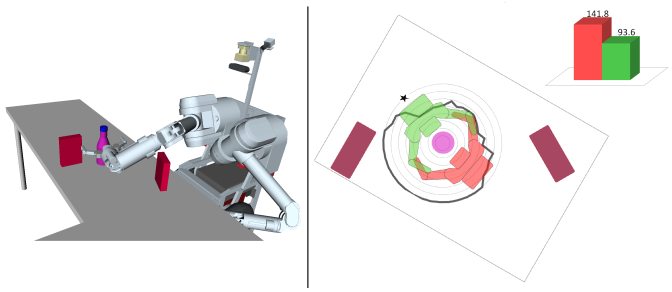

Fig. 7. Left: the robot in one of the goal configurations for grasping the bottle. Right: for the same scene, the black contour is a polar coordinate plot of the final cost of the trajectory Goal Set CHOMP converges to as a function of the goal it starts at; goals that make it hard to reach the object are associated with higher cost; the bar graph shows the difference in cost between the best goal (shown in green and marked with *) and the worst goal (shown in red) [50].

Process (GP) and Neural Network (NN) learn to map initial goals to final trajectory costs.

For each of these algorithms, we found it important to make efficient use of the available data: the classifier and the inverse optimal control method, which traditionally focus solely on the best option, should take into account the true cost of the other candidates; at the same time, the regressors, which traditionally focus on all the data, should not waste resources on the poor options. In Fig.6(right), the light colors represent the performances of the vanilla versions of these algorithms, which are not able to predict the best option as well. The data-savvy version of IOC obtains the best results by combining the focus on predicting the best goal with the awareness of costs from other goals.

C. Lessons Learned

In moving away from the random sampling approaches from Section VI, we are giving up the fast exploration that makes RRTs successful for optimal, predictable motion. Even though CHOMP is armed with an exploration technique derived from the dynamics of physical systems, namely Hamiltonian Monte Carlo, exploring while optimizing is more costly than pure exploration. An idea for merging optimization and randomized sampling is using CHOMP as the extension operator for the tree. However, such an algorithm will spend too many resources optimizing paths that do not contribute to the final solution.

At the core of this work lies the idea that while trajectory optimization in high-dimensional spaces is hard, we can make it easier in the case of manipulation by taking advantage of the structure inherent in the problem: tasks are described by sets of goals that can be exploited, and the repeatability of tasks allows for self-improvement over time. We found that even naive learning methods improve the optimization process, and that the benefit is even greater when using the data in a way tailored to the problem. For future work, we are excited about using other attributes of trajectories, beyond goal choices, to guide this learning process. Due to its predictable nature, trajectory optimization can benefit in a lot of cases from machine learning in overcoming the need for exploration.

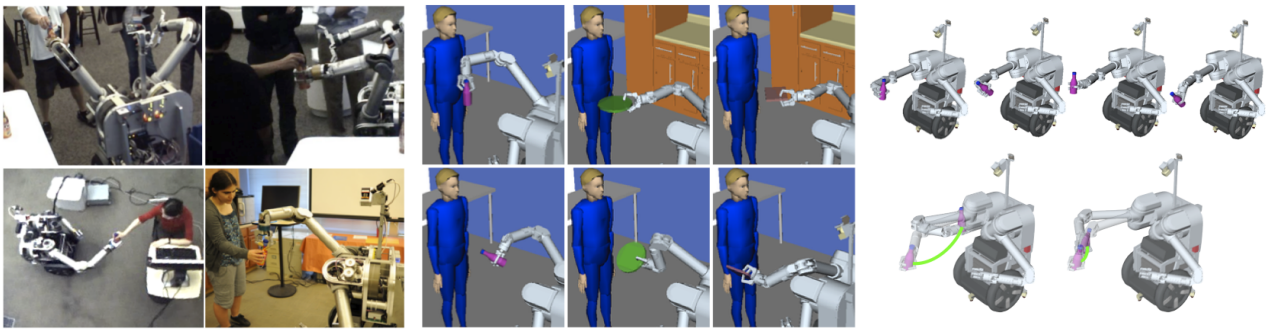

Fig. 8. Robot-human hand-overs with HERB 2.0. (Left) Handing objects to a human during *Research at Intel Day* and systematic user studies. (Middle) Hand-over configurations *learned* from human examples (top) and *planned* using the kinematic model of the human (bottom). (Right) Poses that best convey the intent of handing over (top) and two sample trajectories with high and low contrast between carrying and hand-over configurations. Left and Right images are taken from [52], Middle image is taken from [53].

IX. INTERACTING WITH PEOPLE

Personal robots that are intended for assisting humans in daily tasks will need to interact with them in a number of ways. The design of human interaction behaviors on these robots is particularly challenging since the average user will have little knowledge about how they work. It is essential to tune these behaviors for the users' expectations and preferences. To this end, we have performed a number of systematic user-studies with HERB 2.0 [52], [53] and we have put its human-interaction behaviors to test at a number of public events (Fig.8(Left)).

In designing HERB 2.0, we are particularly interested in collaborative manipulation with humans, focusing on robot-human hand-overs. Many of the potential tasks for personal robots, such as fetching objects for the elderly or individuals with motor-impairment, involve hand-over interactions. Different aspects of robot-human hand-overs have been studied within robotics, including motion control and planning [54]–[57], grasp planning [58], social interaction [59]–[61] and grip forces during hand-over [62], [63]. A few of these report results from user-studies involving hand-overs between a physical robot and a human [55], [59]–[61].

A. Robot-human hand-overs

The problem of planning a hand-over is highly under-constrained. There are infinite ways to transfer an object to a human. As a result, it is easy to find a viable solution, however it is hard to define what a *good* solution is from the human's perspective. Our approach involves parametrizing hand-over behaviors and identifying heuristics for searching desirable hand-overs in these parameter spaces.

Hand-overs involve several phases starting from approaching the human with an object, to retracting the arm after releasing the object. The object and robot configuration at the moment of transfer and the trajectory that leads to this configuration are critical. The hand-over configuration influences how the object will be taken by the human and the trajectory leading to this configuration lets the human predict the timing of the hand-over and synchronize their movements.

1) Hand-over configurations: A hand-over configuration is fully specified by a grasp on the object and a 7-DOF arm configuration. We conducted three user-studies to identify heuristics for choosing good hand-over configurations. The first study (10 participants) asked users to configure several good and bad handing configurations for five objects through a graphical user interface. Afterwards, users were asked to choose between two hand-over configurations that differed by one parameter. We found that the set of good configurations provided by participants are concentrated around a small region in the space of all configurations and has little variance across participants. These good configurations expose a large portion of the object surface, and tend to present the object in its default orientation. While confirming these results, the forced-choice questions revealed that participants prefer extended arm configurations that look natural (are mappable to a human arm configuration) [53].

In the second study (10 participants) we compared the configurations learned from good examples collected in the first study, with configurations planned using a kinematic model of the human (Fig.8(Middle)). The robot delivered each object twice, with the *learned* and *planned* configurations, and the participant was asked to compare them. Participants preferred the learned configurations and thought they were more *natural* and *appropriate*, however they had greater reachability over the objects presented with the planned configurations [53].

Besides allowing humans to easily take the object, a hand-over configuration needs to convey its intention. Our third study (50 participants) involved a survey that asks the participant to categorize the intention of a robot configuration holding an object. We find that the intention of handing an object is best conveyed by configurations with an extended arm, grasping the object from the side opposite to the human and tilting the object towards the human (Fig.8(Right)) [52].

2) Hand-over trajectories: We parametrize hand-over trajectories by the configuration in which the object is carried while approaching the human. When the robot is ready to deliver the object, it transitions to the hand-over configuration through a smooth trajectory. We conducted a fourth user-study (24 participants) to analyze the effects of the carrying configuration. We found that carrying configurations that have high contrast with the handing configuration results in the most fluent hand-overs [52]. These are configurations in which the robot holds the object close to itself, obstructing the human from taking the object (Fig.8(Right)). They improve the fluency by avoiding the human's early attempts to take the

object and by distinctly signaling the timing of the hand-over.

B. Lessons learned

We have made a lot of progress on manipulating objects in real-world environments through novel and improved techniques (Sec.VI,VII,VIII). However, adding humans into the equation imposes unique constraints, such as usability of interaction interfaces or legibility of the robot's movements, that can only be addressed through user-studies. We have seen, in the context of robot-human interactions, that such user-studies can reveal interesting heuristics that can be used in manipulation and motion planning to produce desirable and human-friendly behaviors.

X. Perception

The complexity of the tasks HERB 2.0 can perform is strongly tied to its perceptual capabilities. In the field of service/personal robotics, most tasks require interaction with objects, which we need to identify and localize prior to interacting with them. In order to operate in realistic household environments, we require robust object recognition performance in complex scenes (Fig.9 for examples), low latency for real-time operation, and scalability to a large number of objects. To address these challenges, we have developed MOPED [64], a framework for Multiple Object Pose Estimation and Detection that integrates single-image and multi-image object recognition and pose estimation. Using sparse 3D models built from SIFT features [65], MOPED recovers the identity and 6-DOF pose of objects for HERB 2.0 to interact with them (see Fig.9).

A. Iterative Clustering Estimation (ICE)

The task of recognizing objects from local features in images requires solving two sub-problems: the *correspondence problem* and the *pose estimation problem*. The correspondence problem refers to the accurate matching of image features to features that belong to a particular object. The pose estimation problem refers to the generation of object poses that are geometrically consistent with the found correspondences.

With MOPED, we developed a scalable framework for object recognition specifically designed to address increased scene complexity, limit false positives, and utilize all computing resources to provide low latency processing for one or multiple simultaneous high-resolution images. The Iterative Clustering-Estimation (ICE) algorithm is our most important contribution to handle scenes with high complexity and keep latency low. In essence, ICE jointly solves the correspondence and pose estimation problems through an iterative procedure. ICE estimates groups of features that are likely to belong to the same object through clustering, and then searches for object hypotheses within each of the groups. Each hypothesis found is used to refine the feature groups that are likely to belong to the same object, which in turn helps finding more accurate hypotheses. The iteration of this procedure focuses the object search only in the regions with potential objects, avoiding the waste of processing power in unlikely regions. In addition, ICE allows for an easy parallelization and the integration of multiple cameras in the same joint optimization.

B. Scoring and filtering object hypotheses

Another important contribution of MOPED is a robust metric to rank object hypotheses based on M-estimator theory. A common metric used in model-based 3D object recognition is the sum of reprojection errors. However, this metric prioritizes objects that have been detected with the least amount of information, since each additional recognized object feature is bound to increase the overall error. Instead, we propose a quality metric that encourages objects to have as most correspondences as possible, thus achieving more stable estimated poses. This metric is relied upon in the clustering iterations within ICE, and is specially useful when coupled with our novel pose clustering algorithm. The key insight behind our pose clustering algorithm —called *Projection Clustering*— is that our object hypotheses have been detected from camera data, which might be noisy, ambiguous and/or contain matching outliers. Therefore, instead of using a regular clustering technique in pose space (using e.g. Mean Shift [66] or Hough Transforms [67]), we evaluate each type of outlier and propose a solution that handles incorrect object hypotheses and effectively merges their information with those that are most likely to be correct.

C. Scalability and latency

In MOPED, we also address the issues of scalability, throughput and latency, which are vital for real-time robotics applications. ICE enables easy parallelism in the object recognition process. We also introduce an improved feature matching algorithm for large databases that balances strong per-object matching accuracy with logarithmic complexity in the number of objects. We thus improve on common per-object matching approaches (which have linear complexity in the number of objects), and per-database matching approaches (which suffer from reduced matching ability). Our GPU/CPU hybrid architecture exploits parallelism at all levels. MOPED is optimized for bandwidth and cache management and SIMD instructions. Components like feature extraction and matching have been implemented on a GPU. Furthermore, a novel scheduling scheme enables the efficient use of symmetric multiprocessing(SMP) architectures, utilizing all available cores on modern multi-core CPUs.

D. Lessons learned

MOPED is a framework designed to recognize objects as fast as possible and minimize end-to-end latency. In order to achieve these goals, we completely redesigned MOPED to maximize parallelism both at the algorithmic and architectural level: all algorithms within MOPED are parallelizable, different tasks can be executed simultaneously in the CPU and GPU units, and an optimized resource scheduler enables the utilization of all available computing for object recognition. The multiple architectural improvements in MOPED provide over 30x improvement in latency and throughput, allowing MOPED to perform in real-time robotic applications. Unfortunately, the integration of MOPED within HERB 2.0 originally resulted in MOPED consuming most of the available computing power

Fig. 9. Recognition of real-world scenes. (Left) High-complexity scene. MOPED finds 27 objects, including partially-occluded, repeated and non-planar objects. Using a database of 91 models and an image resolution of 1600×1200, MOPED processes this image in 2.1 seconds. (Middle) Medium complexity scene. MOPED processes this 640×360 image in 87 ms and finds all known objects (The undetected green soup can is not in the database). (Right) HERB 2.0 grasping object recognized and registered by MOPED. Images taken from [68].

for all other tasks, because on-board computing is not unlimited. We leveraged this problem by linking MOPED with the Behavior Engine, so as to dynamically enable or disable MOPED processing depending on the task at hand.

An additional issue that often arises in MOPED is the model building stage to add new objects to the database. The model building stage we use in MOPED, despite being mostly automatic, still requires a certain amount of human supervision. An important path to follow in the future is the use of object discovery techniques and multi-modal data to generate fully automated models for MOPED. In particular, we are working on joint camera-laser discovery of objects [68] with automated modeling, which is a necessary task for HERB 2.0 to truly achieve long-term autonomous operation.

XI. NAVIGATION

In navigating through diverse personal spaces, HERB 2.0 employs several motion planning and control strategies that decouple levels of functionality into hierarchical layers. Two different approaches to navigation mirror two types of cognitive processes involved in human navigation.

People execute many habitual motions daily, such as walking from the cupboard to the dinner table. Through repetition, they refine actions for smoothness and efficiency, while retaining the ability to handle transient obstructions like other people passing through. This type of navigation imposes minimal cognitive load because it merely replays a stored trajectory. This scenario inspires a navigator called virtual rails.

Another class of navigation action comprises non-habitual motions. This category includes routes too long, complex, or unfamiliar to have imprinted strongly in the brain. In following such routes, humans continually adapt to nearby obstacles, incurring greater cognitive load than with habitual actions. A more adaptable navigation planner, called the Model-Based Hierarchical Planner, fulfills this role.

A. Virtual Rails

A first approach to navigating HERB 2.0 has been derived from an autonomous driving project [69]–[72], where it was used to navigate a car within a road network. Note that driving on-road clearly falls into the aforementioned *structured* or *habitual* class of scenarios.

For HERB 2.0, a path network which connects points of interest within the robot's workspace was layed out manually (Fig.10). This reduces motion planning to a small scale graph search problem. Path execution is accomplished by an orbital tracking controller, which feeds back the robot pose at a rate of 100 Hz. The method calls for a global localization system (cf. Section III-B).

Effectively, this approach turns the robot into a virtually railborne system. This carries advantages and disadvantages: On the pro side, the robot becomes very predictable. It follows the prescribed paths accurately (at cm precision), and, due to the quick positional feedback, quite fast and smoothly (the robot can maneuver safely at up to 1.5 m/s). The method facilitated easy incorporation of some basic reactive abilities into the system: In case its virtual railtrack is occupied, the robot stops in front of the obstacle, or it slowly follows if the obstacle is moving.

The obvious drawback of this method is its limited capability to cope with changes in the environment. If a railtrack gets blocked permanently, the robot will not find an alternative route around the blockage. This is where genuine motion planning concepts come to bear, as will be outlined in the next section.

B. Model-Based Hierarchical Planner

The Model-Based Hierarchical Planner, or MBHP, simultaneously performs motion planning and path following to continually adapt to unstructured, partially-known, or changing environments (Fig.10). Hierarchical planners of this design trace back to motion planners deployed outdoors in rough terrain [73], [74]. MBHP splits the navigation task into three layers differentiated by scale, fidelity, and planning rate.

At the largest scale, a global planner generates an approximate navigation plan with the expectation that the plan will likely change as HERB 2.0 discovers new obstacles. As such, extensive preplanning would constitute wasted effort. Using a simple approximation of robot motions, the global planner is capable of rapidly rerouting around newly discovered obstacles, replanning only as necessary.

At the middle level, a local planner models robot motions at high fidelity, thus exclusively generating paths that are inherently followable by the robot. Like a car, HERB 2.0's Segway base cannot move sideways. The local planner samples a variety of command trajectories; for each, the model simulates its execution on the robot to predict the resulting path trajectory. Finding the optimal set of command

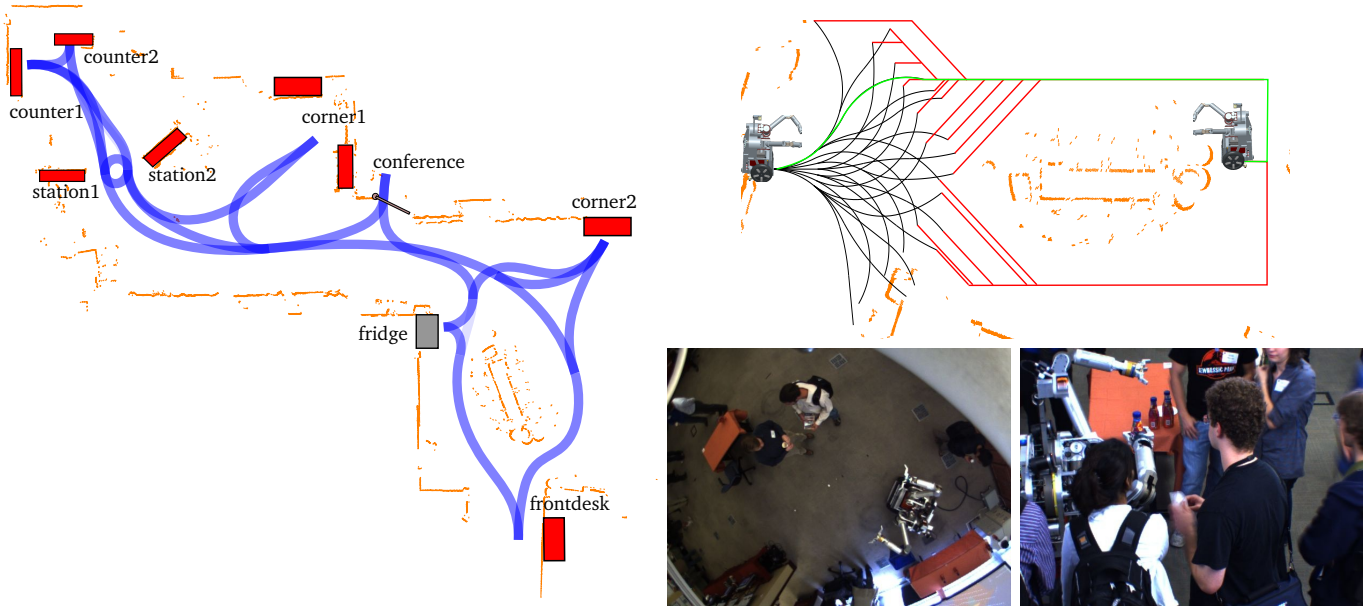

Fig. 10. (Left) Part of the virtual rail network with points of interest, superimposed on an Lidar based obstacle map. (Right-Top) Model-Based Hierarchical Planner (MBHP). The robot at left must navigate into place at right to assist in a home care task. It considers many possible initial motions in detail, (some sample curves shown in black). The remainder of each path (in red) is approximated. The robot begins to follow the chosen path (green) while replanning. Thus, a series of smooth path segments make up a complete route while reacting to changes in the environment. (Right-Bottom) HERB 2.0 navigating with Rails at the Intel Open House.

trajectories to test remains an active area of research [75]–[77]. In recognition of the computational expense of the robot model, the local planner subjects itself to a limited planning horizon. Each local path considered by the robot looks ahead 5 m or less and is concatenated with a global path to form a complete path to the goal. In order to incorporate changes in the environment, the local planner replans at a rate of 5 Hz.

Finally, a low-level controller accepts path commands from the local planner and executes them on HERB 2.0. The controller's purpose is to ensure safety. Running at 10 Hz, the scan rate of the ground-level Hokuyo laser scanner, the controller commands HERB 2.0's Segway base to follow the chosen command trajectory—unless the laser scanner detects an impending collision. This situation arises rarely since the local planner selects only collision-free paths, but the controller can respond more rapidly to fast-moving people.

C. Lessons learned

Although MBHP is the more general navigator, the contrast between the two approaches in public spaces crowded with people is startling. In following a virtual rail, a blocked robot must wait for people to move out of the way. Coupled with some audio feedback towards its environment (the robot honks if something blocks its path), the virtual rails system proves surprisingly capable. When honked at, most people adjust themselves quickly and naturally to the robot. Thus, virtual rails exploits a kind of human social structure in which people clear a path to allow another person to pass through.

MBHP surpasses virtual rails at the task of navigating among static clutter, but the current implementation performs poorly in environments crowded with people since the planner does not reason about people as intelligent obstacles who can get out of the way. It instead searches for a completely free path; if one is not found, HERB 2.0 will sit and wait (or worse, waffle between various motions). Lacking the predictability of singular intent possessed by virtual rails, MBHP does not communicate clearly to people where HERB 2.0 is trying to go. The contrast of these two navigators in a crowd highlights the importance of continued work to enhance robotic capabilities in the detection, recognition, and prediction of human behavior in response to robot actions.

XII. CONCLUSION

We have presented a snapshot of two years of effort on HERB 2.0 2.0. The platform is evolving, and will forever be evolving. Key to our progress has been a commitment to working in a home environment and understanding the nuances of how humans structure their environments. Understanding this structure has enabled more robust, efficient, and predictable behavior. Some of our observations have surprised us and they point towards much deeper research questions, on understanding human intent, on collaborative manipulation, and on addressing extreme clutter with physics-based manipulation planners. We are now well positioned to move towards new unsolved problem domains.

a) Reconciling geometric planning with physics-based manipulation: Humans have instinct. Robots have search. We are able to pick up knives and forks, stack plates, move objects with our arms, balance dishes, kick open kitchen doors, and load dishwashers. HERB 2.0 has barely scratched the surface of what he can do with his arms and base. Our work on push-grasping clutter is a start towards merging physics-based manipulation with geometric search but we are excited to go beyond that. There are two immediate questions to answer:(1) how can we incorporate sensor feedback into our strategies,

and (2) how can we automatically learn strategies from observation or demonstration? Answering the first question will enable robust execution of existing strategies. The second question is much harder to answer but is critical. There are countless strategies to learn but robots can use their prior knowledge and also learn from their own and other robots' experience.

b) Collaborative Manipulation: We envision HERB 2.0 performing complex manipulation tasks *with* humans: HERB 2.0 should be able to prepare a meal and clear a table with a person, or to build an IKEA shelf with them. Human-robot interaction must go beyond dialog management, to physical collaboration. Currently, HERB 2.0 is on its way to performing each of these tasks autonomously. But, strangely enough, doing these tasks *with a human* will be much harder: HERB 2.0 might need to sense human kinematics, human intent, understand turn-taking, and react to the environment *and* humans in real-time. So, do we really need collaboration? There are definitely scenarios in which collaboration is critical: in the battlefield for assembling a mortar or carrying an injured soldier, or in the home assisting a patient with disabilities with their activities of daily living. But even in these cases, the role of a robot and the balance of autonomy and collaboration will change dynamically. We are excited to explore this balance, and the challenge of humans and robots performing tightly-coupled manipulation tasks collaboratively.

c) Sensing and Actuation for Robotics: Do we really need a $\$500,000$ robot like HERB 2.0 to enable lifelong mobile manipulation? We have been exploring the capabilities of simple hands. Surely, simple hands can do a lot less than more expensive complex hands, but the details of their limits are important. Our results to date have been surprising: we have demonstrated that by shifting the complexity from the mechanisms to the computational algorithms, complex tasks like bin-picking and singulation can be achieved with simple hands [78]–[80].

With robots like HERB 2.0, we now have the ability to excavate these questions and are perfectly positioned to identify, prove, develop, and demonstrate the principles of mobile manipulation that will enable our robots to interact with us in our environment and impact our lives in meaningful ways.

ACKNOWLEDGMENT

This material is based upon work supported by DARPA-BAA-10-28, NSF-IIS-0916557, and NSF-EEC-0540865. Dmitry Berenson was partially supported by the Intel PhD Fellowship, Alvaro Collet by the Caja Madrid Fellowship, Maya Cakmak and Tim Niemueller by the Intel Summer Fellowship, Julius Ziegler by the Karlsruhe House of Young Scientists, and Mehmet Dogar by the Fulbright Fellowship.

REFERENCES

[1] S. Srinivasa, D. Ferguson, J. Vande Weghe, D. Berenson, R. Diankov, C. Helfrich, and H. Strasdat, "The Robotic Busboy: Steps Towards Developing a Mobile Robotic Home Assistant," in *IEEE International Conference on Intelligent Autonomous Systems*, July 2008.

[2] S. Srinivasa, D. Ferguson, C. Helfrich, D. Berenson, A. Collet, R. Diankov, G. Gallagher, G. Hollinger, J. Kuffner, and J. Vande Weghe, "HERB: a Home Exploring Robotic Butler," *Autonomous Robots*, vol. 28, no. 1, pp. 5–20, 2009.

[3] N. Nilsson, "A mobile automation: An application of artificial intelligence techniques," in *Proceedings of the 1st Int. Joint Conference on Artificial Intelligence*, 1969, pp. 509–520.

[4] A. P. Ambler, H. G. Barrow, C. M. Brown, R. M. Burstall, and R. J. Popplestone, "A versatile computer-controlled assembly system," in *Proceedings of the 3rd Int. Joint Conference on Artificial Intelligence*, 1973, pp. 298–307.

[5] T. Lozano-Perez, J. Jones, E. Mazer, P. O'Donnell, W. Grimson, P. Tournassoud, and A. Lanusse, "Handey: A robot system that recognizes, plans, and manipulates," in *Robotics and Automation. Proceedings. 1987 IEEE International Conference on*, vol. 4. IEEE, 1987, pp. 843–849.

[6] T. Lozano-Perez, J. Jones, E. Mazer, and P. O'Donnell, "Handey: a robot task planner," 1992.

[7] A. M. Thompson, "The navigation system of the JPL robot," in *Proceedings of the 5th Int. Joint Conference on Artificial Intelligence*, 1977, pp. 749–757.

[8] O. Khatib, K. Yokoi, K. Chang, D. Ruspini, R. Holmberg, and A. Casal, "Coordination and decentralized cooperation of multiple mobile manipulators," *Journal of Robotic Systems*, vol. 13, no. 11, pp. 755–764, 1996.

[9] R. Alami, L. Aguilar, H. Bullata, S. Fleury, M. Herrb, F. Ingrand, M. Khatib, and F. Robert, "A general framework for multi-robot cooperation and its implementation on a set of three hilare robots," *Experimental Robotics IV*, pp. 26–39, 1997.

[10] Y. Sakagami, R. Watanabe, C. Aoyama, S. Matsunaga, N. Higaki, and K. Fujimura, "The intelligent asimo: System overview and integration," in *Intelligent Robots and Systems, 2002. IEEE/RSJ International Conference on*, vol. 3. Ieee, 2002, pp. 2478–2483.

[11] K. Nishiwaki, T. Sugihara, S. Kagami, F. Kanehiro, M. Inaba, and H. Inoue, "Design and development of research platform for perception-action integration in humanoid robot: H6," in *Intelligent Robots and Systems, 2000.(IROS 2000). Proceedings. 2000 IEEE/RSJ International Conference on*, vol. 3. IEEE, 2000, pp. 1559–1564.

[12] S. Kagami, K. Nishiwaki, J. Kuffner Jr, Y. Kuniyoshi, M. Inaba, and H. Inoue, "Online 3d vision, motion planning and bipedal locomotion control coupling system of humanoid robot: H7," in *Intelligent Robots and Systems, 2002. IEEE/RSJ International Conference on*, vol. 3. IEEE, 2002, pp. 2557–2562.

[13] K. Kaneko, F. Kanehiro, S. Kajita, H. Hirukawa, T. Kawasaki, M. Hirata, K. Akachi, and T. Isozumi, "Humanoid robot hrp-2," in *Robotics and Automation, 2004. Proceedings. ICRA'04. 2004 IEEE International Conference on*, vol. 2. IEEE, 2004, pp. 1083–1090.

[14] K. Akachi, K. Kaneko, N. Kanehira, S. Ota, G. Miyamori, M. Hirata, S. Kajita, and F. Kanehiro, "Development of humanoid robot hrp-3p," in *Humanoid Robots, 2005 5th IEEE-RAS International Conference on*. IEEE, 2005, pp. 50–55.

[15] K. Kaneko, F. Kanehiro, M. Morisawa, K. Miura, S. Nakaoka, and S. Kajita, "Cybernetic human hrp-4c," in *Humanoid Robots, 2009. Humanoids 2009. 9th IEEE-RAS International Conference on*. IEEE, 2009, pp. 7–14.

[16] M. Ciocarlie, K. Hsiao, E. Jones, S. Chitta, R. Rusu, and I. Sucan, "Towards reliable grasping and manipulation in household environments," in *Proceedings of RSS 2010 Workshop on Strategies and Evaluation for Mobile Manipulation in Household Environments*, 2010.

[17] A. Albu-Sch
"affer, S. Haddadin, C. Ott, A. Stemmer, T. Wimb
"ock, and G. Hirzinger, "The dlr lightweight robot: design and control concepts for robots in human environments," *Industrial Robot: An International Journal*, vol. 34, no. 5, pp. 376–385, 2007.

[18] M. Beetz, L. Mošenlechner, and M. Tenorth, "Crama cognitive robot abstract machine for everyday manipulation in human environments," in *Intelligent Robots and Systems (IROS), 2010 IEEE/RSJ International Conference on*. IEEE, 2010, pp. 1012–1017.

[19] A. Jain and C. Kemp, "El-e: an assistive mobile manipulator that autonomously fetches objects from flat surfaces," *Autonomous Robots*, vol. 28, no. 1, pp. 45–64, 2010.

[20] M. Quigley, E. Berger, and A. Ng, "Stair: Hardware and software architecture," in *AAAI 2007 Robotics Workshop, Vancouver, BC*, 2007.

[21] M. Quigley, K. Conley, B. P. Gerkey, J. Faust, T. Foote, J. Leibs, R. Wheeler, and A. Y. Ng, "ROS: an open-source Robot Operating System," in *ICRA Workshop on Open Source Software*, 2009.

[22] K. Konolige, "COLBERT: A Language for Reactive Control in Sapphira," in *Proceedings KI-97: Advances in Artificial Intelligence, 21st*

Annual German Conference on Artificial Intelligence, Freiburg, Germany, September 9-12, 1997, 1997, pp. 31–52.

[23] M. Loetzsch, M. Risler, and M. Jungel, "XABSL - A Pragmatic Approach to Behavior Engineering," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2006, pp. 5124–5129.

[24] T. Niemueller, A. Ferrein, and G. Lakemeyer, "A Lua-based Behavior Engine for Controlling the Humanoid Robot Nao," in *Proc. of RoboCup XIII Symposium*. Graz, Austria: Springer, 2009.

[25] T. Niemueller, A. Ferrein, D. Beck, and G. Lakemeyer, "Design Principles of the Component-Based Robot Software Framework Fawkes," in *Second International Conference on Simulation, Modeling, and Programming for Autonomous Robots (SIMPAR)*, 2010.

[26] R. Ierusalimschy, *Programming in Lua*, 2nd ed. Lua.org, 2006.

[27] J. Bohren and S. Cousins, "The SMACH High-Level Executive," *Robotics Automation Magazine, IEEE*, vol. 17, no. 4, pp. 18–20, 2010.

[28] D. Berenson, S. Srinivasa, and J. Kuffner, "Task Space Regions: A framework for pose-constrained manipulation planning," *The International Journal of Robotics Research*, March 2011.

[29] D. Berenson, T. Simeon, and S. Srinivasa, "Addressing cost-space chasms in manipulation planning," in *IEEE International Conference on Robotics and Automation (ICRA)*, May 2011.

[30] M. Stilman, "Task constrained motion planning in robot joint space," in *Proc. IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2007.

[31] Y. Koga, K. Kondo, J. Kuffner, and J. claude Latombe, "Planning motions with intentions," in *SIGGRAPH*, 1994.

[32] K. Yamane, J. Kuffner, and J. Hodgins, "Synthesizing animations of human manipulation tasks," in *SIGGRAPH*, 2004.

[33] Z. Yao and K. Gupta, "Path planning with general end-effector constraints: using task space to guide configuration space search," in *Proc. IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2005.

[34] E. Drumwright and V. Ng-Thow-Hing, "Toward interactive reaching in static environments for humanoid robots," in *Proc. IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2006.

[35] D. Bertram, J. Kuffner, R. Dillmann, and T. Asfour, "An integrated approach to inverse kinematics and path planning for redundant manipulators," in *Proc. IEEE International Conference on Robotics and Automation (ICRA)*, 2006.

[36] D. Berenson, J. Chestnutt, S. S. Srinivasa, J. J. Kuffner, and S. Kagami, "Pose-Constrained Whole-Body Planning using Task Space Region Chains," in *Proc. IEEE-RAS International Conference on Humanoid Robots*, 2009.

[37] D. Berenson, S. Srinivasa, and J. Kuffner, "Addressing Pose Uncertainty in Manipulation Planning Using Task Space Regions," in *Proc. IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2009.

[38] D. Berenson, S. S. Srinivasa, D. Ferguson, and J. Kuffner, "Manipulation planning on constraint manifolds," in *Proc. IEEE International Conference on Robotics and Automation (ICRA)*, 2009.

[39] J. H. Yakey, S. M. LaValle, and L. E. Kavraki, "Randomized path planning for linkages with closed kinematic chains," *IEEE Transactions on Robotics and Automation*, vol. 17, no. 6, pp. 951–958, 2001.

[40] J. Cortes and T. Simeon, "Sampling-based motion planning under kinematic loop-closure constraints," in *Proc. Workshop on the Algorithmic Foundations of Robotics (WAFR)*, 2004.

[41] D. Berenson and S. Srinivasa, "Probabilistically complete planning with end-effector pose constraints," in *Proc. IEEE International Conference on Robotics and Automation (ICRA)*, May 2010.

[42] S. Dalibard, A. Nakhaei, F. Lamiraux, and J.-p. Laumond, "Whole-Body Task Planning for a Humanoid Robot: A Way to Integrate Collision Avoidance," in *Humanoids*, 2009.

[43] D. Berenson, "Constrained manipulation planning," Ph.D. dissertation, Robotics Institute, Carnegie Mellon University, Pittsburgh, PA, May 2011.

[44] M. Dogar and S. Srinivasa, "Push-grasping with dexterous hands," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, October 2010.

[45] ——, "A framework for push-grasping in clutter," in *Robotics: Science and Systems VII*, 2011.

[46] T. Lozano-Perez, M. T. Mason, and R. H. Taylor, "Automatic synthesis of fine-motion strategies for robots," *IJRR*, vol. 3, no. 1, 1984.

[47] S. Goyal, A. Ruina, and J. Papadopoulos, "Planar sliding with dry friction. Part 1. Limit surface and moment function." *Wear*, no. 143, pp. 307–330, 1991.

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

[49] A. D. Dragan, N. D. Ratliff, and S. S. Srinivasa, "Manipulation planning with goal sets using constrained trajectory optimization," in *IEEE International Conference on Robotics and Automation*. IEEE, 2011.

[50] A. D. Dragan, G. J. Gordon, and S. S. Srinivasa, "Learning from experience in manipulation planning: Setting the right goals," in *International Symposium on Robotics Research (ISRR)*, 2011.

[51] N. D. Ratliff, J. A. Bagnell, and M. A. Zinkevich, "Maximum margin planning," in *In Proceedings of the 23rd International Conference on Machine Learning (ICML06*, 2006.

[52] M. Cakmak, S. Srinivasa, M. Lee, S. Kiesler, and J. Forlizzi, "Using spatial and temporal contrast for fluent robot-human hand-overs," in *Proceedings of the 6th International Conference on Human-Robot Interaction (HRI)*, 2011, pp. 489–497.

[53] M. Cakmak, S. Srinivasa, M. Lee, J. Forlizzi, and S. Kiesler, "Human preferences for robot-human hand-over configurations," in *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and System (IROS)*, 2011.

[54] A. Agah and K. Tanie, "Human interaction with a service robot: Mobile-manipulator handing over an object to a human," in *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, 1997, pp. 575–580.

[55] M. Huber, M. Rickert, A. Knoll, T. Brandt, and S. Glasauer, "Human-robot interaction in handing-over tasks," in *Proceedings of the IEEE International Symposium on Robot and Human Interactive Communication (RO-MAN)*, 2008, pp. 107–112.

[56] E. Sisbot, L. Marin, and R. Alami, "Spatial reasoning for human robot interaction," in *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and System (IROS)*, 2007.

[57] S. Kajikawa, T. Okino, K. Ohba, and H. Inooka, "Motion planning for hand-over between human and robot," *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 193–199, 1995.

[58] E. Lopez-Damian, D. Sidobre, S. DeLaTour, and R. Alami, "Grasp planning for interactive object manipulation," in *Proceedings of the International Symposium on Robotics and Automation*, 2006.

[59] A. Edsinger and C. Kemp, "Human-robot interaction for cooperative manipulation: Handing objects to one another," in *Proceedings of the IEEE International Symposium on Robot and Human Interactive Communication (RO-MAN)*, 2007.

[60] K. Koay, E. Sisbot, D. Syrdal, M. Walters, K. Dautenhahn, and R. Alami, "Exploratory study of a robot approaching a person in the context of handing over an object," in *Proceedings of the AAAI Spring Symposium on Multidisciplinary Collaboration for Socially Assistive Robotics*, 2007, pp. 18–24.

[61] Y. Choi, T. Chen, A. Jain, C. Anderson, J. Glass, and C. Kemp, "Hand it over or set it down: A user study of object delivery with an assistive mobile manipulator," in *Proceedings of the IEEE International Symposium on Robot and Human Interactive Communication (RO-MAN)*, 2009.

[62] K. Nagata, Y. Oosaki, M. Kakikura, and H. Tsukune, "Delivery by hand between human and robot based on fingertip force-torque information," in *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and System (IROS)*, 1998, pp. 750–757.

[63] I. Kim and H. Inooka, "Hand-over of an object between human and robot," in *Proceedings of the IEEE International Symposium on Robot and Human Interactive Communication (RO-MAN)*, 1992.

[64] A. Collet, M. Martinez, and S. S. Srinivasa, "The moped framework: Object recognition and pose estimation for manipulation," *International Journal of Robotics Research*, vol. 30, no. 10, pp. 1284–1306, 2011.

[65] D. Lowe, "Distinctive Image Features from Scale-Invariant Keypoints," *International Journal of Computer Vision*, vol. 60, no. 2, pp. 91–110, 2004. [Online]. Available: http://portal.acm.org/citation.cfm?id=996342

[66] Y. Cheng, "Mean shift, mode seeking, and clustering," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 17, no. 8, p. 790, 1995.

[67] C. F. Olson, "Efficient pose clustering using a randomized algorithm," *International Journal of Computer Vision*, vol. 23, no. 2, p. 131, 1997.

[68] A. Collet, S. S. Srinivasa, and M. Hebert, "Structure discovery in multi-modal data : a region-based approach," in *IEEE International Conference on Robotics and Automation*, May 2011.

[69] S. Kammel, J. Ziegler, B. Pitzer, M. Werling, T. Gindele, D. Jagzent, J. Schröder, M. Thuy, M. Goebl, F. v. Hundelshausen, O. Pink, C. Frese, and C. Stiller, "Team AnnieWAY's autonomous system for the 2007

DARPA urban challenge," *International Journal of Field Robotics Research*, vol. 25, pp. 615–639, 2008.

[70] J. Ziegler, M. Werling, and J. Schröder, "Navigating car-like vehicles in unstructured environment," in *Proceedings of the IEEE Intelligent Vehicles Symposium*, 2008, pp. 787–791.

[71] M. Werling, J. Ziegler, S. Kammel, and S. Thrun, "Optimal trajectory generation for dynamic street scenarios in a frenet frame," in *IEEE International Conference on Robotics and Automation*, 2010, pp. 987–993.

[72] T. Gindele, D. Jagzent, B. Pitzer, and R. Dillmann, "Design of the planner of Team AnnieWAY's autonomous vehicle used in the DARPA urban challenge 2007," in *Proceedings of the IEEE Intelligent Vehicles Symposium*, Eindhoven, Die Niederlande, 2008.

[73] A. Kelly, A. Stentz, O. Amidi, M. W. Bode, D. Bradley, A. Diaz-Calderon, M. Happold, H. Herman, R. Mandelbaum, T. Pilarski, P. Rander, S. Thayer, N. M. Vallidis, and R. Warner, "Toward reliable off road autonomous vehicles operating in challenging environments," *International Journal of Robotics Research*, vol. 25, no. 1, pp. 449–483, May 2006.

[74] T. Allen, J. Underwood, and S. Scheding, "A path planning system for autonomous ground vehicles operating in unstructured dynamic environments," in *Proceedings of the Australasian Conference on Robotics and Automation*, 2007.

[75] C. Green and A. Kelly, "Toward optimal sampling in the space of paths," in *International Symposium on Robotics Research*, Hiroshima, Japan, November 2007.

[76] L. Erickson and S. LaValle, "Survivability: Measuring and ensuring path diversity," in *IEEE International Conference on Robotics and Automation*, Kobe, Japan, May 2009.

[77] R. A. Knepper and M. T. Mason, "Path diversity is only part of the problem," in *IEEE International Conference on Robotics and Automation*, May 2009.

[78] M. Mason, S. S. Srinivasa, and A. Vazquez, "Generality and simple hands," in *International Symposium on Robotics Research*, July 2009.

[79] A. R. Garcia, M. Mason, and S. S. Srinivasa, "Manipulation capabilities with simple hands," in *International Symposium on Experimental Robotics*, December 2010.

[80] M. Mason, A. R. Garcia, S. S. Srinivasa, and A. Vazquez, "Autonomous manipulation with a general-purpose simple hand," *International Journal of Robotics Research*, 2011.