Auton Robot (2010) 28: 131–149
DOI 10.1007/s10514-009-9142-y

# An autonomous mobile manipulator for assembly tasks

**Brad Hamner · Seth Koterba · Jane Shi · Reid Simmons · Sanjiv Singh**

Received: 22 January 2009 / Accepted: 13 August 2009 / Published online: 1 September 2009


**Abstract** The fundamental difference between autonomous robotic assembly and traditional hard automation, currently utilized in large-scale manufacturing production, lies in the specific approaches used in locating, acquiring, manipulating, aligning, and assembling parts. An autonomous robotic assembly manipulator offers high flexibility and high capability to deal with the inherent system uncertainties, unknowns, and exceptions. This paper presents an autonomous mobile manipulator that effectively overcomes inherent system uncertainties and exceptions by utilizing control strategies that *employ coordinated control, combine visual and force servoing*, and *incorporate sophisticated reactive task control*. The mobile manipulation system has been demonstrated experimentally to achieve high reliability for a "peg-in-hole" type of insertion assembly task that is commonly encountered in automotive wiring harness assembly.

**Keywords** Mobile manipulation · Coordinated control · Constrained motion · Task architecture

B. Hamner (✉) · S. Koterba · R. Simmons · S. Singh
Carnegie Mellon University, 5000 Forbes Ave., Pittsburgh, PA 15213, USA
e-mail: bhamner@andrew.cmu.edu

S. Koterba
e-mail: skoterba+@cs.cmu.edu

R. Simmons
e-mail: reids@cs.cmu.edu

S. Singh
e-mail: ssingh@cs.cmu.edu

J. Shi
Research and Development Center, Manufacturing Systems Research Lab., General Motors Company, 30500 Mound Road, Warren, MI 48090, USA
e-mail: jane.shi@gm.com

## 1 Introduction

Across many domains, there is increasing demand for robots capable of performing complex and dexterous manipulation tasks. A prominent example is the need for automated assembly. From terrestrial applications, such as factory assembly lines, to extraterrestrial scenarios, such as automated assembly of lunar habitats, robotic manipulation is, and will be, integral. While robotic assets currently perform many tasks, especially in factory settings, these tasks are typically highly structured. In other words, the tasks and robots are designed such that minimally sophisticated technology can achieve success. Often, hardware and software are designed for a very specific task. Looking ahead, future robots may assume a variety of duties, from transporting payloads across large workspaces to precise insertion or placement of small components. The tasks will be highly variable and complex. With the combination of a dexterous manipulator and a mobile base, mobile manipulators are well suited to these complex assembly tasks.

Although mobile manipulators provide the mobility and dexterity to meet the challenges of assembly tasks, sensors and software play a critical role in achieving successful results. Given any complex scenario, the mobile manipulator must be able to sense its environment, make decisions based on the configuration of the environment, and correctly perform actions informed by those decisions.

In this paper we present work on an advanced mobile manipulator capable of performing assembly tasks in the domain of automotive manufacturing. First, we describe our controller, which coordinates the motion of the mobile base and manipulator to achieve desired end-effector poses while maintaining high manipulability. Then, we present competing control schemes that use differing combinations of visual servoing and force control, and compare the schemes

in their abilities to perform a tight-tolerance "peg-in-hole" type of insertion assembly task. We define a framework for constraining the end-effector of a manipulator based on position and/or force using a high-frequency, low-level controller, and combine this with a sophisticated task-level executive. Finally, we demonstrate the capability of the system with results from repeated experiments of the insertion task on an automotive task board. The results show that we can achieve these tight-tolerance assembly tasks with high reliability and robustness to error with the robot starting from varying, unknown positions.

## 2 Overview of approach and related work

Our research goal is to perform sophisticated assembly tasks autonomously. To achieve this, we have developed an autonomous mobile manipulator that effectively overcomes inherent system uncertainties and exceptions, with a high degree of performance and reliability, by utilizing control strategies that employ coordinated control of the base and manipulator, combine visual and force servoing, and incorporate sophisticated reactive task-level control. Coordination of the mobile base and manipulator increases the range of the manipulator. The combination of visual and force servoing increases the performance of the system beyond that of a system which uses either method alone. Task-level control provides an architecture for flexible and reliable task execution. In the following subsections, we discuss these three aspects of our approach that enable high-performance and reliable mobile manipulation, as well as related approaches.

### 2.1 Coordinated control of base and manipulator

Task flexibility and robotic mobility are two main advantages that mobile manipulators can bring to manufacturing assembly applications. Compared with traditional industrial robots, it is easier for mobile manipulators to adapt to changing environments and perform a variety of assembly tasks. By placing the manipulator on a mobile base, the robot can travel between the work area and parts feeder and can follow moving assembly lines.

Early work in mobile manipulation often treated the mobile base and the manipulator as two independent robots. Shin et al. demonstrated the limitations of this approach and the need for coordination. They developed a system that employs decoupled motion of the mobile manipulator (Shin et al. 2003). Decoupled motion describes motion in which the base remains fixed while the manipulator moves. This allows for high tracking accuracy of the end effector, but takes longer for tasks to be accomplished as tracking must be interrupted to allow for repositioning of the base. The planner used chooses base positions to optimize the arm's manipulability in the direction of the desired trajectory while minimizing the number of times the base is required to reposition (Shin et al. 2003).

In the context of coordination of mobile bases and manipulators, many authors have contributed to the literature. The majority of contributions perform coordination in terms of dynamics, i.e. controlling forces and torques rather than positions and velocities, a technique that differs from our kinematic approach. Some of these authors have investigated internal control in this dynamic context. For example, Yamamoto and Yun described a dynamic coordinated controller in which the end-effector is constrained to follow a trajectory while the base is controlled to maintain arm manipulability when possible (Yamamoto and Yun 1992). Similarly, Holmberg and Khatib defined a dynamically decoupled model of control to achieve smooth, precise control of a mobile base (Holmberg and Khatib 2000). This work seeks to minimize forces on a mobile base, called a powered caster vehicle, while producing and maintaining desired coordinated motion. Kim et al. present a coordinated control scheme that uses the null space of an overactuated mobile manipulator to achieve dynamically stable motion (Kim et al. 2002).

Although it is not explicitly dealt with in our current experimental scenario, avoiding obstacles is an important capability of mobile manipulators in manufacturing and assembly. A manipulator may have to reach around, or through, already-assembled components to achieve its task. Also, in addition to extending the range of the manipulator, the addition of a mobile base adds kinematic redundancy, which allows the robot to avoid obstacles while completing tasks efficiently. Brock and Khatib defined the elastic strips framework in which a dynamic planner allows the mobile manipulator to move through a tunnel of free space (Brock and Khatib 2002). The controller tracks a desired end-effector path while keeping the manipulator's joints and the base away from obstacles. This is achieved by generating protective hulls, which are constructed to ensure that neither the base nor the arm will collide with any obstacles during trajectory following. Without using a planner, a controller like the one presented in Sciavicco and Siciliano (2005) can use the null space of the manipulator's Jacobian to keep the joints away from obstacles. Our controller uses this control scheme, with the exception that we use the null space to achieve a higher manipulability (see Sect. 4).

### 2.2 Combining visual and force servoing

In the domain of robotic assembly, the critical portion of a control task is the physical mating of components. We use visual servoing and force servoing together in a manner that improves the system robustness, without the need for highly

accurate visual sensing or precisely known global reference coordinates. The visual servoing is used to bring the end-effector of the mobile manipulator to a rough alignment in a gross area of the assembly location. Hybrid control of pose and force (as detailed in Sect. 5) enables the mobile manipulator to overcome the geometric location uncertainty from visual servoing to contact the task board and complete the peg-in-hole assembly at a precise location, with tight tolerance.

Mason (1979) laid the foundation for the earliest pose/force control work by defining the constraint frame based on natural constraints (rigid objects in the space) and artificial constraints (desired positions or forces on the end-effector). Raibert and Craig (1981) defined a hybrid pose/force control scheme based on these descriptions. They run separate, decoupled pose and force controllers. Each dimension of the constraint frame is controlled by at most one of the controllers, not both. By decoupling the problem in this way, they ensure that the force constraints and pose constraints do not work at odds with one another. The results of the two controllers are summed and then converted into commands for the manipulator's joints via inverse kinematics. Yoshikawa (1987) built on this formulation, but extended it to incorporate the dynamics of the manipulator. He went on to show that if the dynamic models are correct, it can be proven that both desired pose and force can be achieved. Mills and Goldenberg (1989) also take a dynamics approach. However, they linearize the dynamics equations locally to ease computation. Their method requires an explicit definition of the constraint surface, but the controller operates in the task space of the end-effector, rather than requiring a separate constraint space, as in the earlier methods.

In more recent work, the domain of assistive robotics provides many of the same challenges as in mobile assembly. A robot working with humans may be tasked to search for and track objects whose locations are not known *a priori*, and may have to pick up, carry, and place objects (Schaal 2007 presents an extensive summary of the challenges of robots operating in human environments). Similarly, a robot operating on a mobile assembly line must be able to acquire a part, locate, track, and approach the piece undergoing assembly, and perform the assembly operation. Service robots also have the added difficulty of needing to appear friendly and operate smoothly in environments designed for humans without consideration for robots. Industrial robots have no such restrictions, but the precision of the pickup and place tasks is higher. Depending on the part being assembled, allowable position error may be measured in millimeters, and the orientation may have to be nearly perfect (to plug in an electronic component, for example).

In service and industrial robotics, the requirements of working in a large area, yet being able to grasp objects, lends itself to the use of mobile manipulators with many-degree-of-freedom arms. Bischoff developed the humanoid robot HERMES as a flexible, mobile, and dependable platform for assistive robotics (Bischoff 1997). Petersson et al. describe a system in which a mobile manipulator must detect, approach, and pick up an item from a previously-unknown location. An emphasis is placed on integrating sensing, planning, and control to achieve the task at hand (Petersson et al. 2002). Dillman et al. also study pickup and placement with mobile manipulators, on command from humans, doing so in human-friendly environments (Dillman et al. 2002).

Force feedback and force control have been used in both service and industrial robotics. Peshkin et al. developed the *Cobot*, a collaborative robot for assisting human operators with heavy lifting and control of heavy objects in manufacturing and assembly tasks. They define a virtual surface along which to move an assembly. As the operator pushes the assembly, the machine prevents its motion past the surface, instead applying resistive force normal to the surface and allowing it to slide along the surface, as if the assembly were on a smooth rail in the direction of the intended motion (Peshkin et al. 2000). Duchaine and Gosselin use impedance control to let a human control a manipulator, where the force being applied to the robot by an operator is read as intention for the end-effector to move in a particular direction (Duchaine and Gosselin 2007). They implement a velocity controller to achieve the smooth motion necessary when working with a human in the loop.

Force feedback may be used to sense objects where other sensors cannot be as precise. Kragic et al. also working in the domain of service robotics, use tactile sensing to account for visual servo inaccuracy (Kragic et al. 2003). During a grasp task, a visual servo moves a gripper hand close to the object to be grasped. When the hand makes contact with the object, the robot re-centers the hand about the point of contact. This is similar to our scheme, presented below, in which a visual servo alignment moves the gripper close enough to begin using force feedback to feel for a target. Edsinger and Kemp use force sensing in multiple ways (Edsinger and Kemp 2006). One way is similar to that of Kragic, in that force feedback provides assurance that the object to be grasped is indeed being grasped, as well as some information on the diameter of the object. Force information is also used to augment possibly inaccurate visual information. For instance, when placing an object, the robot feels to get a precise location for a shelf whose initial position comes from fiducials detected by a camera. As presented below, we use a similar scheme to find the task board and target hole in our assembly task after rough alignment using fiducials.

Our implementation of the assembly task requires added, known structure to the environment (ARTag fiducials) to aid in localization. However, this may not be possible in all situations. To extend our work to more scenarios, we expect

to use a localization algorithm that makes use of the existing structure in the environment. For instance, Se et al. have proposed a global localization method that uses SIFT features to build local maps, which are then merged and integrated with a global map (Se et al. 2005).

### 2.3 Reactive task control

In order to be deployed in manufacturing and assembly environments, mobile manipulators must have extremely high performance and be able to deal with system uncertainty and errors autonomously. Reliability in task execution is critical, since the environment is rarely exactly the same between runs, and errors are frequent. To achieve complex assembly operations reliably, sophisticated, reactive task-level control is required. The reactive task-level control framework decomposes high-level tasks into subtasks. An executive then sequences the subtasks, dispatches them, and monitors their execution. The executive can react to contingencies, retrying subtasks or removing them from the queue and attempting a different approach, as it deems necessary. Our approach uses the executive developed for the Syndicate architecture (Sellner et al. 2006), which extends the 3T architecture (Bonasso et al. 1997) in that each robotic agent runs an executive that detects errors and exceptions and modifies an intended schedule of tasks to ensure completion of the main goal.

Effective error detection and recovery is one of several techniques that can increase task execution reliability and robustness. Donald (1990) studied the planning aspect of error detection and recovery methods for robotic peg-in-hole assembly. He uses a geometry model to enumerate a variety of assembly failure conditions of a given peg-in-hole assembly task for presumed system uncertainties, and then plans either a single or multi-step motion strategy to deal with those assembly failures in the task execution. The method is based on a "push-forwards" algorithm and "failure mode analysis" to plan a motion region for a trial-and-error move to improve the assembly robustness. The strength of the technique is the comprehensive analysis of system uncertainties and their synthesis. The weakness is that it is extremely hard to program a general solution of recovery for all possible failures that have been anticipated at the design stage.

Recently, De Schutter et al. (2007) generalized a task specification approach, called constraint-based programming, to specify sensor-based manipulation tasks by assigning control modes and associated constraints to arbitrary directions in the six dimensional manipulation space. As described in Sect. 5, we have adopted a similar approach with constrained motion primitives and their associated constraints. The main concept is to specify a general error recovery for an assembly task by utilizing a set of constraints. With the knowledge of a single constraint violation, or violations of various constraint combinations, the user can program an error recovery decision based on experimental results and observations. Thus, reactive task control can be realized by implementing recovery procedures for different failure conditions in the executive layer, as detailed in Sect. 6.

## 3 Application and experimental system

This section details the application scenario that we are tackling and the mobile manipulator that we designed, which was used for the experiments.

### 3.1 Application scenario

Although robots are commonplace on modern automotive body assembly lines, there are many tasks that continue to be performed by humans in automotive general assembly. The scenario chosen in this project is an example of one such manual task. The task is that of securing a wiring harness to a vehicle body using "Christmas tree" style push-mount plugs. The plugs are secured to the harness with plastic tie-wraps and secured to the vehicle by forcing them into predrilled holes as shown in Fig. 1. For our experiments, we acquired the rear section of an actual vehicle (Fig. 2), which will be referred to as the task board. The goal of this scenario is to secure a plug into a specified hole on the task board starting from an arbitrary position within the vicinity of the task board. A fiducial is mounted on the task board for purposes of visual identification and servoing. The task has fairly tight tolerance, on the order of a few millimeters.

### 3.2 The mobile manipulator

To achieve this task, we developed an advanced mobile manipulator. The mobile manipulator, shown in Fig. 3, consists

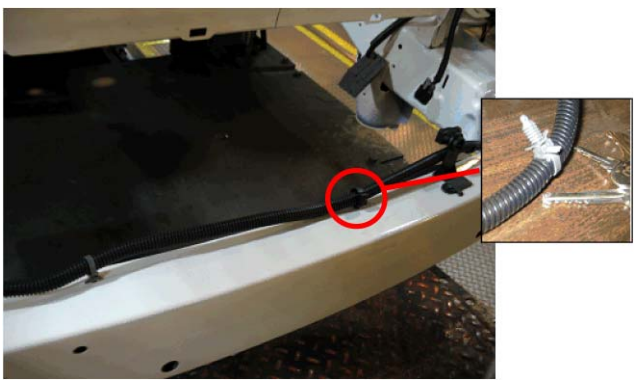

**Fig. 1** "Christmas-tree" type peg-in-hole assembly in automotive general assembly

**Fig. 2** Our task board is the back end of an actual vehicle. A fiducial is mounted near the target hole

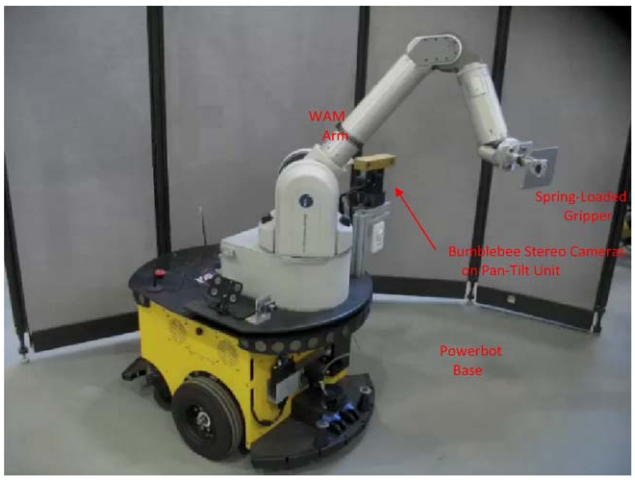


**Fig. 3** The mobile manipulator consists of a PowerBot base with a 7-DOF WAM arm

of two primary components: a dexterous manipulator and a mobile base. The manipulator is a 7-DOF WAM arm by Barrett Technologies. Its back-drivable design allows for the measurement of end effector forces based on the external torques applied at each joint. The arm has a built-in PC104 computer that runs a dedicated controller at 500 Hz. The mobile base is a Powerbot from MobileRobots.com. The arm is powered from the base's battery banks. A custom built computer is built into a compartment in the Powerbot's chassis. The arm and base communicate via an onboard hub. In addition to the force sensing capabilities of the arm, the other onboard sensor is a Point Grey Bumblebee2 stereo pair mounted on a Directed Perception pan-tilt unit (PTU). A custom spring loaded plug gripper (Fig. 4) was designed to hold a plug and release it when the forces exceed a threshold.

## 4 Coordinated control

This section highlights our work on coordination of a dexterous manipulator and mobile base to achieve flexible 6-DOF end effector placement.

**Fig. 4** A spring-loaded gripper holds a plug, ready for insertion into the target hole on the task board

### 4.1 Resolved Motion Rate Control (RMRC)

Traditionally, high degree of freedom manipulators have been controlled using Resolved Motion Rate Control (RMRC) (Whitney 1969; Sciavicco and Siciliano 2005). RMRC is a reactive algorithm that has the advantage that it does not require inverse kinematics methods, which are prone to many numerical problems, including non-unique solutions. RMRC, on the other hand, maps joint velocities to end effector velocities through the Jacobian matrix,

$$\upsilon = J(q)\dot{q}, \tag{1}$$

where $q$ and $\upsilon$ represent the state and velocity of the controllable degrees of freedom and $J$ is the Jacobian. By rearranging (1), one can see that desired end effector velocities are converted into joint velocities via the inverse Jacobian or, where the inverse does not exist, the pseudo-inverse.

This control scheme can be extended to the control of mobile manipulators. In this extension, the Jacobian becomes a combination of the base and arm Jacobians,

$$\upsilon = [J_{Base}(q)\, J_{Arm}(q)]\dot{q}. \tag{2}$$

As with the manipulator Jacobian, motions of the mobile base map to linear and angular velocities of the end effector via the mobile base Jacobian. That mapping is dependent on the configuration of the manipulator, meaning that motions of the mobile base will affect the end effector linear and angular velocities differently given the manipulator joint angles. The mobile base Jacobian is also dependent on the particular base and its mode of control. The Jacobian is

a partial derivative of the base control model with respect to the control parameters,

$$J_{Base}(q) = \frac{\partial S(q)}{\partial q_i}, \tag{3}$$

where $S(q)$ is the control model of the mobile base, which is dependent on the current state of the base. Therefore, the size and representation of the Jacobian will be different for different base types. Note that this technique is only of use on mobile platforms where a control input maps directly to a change in pose, such as skid steer, differential drive, translation stages, bases with mecanum wheels, etc. Platforms that do not adhere to this restriction, such as sychro-drive or omni-drive bases, require further consideration beyond the scope of this paper.

### 4.2 Secondary control

In typical applications, RMRC tends to produce undesirable effects. Often the manipulator will hit singularities, reach joint limits, or become maximally extended. To overcome these limitations and achieve better control of the system, secondary controllers can be used. A technique that we use to minimize these undesirable effects is to reconfigure in the *null space* of the mobile manipulator's configuration space. High degree of freedom mobile manipulators often have a high degree of redundancy and therefore large null spaces in which the system can reconfigure, without changing the pose of the end effector. Secondary control within RMRC is expressed mathematically as:

$$\dot{q} = J^{-1}\upsilon + N\dot{q}_0 \tag{4}$$

where $N$ is a null space projection of the Jacobian and $q_0$ is a set of additional joint velocities. Specifying $q_0$ is not a trivial task, but there are several standard approaches from the manipulator research literature (Wampler 1986; Nakamura and Hanafusa 1986; Chiaverini 1997). In particular, improving the manipulability measure is a common technique and one that takes on new possibilities in the context of mobile manipulation. Manipulability is a metric that represents distance to a manipulator singularity in joint space. In this context, manipulability is computed as

$$M = \sqrt{\det(J J^T)}. \tag{5}$$

Note that when a joint reaches a singular configuration, the Jacobian loses rank and M becomes zero due to an eigenvalue of zero for $J J^T$. We would like to maximize this metric to avoid singularities and improve control. In the domain of mobile manipulation, the base can be used to improve manipulability much more than is possible with a manipulator alone. For example, as the arm extends towards the edge of its workspace, the mobile base can assist by maneuvering closer to the goal and allowing the arm to stay in a state of high manipulability. This technique proved to be valuable in our experiments.

### 4.3 Orientation control

In the framework of RMRC, end effector velocities are often expressed as the difference of the current and desired poses. While this technique works well in Cartesian coordinates, computing orientation differences is prone to problems. Expressing orientation error in terms of differences of Euler angles is a classic problem in fields that deal with attitude control. In particular, the problem referred to as *gimbal lock* becomes a significant hurdle in the context of high degree of freedom manipulators. This problem occurs when the pitch angle of a controlled reference frame is near 90 degrees, resulting in the alignment, or near alignment, of the roll and yaw axes. Due to this redundancy of axes, there is a continuity of valid orientation representations (i.e. non-uniqueness) and therefore a continuity of valid error calculations. The result of this alignment of axes is instability in error-minimizing control schemes.

In the past, to avoid this problem, researchers have been constrained to operate in a range of small pitch angles. However, we have implemented a solution based on quaternions. Previously, orientations were expressed as matrices and then converted to Euler parameters for differencing.

$$e_0 = \begin{bmatrix} roll_d - roll_c \\ pitch_d - pitch_c \\ yaw_d - yaw_c \end{bmatrix}. \tag{6}$$

Under the new formulation, rotation matrices are converted directly into quaternion vectors and differencing is performed without any use of Euler parameters (Bar-Itzhack 2000). Given two coordinate systems, separated by a rotation, the quaternion orientation difference between these frames is expressed as,

$$\eta = \cos(\phi/2); \qquad q = \sin(\phi/2)r, \tag{7}$$

where $\phi$ represents the rotation between the two frames about a unit vector $r$. Now, consider two independent frames, current $\{\eta_c, q_c\}$ and desired $\{\eta_d, q_d\}$, measured relative to a common origin. The relative orientation between these frames is given by $\{\delta\eta, \delta q\}$ where

$$\delta\eta = \eta_d\eta_c + q_d^T q_c; \qquad \delta q = \eta_d q_c - \eta_c q_d - q_d^{\times} q_c. \tag{8}$$

$$r = \begin{bmatrix} r_1 \\ r_2 \\ r_3 \end{bmatrix}; \qquad r^x = \begin{bmatrix} 0 & -r_3 & r_2 \\ r_3 & 0 & -r_1 \\ -r_2 & r_1 & 0 \end{bmatrix}. \tag{9}$$

Yuan (1988) shows that relative orientation error between two arbitrary orientations is given by $e_0 = 2\delta\eta\delta q$, and is

shown to have two global equilibrium points. Only one results in true convergence; the other results in alignment rotated by 180 degrees. However, the undesirable equilibrium is eliminated if $\delta\eta$ is constrained to be positive. As Yuan has shown, this technique results in guaranteed convergence of the current orientation to the desired orientation under standard error reduction, i.e. proportional control. This guarantee is a major advantage over differencing of Euler parameters. Given this stability in error computation, the control inputs also become stable, since the error is mapped directly from the workspace to configuration space via the Jacobian. Therefore, when the manipulator is free of constraints, such as joint limits and obstacles, control of the end effector at arbitrary orientations is stable, as well.

This quaternion based approach to orientation control has expanded the workspace in which we can safely and effectively control the end effector. Our previous work was limited to a relatively small range of the pitch dimension, but now we are able to achieve stable control in any orientation.

## 5 Combining visual and force servoing

Due to sensor uncertainty, many researchers use a servoing approach to "peg-in-hole" type insertions. In our application, we are trying to insert a plug into a target hole (Fig. 5), where the tolerance is similar to the sensor noise. This makes reliable insertion particularly challenging. To overcome this, we use a combination of visual and force servoing techniques. This section describes both servo techniques and how they are combined in different ways. Results are presented showing the efficacy and efficiency of the various approaches.

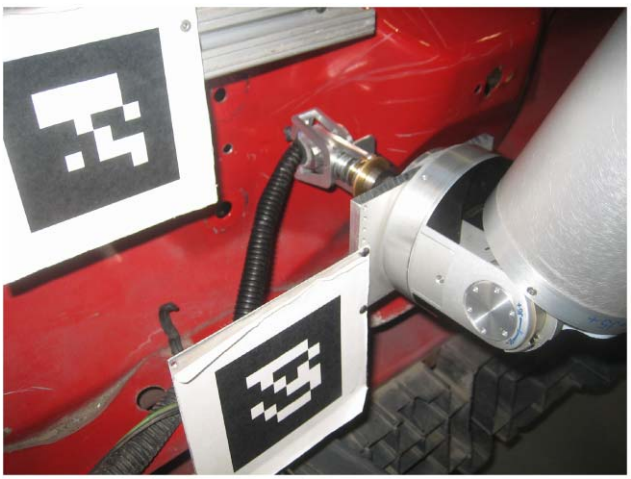

**Fig. 5** The arm servos the plug towards the target hole (the middle of the three holes shown)

### 5.1 Visual servoing

A common approach in vision based systems is to augment the environment to mitigate the challenges of object recognition and pose estimation. In our work, we utilize the ARTag design to help visually identify reference frames in the image (Fiala 2005). The ARTag system identifies the corners of each tag (one on the task board, one on the manipulator wrist) in both the left and right images of a stereo pair. Given the corresponding points in each image, our system performs triangulation to estimate the pose of the tags relative to the cameras. The relative location of tags is used to compute servo commands sent to the mobile manipulator. Figure 6 shows the error of the triangulation related to the size of the tag in the image using a Bumblebee 2 camera at $1024 \times 768$ resolution. Estimation error is generally less than 2 cm, but dramatically increases when the tag is more than 3 meters away from the camera.

All commands are sent relative to the end-effector's current position. If $p_d$ is the desired position of the end-effector relative to the target body and $p_m$ is the measured relative position from the cameras, then the controller adjusts the desired position of the end-effector as follows:

$$\begin{aligned} p_{err,t} &= p_d - p_m, \\ p_{cmd} &= k_p p_{err,t} + k_d (p_{err,t} - p_{err,t-1}) + k_i \left( \int p_{err} \right) \end{aligned} \tag{10}$$

where $p_{cmd}$ is the commanded position of the end-effector relative to its current position. The PID gains to be used are sent by the commanding process along with the command. The controller converts the relative position command into a new desired position in the global reference frame, then to a desired velocity by integration. Finally, the velocity control law described in the previous section is applied.

### 5.2 Force servoing

Humans make great use of force feedback in tight-tolerance insertion tasks. We would like our robots to behave similarly, by augmenting visual servoing with force control strategies. Using force feedback data from the manipulator, we implemented a simple force controller that allows the end-effector to achieve and maintain desired contact forces with the objects it encounters.

Our force controller works by adjusting the position of the end-effector based on force feedback. If the desired contact force error is too little in a dimension (in the end-effector frame), the controller moves the end-effector in that direction. If the force error is too great, the controller moves the end-effector in the opposite direction. If the desired force is

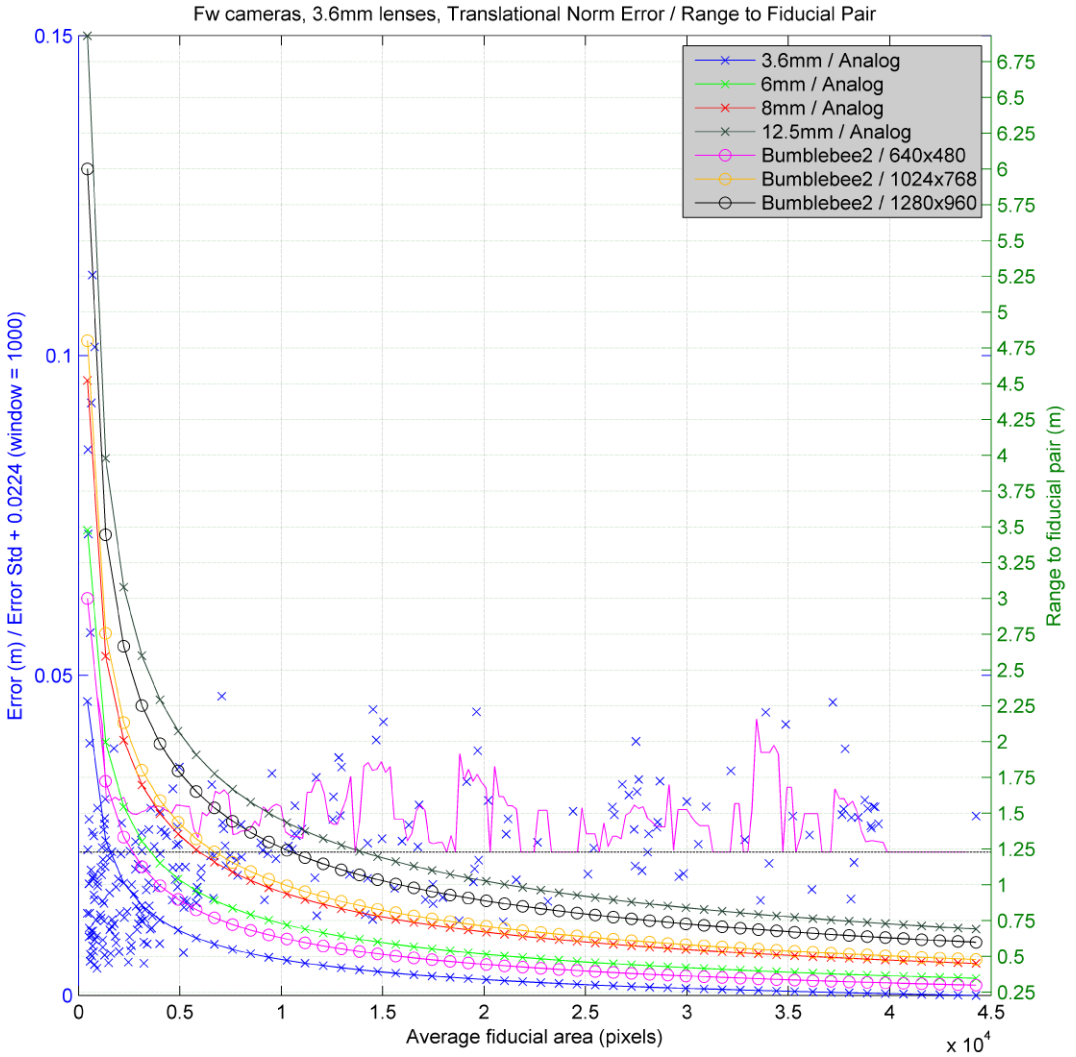


**Fig. 6** (Color online) Hybrid plot of tracking results as the fiducial gets closer to the cameras. The *blue* X's represent the error of individual measurements with our Bumblebee camera system, and are plotted according to the vertical scale on the *left*. As the fiducial gets farther away (on the *left* of the graph), taking up fewer pixels in the image, the standard deviation of the measurements increases (shown in *pink*). The *colored curves* represent, for different types of cameras, the number of pixels that the fiducial takes up in the image as the distance from the camera increases. They are plotted according to the vertical scale on the *right*. For example, we use a Bumblebee 2 at $1024 \times 768$, represented by the *orange curve*. At 0.5 meters from the camera, the fiducial takes up $4 \times 10^4$ pixels. At 1 meter, it takes up $1 \times 10^4$. At 3.0 meters it takes up $0.25 \times 10^4$ pixels. As seen with the *blue* X's, when there are so few pixels, the standard deviation in the fiducial measurement increases rapidly. Detection and tracking at this distance is unreliable

$f_d$ and the measured force is $f_m$, then the controller generates a relative pose command similar to that of the visual servo:

$$p_{cmd} = k_p * (f_m - f_d). \tag{11}$$

We found a simple proportional controller to be sufficient. The proportional gain ($k_p$) is specified along with the force command, since different gains are appropriate for different phases of the insertion task. As with the visual servo commands, this relative pose command is converted into a desired position in the global reference frame, then into a desired velocity. For an example of using different gains in a static situation, see Fig. 7.

### 5.3 Constrained moves

We desire that the manipulator controller be usable in various scenarios. To this end, the control program should know as little as possible about the task at hand. On the other hand, it is infeasible to have the task-level executive monitor the status of the manipulator in real-time at relatively high rates (e.g., 30 Hz). Instead, we prefer to provide the controller with a command to achieve a given pose or force, along with constraints on the conditions under which to stop.

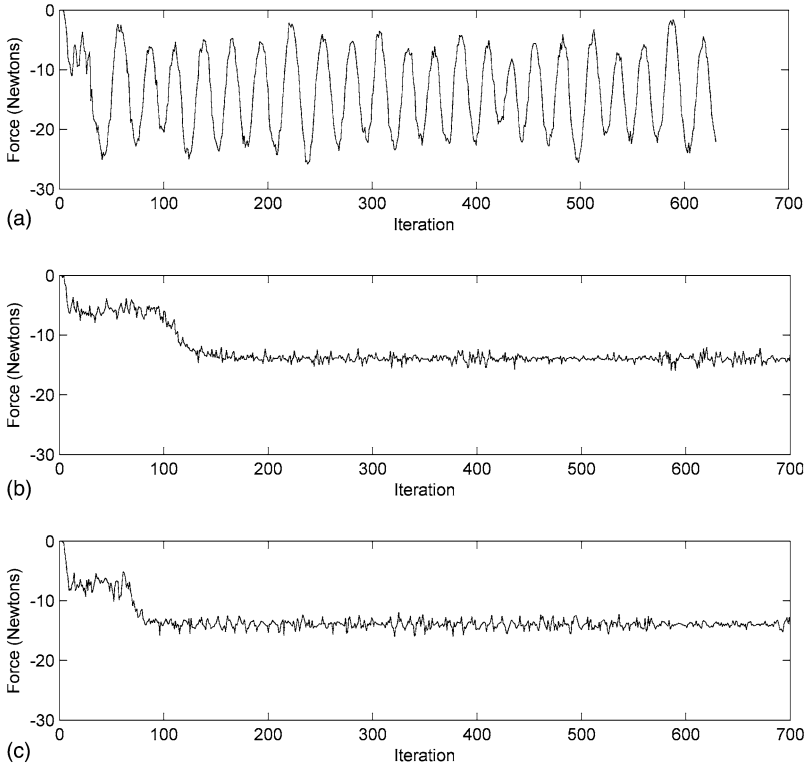


**Fig. 7** Comparison of gains for force control. Each plot shows the force in the $Z$ direction on the end-effector as the system is commanded to maintain a $Z$ force of $-15$ Newtons. The end-effector is started in open space, so there is a short period in which there is little detected force before it contacts the surface. In (**a**), the gain constant is 0.3 and the forces oscillate erratically. In (**b**), the gain is 0.03. The forces oscillate much less, but the response is slow. In (**c**), the gain is 0.1. The response time is much better without causing too much oscillation

In particular, we have designed and implemented a set of *constrained motion commands*. A constrained motion command consists of a *motion primitive* and a set of *constraints* on which to stop the motion. The motion primitives represent simple models of motion commands for the end-effector. Examples include a command to *go to a goal pose*, a *velocity command* to be executed indefinitely, and a command to *spiral outwards from the end-effector's starting point*. In cases where the motion completes (either as desired or due to an exceptional condition), the controller stops the end-effector and sends a status message to the higher-level task executive.

A *constraint* represents limits on end-effector motions. For example, we may want a move to be stopped if the force on the end-effector exceeds a certain value in a given direction. We may also want to ensure that the end-effector does not cross some bounds on its position. To that end, we designed a set of constraints that are as generic as possible—they can be used to place limits on position, velocity, or force, and can work in any combination of the six degrees of freedom: $x$, $y$, $z$, roll, pitch, and yaw. A constraint can be applied in the end-effector frame, the global reference frame, or any other specified reference frame. The constraint can be less than, greater than, or equal to a specified bound.

The set of constraints can be used in one of two ways. The first is as a *hard limit*, where motion is halted, and a "failure" signal is sent, if the limit is reached. For example, a force constraint could be used to improve safety by specifying that the controller should stop if any Cartesian force exceeds a given limit. If the arm collides with an unexpected obstacle and the force exceeds the limit, then the controller will stop the robot.

The second use of constraints is as a *regulator*. In this case, the controller does not stop the motion if the constraint is violated, but instead adjusts its command to the end-effector to attempt to meet the constraint. This is accomplished with the proportional force controller described above, with the gain specified as a parameter of the constraint. If the constraint is not violated, then no adjustment is necessary.

The motion primitive and set of constraints are combined into a *constrained motion command*. The task-level executive sends constrained motion commands, which are then executed. The command completes when either the motion is done or a hard limit constraint is violated. At this point, a signal is generated, consisting of the status of the motion (how close it was to completion) and a list of the constraints, if any, that terminated the motion. This information can then be used to reason about any problems that might have occurred, and to address them.

### 5.4 Servoing strategies

We explored three servoing strategies that differ in how they combine visual servoing and force control, to demonstrate the tradeoff between task cycle time and task completion

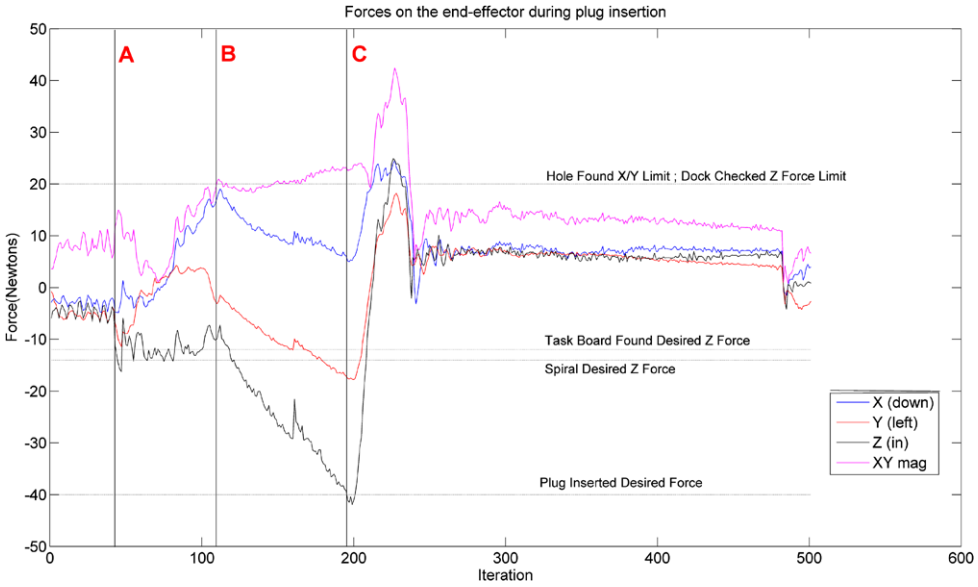


**Fig. 8** Recorded forces on the gripper during the force insertion, broken into component directions. The $+X$ direction is down towards the floor, $+Y$ to the left of the task board, and the $+Z$ direction points towards the task board. The "$XY$ mag" plot is the magnitude of $X$ and $Y$ forces together. The robot first drives the plug towards the task board until the $Z$ force crosses the "board found" threshold, at iteration $A$. Next, the robot spirals the plug along the task board until the "hole found" threshold is exceeded for the $XY$ magnitude, at iteration $B$. Then the robot drives the plug into the target hole until the "plug inserted" threshold is exceeded in the $-Z$ direction, at iteration $C$. Finally, the robot pulls the gripper away from the task board. During this move the force in the $+Z$ direction exceeded the "dock checked" threshold, so the system concludes that the plug was inserted correctly

reliability. The first scheme uses only visual servo (abbreviated VS in the results below). It is our baseline method. The scheme commands the mobile manipulator's wrist to a waypoint near the insertion point (7 cm from the surface, in the results reported below), where the distance is continually updated using visual servoing. This first waypoint has a fairly large tolerance (3 cm). Next, a closer waypoint (2 cm away from the hole) with a tighter tolerance (1 cm) is commanded. Finally, the insertion itself is performed by specifying a waypoint on the other side of the target hole.

There are several things to note about this strategy. First, the first waypoint is specified in order to roughly align the end effector (and the plug) with the target hole. Our experience showed that specifying this waypoint significantly increased the reliability of the insertion, over moving directly to a point in front of the hole. The insertion works best when the plug is inserted straight into the hole, not skew; the alignment step helps ensure that this will happen. Second, when moving to the first waypoint, both the base and the arm of the mobile manipulator can move; for the last two waypoints, only the arm is allowed to move. This is because the base motion is relatively coarse, and we found that allowing coordinated moves when doing fine motions led to insertion failures. The coordinated controller runs position control on the end effector, which relies on pose updates from the arm and mobile base. Base position updates arrive at over 10 Hz, but this is not frequent enough to execute tasks with millimeter precision. Finally, force constraints are added to all of the moves that cause motions to stop if unexpected forces are encountered.

To verify that the insertion occurs successfully, the arm is moved backwards after attempting to arrive at the last waypoint. If a large force is detected (see Appendix A), the system concludes that the plug has been successfully inserted. Otherwise, the system retries the insertion a set number of times (ultimately giving up and notifying the user if the insertions continue to fail).

The second scheme, which we call *force insertion* (abbreviated FI in the results below), uses visual servo only as a rough alignment step, then completes the insertion using a series of constrained motion commands with force limits and force control (Fig. 9). The first step in the force insertion task is to have the end effector touch the surface. This is a relatively simple matter: Assuming that the plug is pointed roughly perpendicular to the surface following the visual servo alignment, the system commands the end effector to move forward until a large force is detected in the negative $Z$ direction. This force means that the plug is pushing against something, presumably the surface. See Appendix A for a list of all force limits, thresholds, and gains used in the force insertion.

Once the surface has been found, the system needs to find the target hole. We use a spiral move to accomplish this. Again, assuming the plug is pointed roughly perpendicular to the surface, a constrained motion command is issued to have the end effector spiral outwards from the point

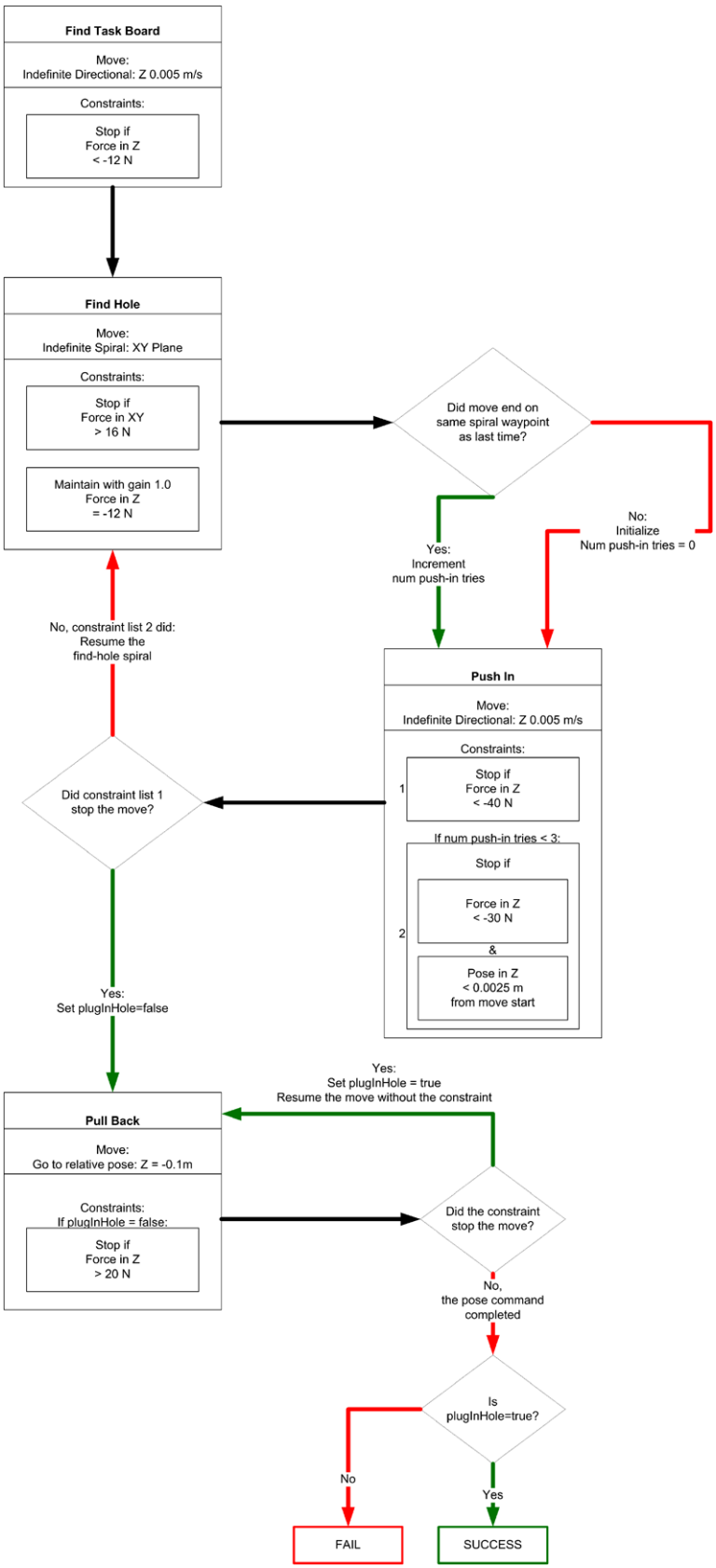


**Fig. 9** A flow chart of the constrained motion commands used to complete the force insertion task. When a motion is completed or a constraint triggers a stoppage, the task logic proceeds as the *arrows* indicate. Note the second constraint in the "Find Hole" step is a regulator constraint, not a hard limit

it hit the surface. The spiral is controlled in the $X$ (down)-$Y$ (right) plane of the end effector. A regulator constraint is provided to ensure that the plug maintains contact with the surface by using force control in the $Z$ direction of the end effector. When the plug crosses the target hole, the force in $Z$ drops because it is no longer pressing against the surface, and the force controller pushes the plug slightly into the hole. With the plug caught in the hole, the spiral is no longer maintained. A hard limit constraint is provided to ensure that motion stops when a large force is detected in the plane of the spiral (the $X$ and $Y$ directions).

Once the hole is found, the system uses a constrained motion command in which the end-effector is moved forward, stopping when a very large force is read, meaning the plug is fully inserted. Finally, the end-effector is commanded to pull away from the surface. The same check is used as in the visual servo scheme to check whether the plug was actually inserted. Figure 8 shows the forces on the end-effector as the force insertion progresses.

The third scheme combines visual servo and force insertion (abbreviated VS-FI). Again, the scheme starts with a visual servo to the same rough waypoint as in the previous two schemes. Then the system attempts to do a visual servo as in the first scheme, with the same limitation on the force on the end-effector. Once the visual servo has stopped, the system starts the force insertion. This scheme also uses the autonomous plug-docked check. The idea of this scheme is that the visual servo could succeed by itself, so the force insertion may not be necessary, but for a small time cost we could still attempt the force insertion and yield greater success.

We tested the three insertion schemes for accuracy and elapsed time. Each trial had three *attempts* at a plug insertion, in which the system attempted an insertion, pulled back, and used the autonomous check to determine whether the plug was inserted. The results are shown in Table 1, where the best results in each row are in bold face. As expected, the visual servo alone was the least successful method, due to the inaccuracy of visual tracking. It had 15 failed attempts and 3 trial failures (i.e., three failed attempts in a row). Due to the stereo cameras' placement on the body of the mobile base (Fig. 3) and the extension of the arm during the docking procedure, the cameras were approximately 1.5 meters away from the task board's fiducial. As shown in Fig. 6, at this distance the standard deviation of pose estimation is over 3 mm, which is the radius of our target hole. Thus, it is not surprising that visual servo alone failed relatively frequently in our tests.

The force insertion performed slightly better, but without a visual servo step the plug is not aligned well enough with the task board to ensure a successful insertion. On the other hand, this insertion scheme is significantly faster than visual servo. The main reason for this is that the visual servo takes time to carefully align the plug with the hole, whereas the force insertion does not use the fine alignment.

Performing a visual servo followed by a force insertion yielded the best results, with only six failed attempts and no trial failures. Though the visual servo is not accurate enough to reliably insert the plug into the target hole alone, the force insertion quickly makes up for small alignment errors. This proved successful when the plug was misaligned from the target hole by up to 20 mm. The unsuccessful tests for this scheme resulted from poor alignments from the visual servo, with the waypoint a large distance away from the surface.

On the other hand, VS-FI takes significantly more time than FI and the same (statistically) amount of time as VS, since it essentially performs all of VS before attempting to do the force insertion. Before conducting later tests, we experimented with the visual servo proportional gains to find an optimal balance between task success and completion time, leading to faster completion times in later results.

The experiments described in this section show that visual and force information can be usefully combined to get better results than using either alone. There are other ways to

**Table 1** Comparison of various servoing strategies

| | Visual Servo (VS) | Force Insertion (FI) | Visual Servo-Force Insertion (VS-FI) |
|---|---|---|---|
| Number of Trials | 30 | 30 | 30 |
| Number of Successes | 27 | 28 | **30** |
| Success Percentage | 90% | 93% | **100%** |
| Number of Failed Attempts | 15 | 16 | **6** |
| Average Time per Attempt (seconds) | 21.33 | **11.84** | 21.99 |
| [standard deviation] | [6.48] | **[3.76]** | [7.29] |
| Average Time per Trial | 29.97 | **17.07** | 26.39 |
| (seconds) [Standard Deviation] | [14.35] | **[6.16]** | [10.84] |
| Average Time per *Successful* Trial | 27.05 | **16.45** | 26.39 |
| (seconds) [Standard Deviation] | [11.84] | **[5.88]** | [10.84] |

combine visual and force feedback, though. For example, a control scheme could simultaneously use visual information to servo left/right and up/down with respect to the target hole while using force information to maintain contact with the task board and insert the plug. It is our intention to explore alternate combinations in future work.

## 6 Task-level control

We organized our system using the *Syndicate* architecture (Sellner et al. 2006), which extends the 3T architecture (Bonasso et al. 1997) to deal with multi-robot coordination. The 3T architecture consists of behavioral, executive, and planning layers. The Syndicate architecture extends this to multi-robot coordination by enabling layers at each level to communicate directly with one another (Fig. 10). Essentially, this enables us to distribute the functionality of each layer amongst the robots. For instance, at the behavioral layer, we can set up a distributed visual servo loop, where the perception is done on one robot and the manipulation is done on another robot, which receives periodic pose estimates from the perceptual robot. In Syndicate, each layer on each robot is implemented as a separate process, communicating via a publish-subscribe message-passing package called IPC (Simmons and Whelan 1997). IPC provides a high-level interface for defining messages, registering handlers for messages, and sending complex data structures in a transparent, machine-independent fashion.

Since our application currently requires only a fixed plan, we focus only on the behavioral and executive layers. Although we have just one robot (where the base and arm are treated as a coordinated whole—see Sect. 4), for the sake of modularity, maintainability, and to facilitate concurrent operation, we use the multi-robot capabilities of Syndicate to divide the system into three independent agents—a *Vision* agent, concerned with visual servoing, a *Manipulation* agent, concerned with the base, arm, and force sensing, and a *Leader* agent, concerned with task decomposition and coordination.

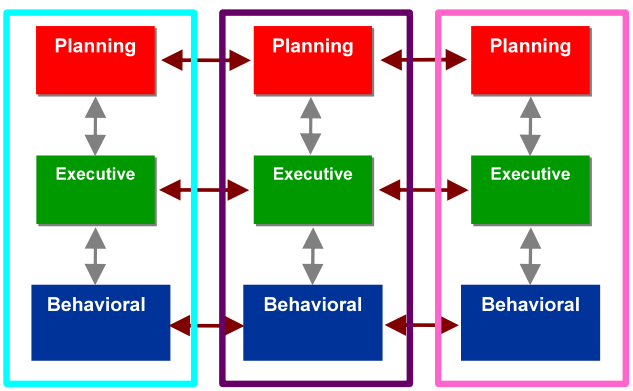


**Fig. 10** The Syndicate layered, multi-robot architecture

The behavioral layer has no knowledge of the greater task at hand, instead being concerned with performing commands as directed. It is primarily made up of *blocks*, each representing a single behavior, or *skill*, to be performed by the robot. Block behaviors include communicating to hardware control programs, receiving data from sensors, performing commands, and signaling to the executive layer. For instance, constrained motion commands are implemented using behavioral blocks, where the executive passes parameters to the command block, and violations of constraints are signaled back to the executive layer. Some behaviors are written to detect their own end conditions (e.g., "move to location $x$, $y$") while other behaviors operate indefinitely, with another behavior being enabled to detect the end condition of the task. This provides significant flexibility for mixing and matching behaviors to achieve needed functionality for different situations.

The blocks are organized by the *Skill Manager*. The Skill Manager can enable or disable blocks, allowing the behavior to run or not. It can also connect blocks, which allows blocks to pass data or commands to each other. To facilitate distribution at the behavioral layer, the Skill Manager includes support for making connections between input and output ports on different processes. This is transparent to the behavior itself: from a coding perspective, the function that implements the behavior cannot tell whether a port is connected to a behavior on the same, or different, process. The Skill Manager uses IPC to send data between ports on different processes.

The executive layer knows information about the task to be completed, and decomposes it into subtasks. It can send commands to the behavioral layer, receive status, and deal with errors by spawning new tasks. It interacts with the behavioral layer by enabling and disabling blocks (skills), setting their parameter values, and specifying dynamic connections between the input and output ports of blocks. The executive also specifies where to route status signals that skills may emit (e.g., success, failure, and sensor data).

The executive layer is implemented with the Task Description Language (TDL) (Simmons and Apfelbaum 1998), a language for distributed task-level control that is an extension of C++. TDL supports hierarchical task decomposition, task sequencing and dispatching, execution monitoring, and exception handling. It is based on a hierarchical representation of tasks call *task trees*. TDL generates task trees dynamically, based on sensed conditions. In this way, the same TDL program can generate many different behaviors, depending on the situation at hand. In addition, in the Syndicate architecture, an executive of one agent can spawn tasks on another agent. This facilitates coordinating of distributed tasks. Figure 11 shows an example task tree for one of the

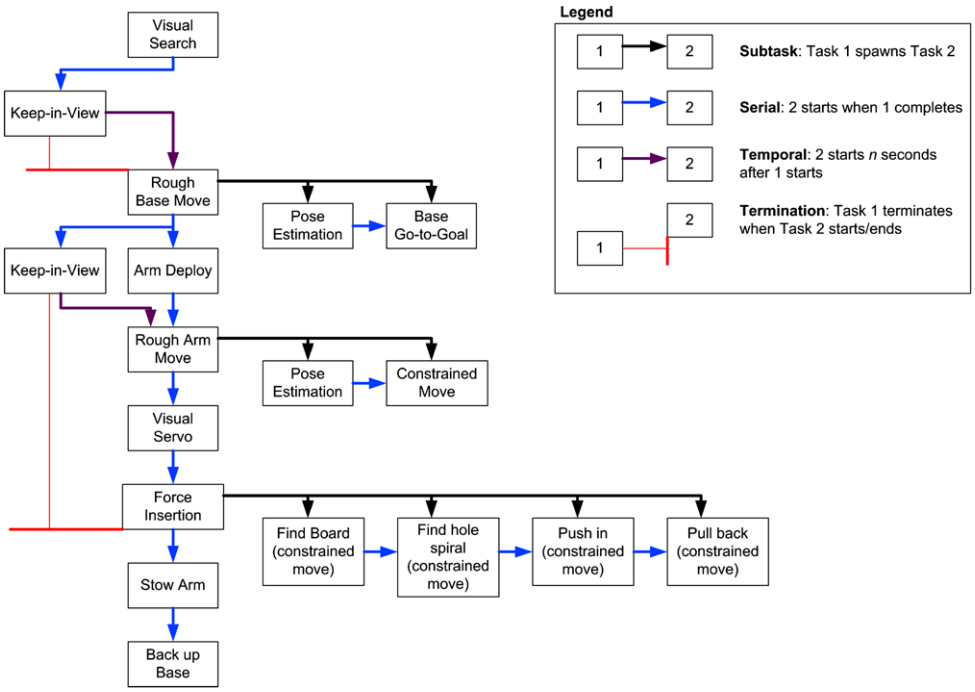


**Fig. 11** Task tree for a scenario in which a mobile base searches for and approaches a task board, then inserts the plug using the VS-FI insertion scheme

servoing strategies used in our application. Not shown are the failure-handling strategies, which we describe in more detail in the next section. The executive interprets certain signals from the behavioral layer as task failures—for instance, unexpectedly high (or low) force readings in certain directions. As a consequence, various tasks may be terminated (stopping the corresponding low-level behaviors) and others can be inserted into the existing task tree (causing recovery actions to take place), as shown in Fig. 9.

## 7 System integration

The autonomous software system runs on two CPUs. The main CPU is on board the mobile base and runs all executive and behavioral agents, as well as the interface to the stereo cameras, pan-tilt unit, and mobile base. The coordinated controller process runs on a computer on-board the manipulator itself, in order to achieve a tight control loop with the arm hardware. A diagram of the integrated system is shown in Fig. 12.

### 7.1 Behavioral/block design

As mentioned in the previous section, the behavioral layer blocks are split between two agents. The vision agent runs the blocks necessary to read images from the camera, detect the fiducials visible in the images, triangulate and determine the pose of the fiducials relative to the cameras, and determine the body poses to which the fiducials are attached. The mobile manipulator agent runs the other blocks, which interface with the hardware controllers and implement the tasks as commanded by the executive. The executive is responsible for enabling and connecting blocks. Intra-agent block connections are implemented with shared memory. Inter-agent blocks are connected with IPC (Simmons and Whelan 1997).

As an example, consider the data flow in a visual servo. The stereo camera interface grabs images from the cameras and feeds them to the visual tracking blocks. The tracking blocks find the fiducials in the image and calculate the poses of the task board and the manipulator's wrist. They send those poses to the visual servo block, which has been activated by the executive. The visual servo block sends new desired poses for the gripper, which the mobile manipulator control block relays to the coordinated controller. The coordinated controller sends velocities to the arm and base to achieve the pose command. When the visual servo block has determined that the gripper pose is within tolerances of the desired pose, it signals to the executive that the visual servo is done. The executive then disables the visual servo block and proceeds to the next task in the task tree.

The force insertion steps described in Sect. 5 are implemented as a series of constrained move tasks. Each task enables the constrained move command block, which sends the command and waits for a "done" response. When the command is completed, or a constraint is violated, the block is notified of the status, along with any violated constraints. It then signals the executive with the same information. When a constrained move calls for force control, this control is implemented within the coordinated controller itself.

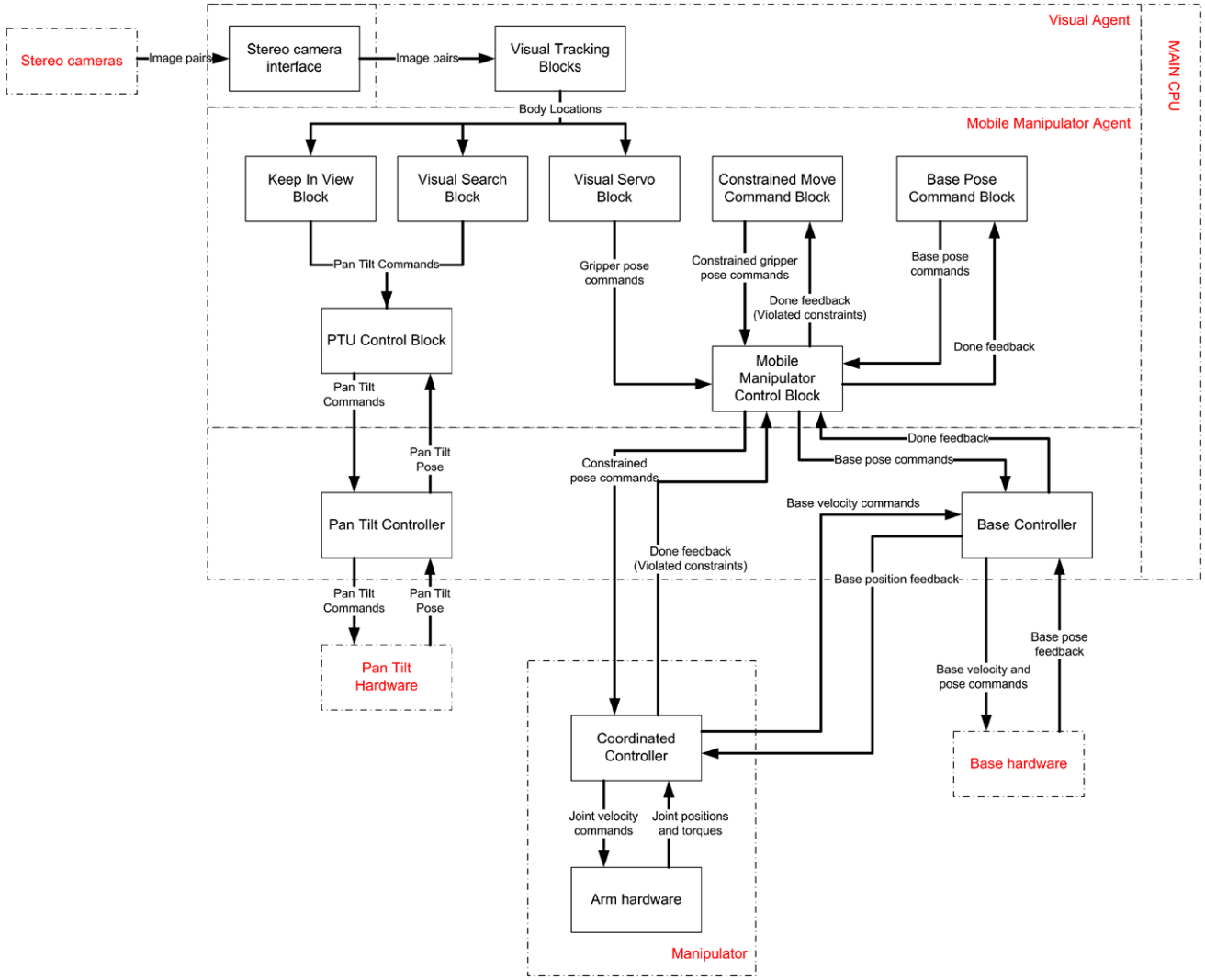


**Fig. 12** Integration of the software modules and hardware on our mobile manipulator. Each agent is a process running on the main CPU. Blocks communicate via connections made by the skill manager, which are implemented with shared memory for intra-agent connections and IPC for connections across agents. All interprocess communication is implemented with IPC

### 7.2 Executive/task design

In our test scenario, the robot starts in an unknown location relative to the task board (Fig. 2). The robot must identify the task board and approach it. This is accomplished by first performing a visual search task (the task tree is shown in Fig. 11). The visual search task commands the mobile manipulator to pan and tilt the stereo camera pair until the task board's fiducial is found in the image. Once the fiducial is found, a keep-in-view task centers the fiducial in the image, and a task is begun to approach the task board. This task is broken into two sub-tasks: one to average a few fiducial pose readings for greater accuracy, followed by a task to move the base to a predefined waypoint in front of the task board.

Once the base is positioned in front of the task board, another keep-in-view task is spawned to center the task board and wrist fiducials in the cameras' images. This task continues to run throughout the insertion process. However, it is still possible for one of the fiducials to go out of range of the camera. In this case, the executive process spawns another visual search task, to find the missing fiducial, and sends a command to the manipulator to bring both fiducials back in view. This case did not actually arise during our experiments, and is therefore not depicted in our task tree.

The manipulator arm begins in a stowed position. It is deployed using a fixed joint position task, then moves in towards the task board using a two-part task similar to the base move described above. The arm move is implemented with a constrained motion command that has a safety constraint applied to stop the arm should it happen to collide with the task board (evidenced by a large force detected in any di-

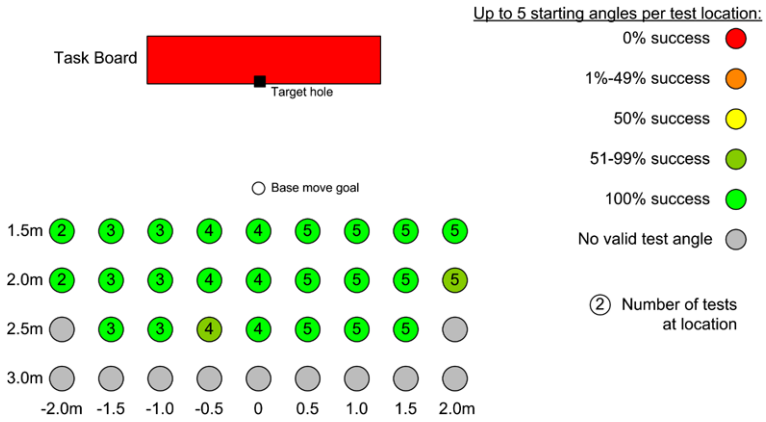


**Fig. 13** (Color online) Results for the second experiment. The number of trials at each location is shown in its *circle*. The number is different only due to the task board fiducial being occluded by the arm at certain test angles. *Color* indicates the success percentage of the trials performed from each starting location. *Grey circles* indicate that the test location is over 3 meters away from the task board

rection). Then the gripper is aligned more precisely with the task board using a visual servo task.

The force insertion process described in Sect. 5 is implemented as a series of constrained move tasks. The gripper finds the task board by moving forward until a large opposing force is detected (see Appendix A for a list of the gains and limits). Next, the coordinated controller maintains contact with the task board while searching for the target hole. When the hole is found, the controller pushes the gripper forward again until the plug bottoms out, and then checks if the plug has been inserted. This is done with a constrained move that pulls the gripper backward, with a constraint that the motion should stop if a large opposing force is detected (meaning the plug has been pulled out of the gripper). When the constrained move task signals its completion, the executive layer checks whether the constraint stopped the motion or whether it completed naturally. If the constraint stopped the motion (success), the executive layer spawns tasks to restow the arm and back the base away from the task board.

If the pull-back motion completed naturally (failure), the executive layer re-spawns the tasks to do the force insertion, starting with the visual servo alignment, with a small adjustment to the alignment waypoint. The executive uses the pose of the end-effector to determine the area of the task board explored by the robot during the spiral sequence, and adjusts the visual servo waypoint to command the robot to explore a different area on the next attempt. If the insertion fails after a pre-determined number of times (3, in our experiments), the task is aborted and the user is notified of the failure.

## 8 Results

### 8.1 Reliability tests

To test the reliability of the system, we ran over 100 experiments of the mobile manipulator performing the task as described in the previous section. The mobile manipulator and task board are those described in Sect. 3, pictured in Figs. 2–4. The robot was commanded to find the task board, approach it, and insert the plug, with the robot starting from a sampled grid of starting locations and angles. The grid was discretized with testing locations every 0.5 meters. It covered the area from 1.5 to 2.5 meters longitudinally from the task board and from −2 to 2 meters laterally. Starting locations over 3 meters from the task board were not considered, since visual pose estimation is so unreliable at that distance (see Sect. 5). At each location, we started the mobile manipulator at five different angles: −60, −30, 0, 30, and 60 degrees. Due to the configuration of the robot, with the camera to one side of the manipulator arm, the task board ARTag fiducial was not visible from all starting angles. Those locations were not considered. The testing locations used are illustrated in Fig. 13.

### 8.2 Analysis of results

We performed 101 experiments, one at each of the starting locations described. The results are described in Table 2. There were 99 successes, for an overall success rate of 98%. One failure was a heartbeat fault in the arm's controller program, most likely caused by a drop in wireless network connectivity. If a drop occurs while the controller is transmitting status to another process, the arm's controller does not receive its heartbeat and faults.

The other failure was also due to a communication issue. In the test, the rough base move was completed and the arm deployed correctly. The mobile manipulator agent registered that the arm deploy had finished and sent a message to

**Table 2** Experiment statistics

| | |
|---|---|
| Number of Trials (all using VS-FI) | 101 |
| Number of Successes | 99 |
| Success Percentage | 98% |
| Avg. Attempts per Successful Test | 1.06 |
| Avg. Time per Attempt (seconds) | 10.51 |
| [Standard Deviation] | [4.27] |

the leader agent. However, the leader agent never received the message. We have never previously seen this error in any of our experience with using the three-layer architecture and the IPC communication system, and do not even know whether to attribute it to a hardware or software issue.

During the 101 experiments, there were four instances of a missed attempt, in which a force reading triggered the low-level controller to complete its motion despite the plug not being inserted correctly. In each case, the system recovered with a completed insertion on the next attempt, which demonstrates the flexibility of the task control system. The executive layer detected the failed insertion attempt, adjusted the waypoints to search for the target hole in a slightly different location, and re-spawned the tasks to attempt the insertion again. In each case the second insertion attempt was successful. Note that the time reported starts when the rough base move has finished and the arm is deployed. Note also that the average time is twice as fast as in the earlier tests (Table 1), which is mainly due to finding visual servo proportional gains that strike an optimal balance between task success and completion time.

## 9 Conclusions

Robotic assembly requires a system with high flexibility and the capability to deal with inherent uncertainties and exceptions. A robust mobile manipulation system is one of many new and promising technologies that could address these challenges. In this paper, we have presented an autonomous mobile manipulator that can effectively overcome uncertainties and exceptions by using three key technologies—coordinated base and manipulator control, combined visual and force servoing, and error recovery through flexible task-level control. In particular, with constrained motion primitives that can detect constraint violation conditions, reactive task control can be used to significantly increase overall system robustness. This mobile manipulation system has been demonstrated experimentally to achieve high system robustness and reliability for a "peg-in-a-hole" task that is commonly encountered in automotive wiring harness assembly.

In the future, we intend to extend our work to deal with a moving target—that is, having the task board move, as if on an assembly line, and having the mobile manipulator follow the task board and doing the insertion on the move. We anticipate that, while details of the task approach will differ from the static case, the technologies reported in this paper will be able to handle this, and similar, more complex scenarios in the realm of autonomous assembly.

If, and when, a robust, reliable and capable autonomous robotic assembly system is achieved, the approach for robotic assembly will be fundamentally advanced, resulting in a much more reactive, adaptable, and intelligent system that requires minimal human intervention and removes the requirement of precisely known and rigid structure from the manufacturing and assembly environments.

## Appendix A: Gains and thresholds used in the docking experiments

| Gain/Threshold | Value | Dimension | Method of Determining |
|---|---|---|---|
| Visual Servo Proportional Gain | 0.7 | $XYZ$ relative to waypoint | Trial and error to maximize success and time of completion |
| Visual Servo Waypoint Distance Tolerance | 3 cm total ($XYZ$) | $XYZ$ relative to waypoint | Trial and error to maximize success and time of completion |
| Visual Servo Waypoint Angular Tolerance | 5 degrees in each direction | Roll, Pitch, Yaw relative to waypoint | Trial and error to maximize success and time of completion |
| Touch Board Force Threshold | −12 Newtons | $Z$ relative to gripper | Analyzed data of experiments driving gripper towards task board |
| Maintain Contact with Board Force Target | −14 Newtons | $Z$ relative to gripper | Trial and error to maintain contact with task board |
| Maintain Contact with Board Proportional Gain | 0.0015 meters/Newtons | $Z$ relative to gripper | Trial and error to maintain contact |
| Find Hole Force Threshold | +20 Newtons | $XY$ relative to gripper | Data analysis of experiments sliding plug along task board freely vs. catching target hole |
| Stop Inserting Plug Force Threshold | −40 Newtons | $Z$ relative to gripper | Data analysis of experiments pushing plug in until fully inserted |
| Plug Inserted Check Force Threshold | +20 Newtons | $Z$ relative to gripper | Data analysis of experiments pulling back uninserted plug vs. inserted plug |

## References


Bar-Itzhack, Y. (2000). New method for extracting the quaternion from a rotation matrix. *AIAA Journal of Guidance, Control, and Dynamics*, *23*(6), 1085–1087.

Bischoff, R. (1997). HERMES—a humanoid manipulator for service tasks. In *Proceedings of the international conference on field and service robotics*, Canberra, December, 1997.

Bonasso, R. P., Firby, R. J., Gat, E., Kortenkamp, D., Miller, D. P., & Slack, M. G. (1997). Experiences with an architecture for intelligent, reactive agents. *Journal of Experimental and Theoretical Artificial Intelligence*, *9*(1).

Brock, O., & Khatib, O. (2002). Elastic strips: a framework for motion generation in human environments. *The International Journal of Robotics Research*, *21*(12), 1031–1052.

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

De Schutter, J., De Laet, T., Rutgeerts, J., Decré, W., Smits, R., Aerbeliën, E., Claes, K., & Bruyninckx, H. (2007). Constraint-based task specification and estimation for sensor-based robot systems in the presence of geometric uncertainty. *International Journal of Robotics Research*, *26*(5), 433–455.

Dillman, R., Ehrenmann, M., Steinhaus, P., Rogalla, O., & Zollner, R. (2002). Human friendly programming of humanoid robots—the German collaborative research center. In *Proceedings of the IEEE international workshop of robot and human interactive communication*.

Donald, B. R. (1990). Planning multi-step error detection and recovery strategies. *The International Journal of Robotics Research*, 892–897.

Duchaine, V., & Gosselin, C. (2007). General model of human-robot cooperation using a novel velocity based variable impedance control. In *Proceedings of the second joint EuroHaptics conference and symposium on haptic interfaces for virtual environment and teleoperator systems*.

Edsinger, A., & Kemp, C. (2006). Manipulation in human environments. In *Proceedings of the IEEE/RSJ international conference on humanoid robotics*.

Fiala, M. (2005). ARTag, a fiducial marker system using digital techniques. *IEEE Computer Vision and Pattern Recognition*, *2*, 590–596.

Holmberg, R., & Khatib, O. (2000). Development and control of a holonomic mobile robot for mobile manipulation tasks. *The International Journal of Robotics Research*, *19*(11), 1066–1074.

Kim, J., Chung, W. K., Youm, Y., & Lee, B. H. (2002). Real-time ZMP compensation method using null motion for mobile manipulators. In *Proceedings of the IEEE international conference on robotics and automation*, Washington DC.

Kragic, D., Crinier, S., Brunn, D., & Christensen, H. I. (2003). Vision and tactile sensing for real world tasks. In *Proceedings of the IEEE international conference on robotics and automation*, Taipei, Taiwan.

Mason, M. (1979). *Compliance and force control for computer controlled manipulators*. MIT Artificial Intelligence Laboratory Memo 515, April 1979.

Mills, J., & Goldenberg, A. (1989). Force and position control of manipulators during constrained motion tasks. *IEEE Transactions on Robotics and Automation*, *5*(1), 30–46.

Nakamura, Y., & Hanafusa, H. (1986). Inverse kinematic solutions with singularity robustness for robot manipulator control. *Journal of Dynamic Systems, Measurement, and Control*, *108*, 163–171.

Peshkin, M. A., Colgate, J. E., Akella, P., & Wannasuphoprasit, W. (2000). Cobots in materials handling, In M. Cutkosky, R. Howe, K. Salisbury & M. Srinivasan (Eds.), *Human and Machine Haptics*. Cambridge: MIT Press.

Petersson, L., Jensfelt, P., Tell, D., Strandberg, M., Kragic, D., & Christensen, H. I. (2002). Systems integration for real-world manipulation tasks. In *Proceedings of the IEEE international conference on robots and automation*.

Raibert, M. H., & Craig, J. J. (1981). Hybrid position/force control of manipulators. *Transactions of ASME, Journal DSMC*, *103*, 126–133.

Sciavicco, L., & Siciliano, B. (2005). *Modeling and control of robot manipulators* (2nd ed.). London: Springer.

Schaal, S. (2007). The New Robotics—towards human-centered machines. *HFSP Journal*, *1*(2), 115–126.

Se, S., Lowe, D., & Little, J. (2005). Vision-based global localization and mapping for mobile robots. *IEEE Transactions on Robotics*, *21*(3), 364–375.

Sellner, B., Heger, F. W., Hiatt, L. M., Simmons, R., & Singh, S. (2006). Coordinated multi-agent teams and sliding autonomy for large-scale assembly. *Proceedings of the IEEE, Special Issue on Multi-Agent Systems*, *94*(7), 1425–1444.

Shin, D., Hamner, B., Singh, S., & Hwangbo, M. (2003). Motion planning for a mobile manipulator with imprecise locomotion. In *Proceedings of IEEE/RSJ international conference on intelligent robots and systems* (pp. 847–853).

Simmons, R., & Whelan, G. (1997). Visualization tools for validating software of autonomous spacecraft. In *Proceedings of international symposium on artificial intelligence, robotics and automation in space*, Tokyo, Japan, July 1997.

Simmons, R., & Apfelbaum, D. (1998). A task description language for robot control. In *Proceedings of conference on intelligent robotics and systems*, Vancouver, Canada, October 1998.

Wampler, C. W. (1986). Manipulator inverse kinematic solutions based on vector formulations and damped least-squares methods. *IEEE Transactions on Systems, Man, and Cybernetics*, *SMC-16*, 93–101.

Whitney, D. E. (1969). Resolved motion rate control of manipulators and human prostheses. *IEEE Transactions on Man-Machine System*, *MMS-IO*, 47–53.

Yamamoto, Y., & Yun, X. (1992). Coordinating locomotion and manipulation of a mobile manipulator. In *Proceedings of the 31st IEEE conference on decision and control* (Vol. 3, pp. 2643–2648), Tucson, AZ, USA.

Yoshikawa, T. (1987). Dynamic hybrid position/force control of robot manipulators—description of hand constraints and calculation of joint driving force. *IEEE Journal of Robotics and Automation*, *RA-3*(5), 386–392.

Yuan, J. S. (1988). Closed-loop manipulator control using quaternion feedback. *IEEE Journal of Robotics and Automation*, *4*(4), 434–440.


**Brad Hamner** received a B.S. in mathematics from Carnegie Mellon University in 2002 and an M.S. in robotics from Carnegie Mellon University in 2006. He currently works in the Robotics Institute on autonomous ground and air vehicles.

**Seth Koterba** has worked with and studied robotics at Carnegie Mellon University since 2003. His current research focuses primarily on mobile manipulation, in particular, coordinated control of high degree of freedom manipulators and mobile bases. A central element of his research is on understanding how to utilize the redundant degrees of freedom in high DOF systems to beneficially reconfigure without impacting end effector placement.

**Jane Shi** is a staff researcher at the Manufacturing Systems Research Lab, General Motor R&D Center of Warren, Michigan. Her current research focuses on addressing the fundamental challenges in achieving a robust, intelligent, and capable autonomous mobile robotic system for automotive assembly tasks. Dr. Shi is a recipient of the GM Boss Kettering Innovation Award (2006) and the GM R&D Charles L. McCuen Special Achievement Award (2006) and is a member of the IEEE and Robotics and Automation (RA) Society.

**Reid Simmons** is a Research Professor in the School of Computer Science at Carnegie Mellon University. He earned his B.A. degree in 1979 in Computer Science from SUNY at Buffalo, and his M.S. and Ph.D. degrees from MIT in 1983 and 1988, respectively, in the field of Artificial Intelligence. His thesis work focused on the combination of associational and causal reasoning for planning and interpretation tasks. The research analyzed the relationships between different aspects of expertise and developed a domain-independent theory of debugging faulty plans. Since coming to Carnegie Mellon in 1988, Dr. Simmons' research has focused on developing self-reliant robots that can autonomously operate over extended periods of time in unknown, unstructured environments. This work involves issues of robot control architectures that combine deliberative and reactive control,probabilistic planning and reasoning, monitoring and fault detection, and robust indoor and outdoor navigation. More recently, Dr. Simmons has focused on the areas of coordination of multiple heterogeneous robots, robotic assembly, and human-robot social interaction. Over the years, he has been involved in the development of over a dozen autonomous robots.

**Sanjiv Singh** received a Ph.D. in robotics from Carnegie Mellon University in 1995, an M.S. in robotics from Carnegie Mellon in 1992, and an M.S. in electrical engineering from Lehigh University in 1985. He is currently an Associate Research Professor at the Robotics Institute at Carnegie Mellon and an editor of the Journal of Field Robotics.