Supervised Remote Robot with Guided Autonomy and Teleoperation (SURROGATE): A Framework for Whole-Body Manipulation

Paul Hebert¹, Jeremy Ma¹, James Borders¹, Alper Aydemir¹, Max Bajracharya¹, Nicolas Hudson¹, Krishna Shankar², Sisir Karumanchi¹, Bertrand Douillard¹, Joel Burdick²

¹Jet Propulsion Laboratory, Pasadena, CA
²California Institute of Technology, Pasadena, CA

Abstract—The use of the cognitive capabilities of humans to help guide the autonomy of robotics platforms in what is typically called “supervised-autonomy” is becoming more commonplace in robotics research. The work discussed in this paper presents an approach to a human-in-the-loop mode of robot operation that integrates high level human cognition and commanding with the intelligence and processing power of autonomous systems. Our framework for a “Supervised Remote Robot with Guided Autonomy and Teleoperation” (SURROGATE) is demonstrated on a robotic platform consisting of a pan-tilt perception head, two 7-DOF arms connected by a single 7-DOF torso, mounted on a tracked-wheel base. We present an architecture that allows high-level supervisory commands and intents to be specified by a user that are then interpreted by the robotic system to perform whole body manipulation tasks autonomously. We use a concept of “behaviors” to chain together sequences of “actions” for the robot to perform which is then executed real time.

I. INTRODUCTION

With search-and-rescue operations occurring every year in harmful conditions as a result of natural disasters coupled with the growing need for in-home care for the growing population of the elderly, the need for robotics platforms to perform human-level tasks with commonplace tools is a growing need in the world today. Completely autonomous systems can often fail in situations not modeled in software algorithms. By leveraging the cognitive capabilities of humans to help guide the autonomy of robotics platforms, in what is typically called “supervised-autonomy”, much more can be achieved. Our framework for a “Supervised Remote Robot with Guided Autonomy and Teleoperation” (SURROGATE) is demonstrated on a robotic platform consisting of a pan-tilt perception head, two 7-DOF arms connected by a single 7-DOF torso, mounted on a tracked-wheel base. This paper presents our unique approach and software architecture that finds a nice balance between human teleoperation and robot autonomy. This paper is organized as follows: Section II presents related works on the topics of manipulation and supervised autonomy, Section III describes our approach and how it differs from other methods, Section IV explains how our approach has been implemented on a real robotic platform, Section V shows our framework in action as demonstrated on three human-level tasks, and Section ?? concludes with our future direction of work.

II. RELATED WORK

The state of the art in autonomous manipulation was highlighted in the DARPA Autonomous Manipulation Software (ARM-S) Program [1] which concentrated on dual-armed manipulation robot on a fixed base. Moving beyond fixed base robots, the problem of mobile manipulation has traditionally consisted of both wheeled [2], [3], [4] and legged robots[5], [6]. There have been numerous examples of wheel-based mobile manipulation robots which typically used single arm manipulators. The Stanford AI Robot (STAIR) [7] consisted of a Segway-like base with a single manipulator and hand from various manufacturers. The HERB robot [2] developed at Carnegie Mellon University and Intel originally started with a single arm manipulator but later added a second arm for dual manipulation. Herb’s mode
of mobility was also a Segway-like base. Willow Garage’s PR2 robot [3] is also a mobile dual-armed manipulation robot consisting of parallel grippers and an omnidirectional base. All of these robots will have difficulty transversing rough terrain unlike the SURROGATE platform. In addition, these robots have used small arm payloads on the order of 4-7 lbs, whereas the SURROGATE platform can handle an arm payload of 60-70 lbs.

The problem of “guided autonomy” has received a fair amount of attention recently as part of the advances in the DARPA Robotics Challenge trials [4], [6], [5]. Of relevance from these trials is the approaches of different teams in communicating the operator’s intent to the robot beyond joint-level teleoperation. Several teams specified operator intent either as robot-centric commands in terms of desired end-effector movements in cartesian space (fixtures by CMU [4]) or in terms of desired object centric commands (affordances by MIT [6]). JPL’s entry to the DRC trials was a quadruped legged robot (RoboSimian)[5]. In this paper, we extend RoboSimian’s approach from the DRC and report on a behavior centric architecture that encompasses language in terms of end-effector state transitions relative to an object as an asynchronous hierarchical state machine (described further in Section III-D). These behaviors stand at a higher level than direct end-effector cartesian teleop commands as they encode state transitions as a function of contact sensing. This enables closed loop execution of tasks and is more suited for generalization to full autonomy.

III. APPROACH

We present a framework for whole-body manipulation using a “Supervised Remote Robot with Guided Autonomy and Teleoperation” (SURROGATE) platform. We discuss here in detail the perception, pose estimation, modeling, planning for whole body maneuvers, behavior specifications, control, and user interface that comprise the entire system.

A. Perception

The SURROGATE robot is equipped with a perception head consisting of a high-resolution color stereo-camera pair, a Velodyne laser-scanner (HDL-32E), and an inertial measurement unit (IMU) all mounted on a pan-tilt unit. The perception system module is responsible for 3D map building, 6-DOF object localization, and pose estimation.

We maintain two voxel-based 3D maps of the environment, namely the manipulation and long-range perception map. The manipulation map is utilized to maintain a detailed, high resolution model of the world. This is used to determine obstacles in the workspace of the robot. Obstacles are segmented out of the map using the same framework described in [8] and represented as convex hulls for fast collision detection during manipulation planning. The manipulation map is also used to provide detailed situational awareness to the operator and compute 6-DOF object poses of known objects in the environment. The object pose is determined by matching the contours of the known object model against the contours in the acquired image. To facilitate the object detection process and leverage the “guided autonomy” available from the human operator, a region of interest and object type is only required as inputs from the user to fit the object. Together with the manipulation map, this provides an accurate enough estimate of the object pose to complete manipulation tasks, such as grasping the J-CAD chemical detector or turning the circular wheel-valve. The full 3D and high resolution nature of the manipulation map together with the real-time requirements of the system often requires the map size to be limited to the robot’s workspace.

The long-range map uses the same voxel-based representation though with lower resolution to extend the map into further distances. Range measurements acquired from the Velodyne sensor are used to populate this crude map which gives full 360° coverage around the robot. This particular map is used in mobility planning and overall situational awareness.

The pose estimation on SURROGATE is primarily stereo-vision based visual odometry [9] coupled with an INS solution for orientation provided by the IMU. However, pure stereo-vision with an IMU proved inadequate in providing accurate 6-DOF pose of the robot, primarily due to the majority of manipulation tasks often involving the robot arms coming into view of the stereo cameras. This has the undesired effect of either causing occlusions, creating phantom motions from tracked features on the arm of the robot, or creating unmodeled high-frequency dynamics of the robot unable to be captured by the slow framerate of the cameras. To mitigate this problem, a secondary form of pose estimation was fused with the existing pose estimator (stereo camera pair with IMU) on the SURROGATE platform using lidar point clouds with scan registration.

B. Pose Estimation from Laser Scan Registration

To provide the platform with accurate localization in all lighting conditions we developed in addition to the visual odometry (VO) system, a lidar based odometry system (LO). LO consists of a real-time implementation of the Iterative Closest Point (ICP) algorithm [10], in particular the point-to-plane version of ICP [11]. The implementation is single core, without hardware acceleration, and can produce a registration of two full Velodyne scans in 50ms on average. An example of the LO trajectory is shown in Fig. 2d. Our approach to lidar-based odometry can be summarized in two simple steps: a fast approximate nearest neighbor search for point association followed by a ground/non-ground segmentation step to lock in certain DOFs of the pose.

a) Fast approximate nearest neighbor search: ICP takes as an input a pair of scans: a model scan and a test scan. The model scan is kept fixed while the transform that allows the alignment of the test scan onto the model scan is estimated. The first step in each ICP iteration consists of finding point associations between the test and the model scan. This is usually implemented as a nearest neighbor (NN) search and is often the main bottleneck in terms of speed [12]. Efficient 3D data structures such as kd-trees or
The stereo cloud provided by the stereo pair on the sensor head is also shown at the center of the image to illustrate the difference in field of view between the two sensors, which contribute to more robust LO registration compared to VO. (b)-(c) Illustration of ICP registration; in (b) white lines indicate point-to-plane associations; in (c) the two scans are correctly aligned. (d) Example of LO path in green, the robot came back to the same position as it started from, the resulting drift was approximately 10cm, over a 10m long trajectory, see Sec. V-D.

oct-trees have been used to limit computation times. These typically result in NN search times of the order of 100\textit{ms} for single test scan matched to a single model scan, in the case of 360° Velodyne HDL-32E scans. In our implementation, these are down-sampled through a simple polar grid, and contain about 60K points each. The aim here is to perform ICP in less than a 100\textit{ms} window so that 10 Velodyne scans per second can be processed (i.e., run real-time). An ICP function call typically requires 20 to 30 NN searches. This means that we aim for NN search times of 5\textit{ms} or less. To achieve this we simply voxelize the model scan using a 3D grid. The voxel size in this grid is defined by the ICP distance threshold: the maximum allowed distance between a test point and its associated model point. Each voxel maintains an average of the model points in it. A NN search thus simply requires projecting tests points into the 3D model grid and only involves index lookups. This is fast and allows to achieve the targeted computation times of $\leq 5\textit{ms}$.

b) Exploiting segmentation: ICPs behavior is fairly sensitive to the point correspondences found during NN searches: ICP may diverge depending on the subset of point correspondences used to estimate the transform. To increase the algorithm robustness we exploit the Velodyne segmenter developed in [13]. This segmenter provides ground and non-ground segments. Our implementation differs from the original algorithm in that the second part of the algorithm, which aims at segmenting objects once ground points have been removed, is entirely performed in the range image as opposed to using an intermediate voxel grid. Neighbor lookups are thus significantly faster, which results in segmentation times of about 15\textit{ms} per scan. This is an order of magnitude faster than the original implementation. Our ICP implementation leverages the resulting segmentation to first align ground points in the test scan to ground points in the model scan. In a second phase, while ground alignment is maintained, only non-ground segments are aligned. Fig. 2b illustrates point associations selected between non-ground segments only during this second phase. This segmentation based approach to alignment implicitly decreases the number of transform components estimated during each of these two phases and thus facilitates ICP convergence. The “ground” phase essentially estimates the roll, pitch and z displacement of the registration while the “non-ground” phase estimates the remaining components. Overall, the alignment of two scans takes about 50\textit{ms} on average: segmentation takes 12\textit{ms}, normal computations (for point-to-plane distances) takes 7\textit{ms}, and ICP takes about 30\textit{ms}. The output of this alignment process is illustrated in Fig. 2c.

C. Modeling

Our modeling layer provides several functions to the system. The first function is to provide visual models for drawing of the robot and object models in the Operator Control Unit (OCU) interface using OpenGL\textsuperscript{TM}. The second function is to provide collision detection capability using the third-party software Bullet. The third function is to provide a kinematic model such that kinematic quantities can be easily computed.

A model manager provides the management of this layer, and maintains the state of the kinematic, geometric, and physical quantities of not just the robot but also objects and the environment. For fast collision detection, the robot model and object models are modeled using primitive objects such as boxes, cylinders, and spheres. Collision detection with the world can be done using the Bullet collision detector library since the perception system produces both bounding boxes and simplified convex hulls of segmented portions of the world map. The model manager also provides functions for kinematic requests such as forward and inverse kinematics (see Section III-E for details), Jacobians, center of mass, and collision information.

D. Behaviors

In order to pass intent to the robot, the operator of the SURROGATE platform requests behaviors through the operator control unit. At the highest level, behaviors can be requested/constructed by specifying one or a combination of the following specifications: desired end effectors, their poses, robot body pose, and objects in the world reference frame. Behaviors that operate on objects can be parametrized by defining attributes of the object. For example, to specify the behavior to rotate a valve, the valve can be represented by a torus’ radius whose central axis also serves to define the rotary motion of the rotation behavior. In other circumstances, for a certain object, a grasping behavior can predefine end-effector poses that represent grasp points based on the object’s geometry. Objects with features, such
as buttons, can also help define other behaviors such as “pushing a button” by their feature location on the object model. Figure 3 illustrates these various specifications.

At a lower level, the behavior system is essentially represented by an asynchronous hierarchical state machine which defines control goals at a task frame and monitors the execution during the synchronous control loop. In the behavior state machine, a single finite state is defined as an action. Each action is defined by a set of controllers (e.g. force setpoint along a particular axis) and end conditions (e.g. end at a particular force value on another axis). The controllers dictate motions, positions, and force and torque setpoint goals at the chosen task frame. End conditions determine the completion or failure of these controllers and determine the transitions in the finite state machine to the next action of the behavior. This translation from higher level intent to a state machine of actions, allows a precise set of controllers and end conditions to be used in the Generalized Compliant Motion (GCM) framework [14] inside the control loop.

Fundamental behaviors (e.g. grasp, lift, release) can be added into a sequence easily for the operator and the on board system is completely agnostic to the use of a sequence or a single behavior. This allows each behavior to have simple transition and well defined conditions aiding the user to test and develop more complicated sequences of behaviors. Note that this ability to use both single behaviors and sequences of behaviors is the key framework allowing our SURROGATE system to transition from heavy user teleoperation (single behaviors) towards full robot autonomy (longer sequences) as our capabilities on the platform become more intelligent.

E. Manipulation Planning for Whole-Body Motion

1) Behavior Planning: Planning for manipulation is ultimately dictated by the behavior the system wants to execute. At the highest level, for a behavior request, a dedicated behavior planner is invoked which takes the high level kinematic requirements of the behavior such as end effector goals, robot body pose, and expected motion of the behavior. For example, in considering a manipulation behavior, the goal is to determine the best end effector pose, robot pose, and starting angles to perform the entire manipulation task. Therefore, the planner can alter the user specified goals such that the entire motion is achievable. The planner can discretize the search for certain behaviors (e.g. grasp points around a valve).

The behavior planner for manipulation first determines an initial inverse kinematics (IK) solution to the start of the behavior, from which it then tests feasibility. For each initial IK solution of each discrete goal, a cost is associated with the difference in joint angles from the current state to the initial IK state. Each solution is ranked first by the percentage of the desired motion that is feasible and then ranked according to the joint-angle cost.

2) Whole-Body Planning: Given the kinematic nature of the SURROGATE robot, with an open link kinematic chain of 21-DOFs, inverse kinematics becomes a challenging problem both for planning purposes but also for real-time controlling purposes (See Section III-G for details). This kinematic problem is formulated as a quadratic program (QP) with linear constraints, such as self-avoidance collisions, torque (accelerations and velocity) limits, gaze directions, and stability. The problem can be formulated by minimizing the following:

\[ \min \| \dot{\theta} \|_2 P_\theta + \| V^{BW} - V^{BW}_\text{WE} \|_2 P_C + \| V^{BW}_\text{WC} - V^{BW}_\text{WE} \|_2 P_\theta \] (1)

where \( \dot{\theta} \) are the joints angle velocities of the robot, \( V^{BW}_\text{WE} \) is the base body velocity of the end effector frame (E) in the world frame W and \( V^{BW}_\text{WC} \) is the center of mass body velocity expressed in the base frame. The first term is the regularizing term and matrices \( P_C \) and \( P_\theta \) are weighting matrices of the end effector and regularizer, respectively. The minimization is subject to joint velocity, torque, and center of support constraints. In depth details can be found in [15].

F. Mobility Planning

Mobility planning consists of a hierarchy of three different planners i) a global grid-based path planner ii) a local trajectory selection scheme that enables path following while taking into account kino-dynamic constraints iii) an end-state planner that is capable of auto-generating navigation goals given desired end-effector poses and/or camera gaze constraints. The autogenerated navigation goals are used as inputs into the global path planner.

1) Global Path Planning: Global path planning is performed on 2D grids with the world classified as obstacle and obstacle-free cells in a binary fashion (See Figure 4a). The \( D^* \) Lite algorithm [16] is used to perform a graph search given a desired \{x, y\} goal. \( D^* \) Lite expands nodes in a depth-first fashion backwards from the goal instead of forwards from the start location. This backwards expansion enables efficient replanning in a dynamic fashion as the robot moves and the latest estimate of the world changes. Rigorous analysis of the \( D^* \) Lite algorithm is presented in [16].

2) End State Search: End state planning searches for a feasible robot state given a desired end effector pose and/or camera gaze constraint. The search is posed as a non-linear programming problem as defined below and is solved
using an open-source interior point method based non-linear optimization package (IPOPT [17]).

\[
\{x^*_\text{base}, \theta^*\} = \arg \min_{x^*_\text{base}, \theta} \{(\theta - \theta_{\text{nom}})^T (\theta - \theta_{\text{nom}})\}
\]

s.t. \[ LB \leq c_{\text{pos}}(f_{\text{ee}}(x_{\text{base}}, \theta), x_{\text{ee, desired}}) \leq UB \]

\[ LB \leq c_{\text{orient}}(f_{\text{ee}}(x_{\text{base}}, \theta), x_{\text{ee, desired}}) \leq UB \]

\[ LB \leq c_{\text{gaze}}(f_{\text{cam}}(x_{\text{base}}, \theta), x_{\text{gaze, target}}) \leq UB \]

where \( \theta \) is joint angles, \( x_{\text{base}} \) is the body pose in world frame, and \( f_{\text{ee/cam}} \) functions represent the end effector and camera forward kinematics. \( x^*_\text{base} \) is the optimized body pose in world frame, and \( \theta^* \) represent the optimized joint angles. \( x^*_\text{base} \) is used as a navigation goal into the dstar path planner discussed in Section III-F.1. \( c_{\text{pos}}(\theta^2) \), \( c_{\text{orient}}(\theta^3) \), \( c_{\text{gaze}}(\theta^4) \) are position, orientation and camera gaze constraint functions.

G. Control

The control module runs nominally at 100Hz while processing 1kHz data, as tested on the SURROGATE platform. The control module is responsible for executing the current action in the behavior’s state machine. In the case of manipulation tasks, actions may consist of planned motions, or task frame objectives, or both simultaneously. Planned motions are specified as a time series of waypoints in joint space. Task frame objectives are specified in terms of the activation status and input parameters of a variety of Generalized Compliant Motion (GCM) controllers, which include a Cartesian trajectory generator, closed-loop effect torque control, active compliance (spring/damper) emulation, and dithering. For mobility-based tasks (i.e. driving), an action is specified by a desired path in 2D space and a final desired heading. The control module implements a simple but effective path-following controller consisting of a look-ahead (i.e. carrot) generator and two virtually decoupled proportional controllers - one each for the translation and rotation velocities:

\[
u_t = K_1 e_t \cos(e_\theta), \quad u_\theta = K_\theta e_\theta
\]

where \( u_t \) and \( u_\theta \) denote the translational and rotational velocities, respectively, and \( e_t \) and \( e_\theta \) denote the distance from the carrot position and the heading error, respectively. The heading error is calculated as the difference between the current heading and the heading along the line connecting the current position to the carrot.

H. User Interface

A large part of the supervised remote operation of our framework relies on an intuitive user interface to the robot. We define this interface to be the “Operator Control Unit” (OCU), an interface which allows the human operator to interact with the robot at various levels of control. In our approach to designing the OCU, we desired to find a balance between complete teleoperation control of the robot and complete autonomy of the robot. To this end, we have developed two types of operator interfaces that allow human level interaction with our robotic platform: an “Engineering-OCU” and a “Tablet-OCU”. The Engineering-OCU is designed to be run on a computer desktop and exposes internal robot state information that additionally allows for direct control of the robot. The Tablet-OCU is designed for field operation use on a handheld device, requiring only high level directives from the user and relying more heavily on robot autonomy to execute actions and behaviors.

1) Engineering OCU: The Engineering OCU is written in C/C++ using OpenGL to define a 3D rendered world that can be navigated using a standard keyboard and mouse. Several widgets are used to frame the rendered world using the QT library that allow the user to insert objects, request maps from the perception module, specify behaviors to execute, and request planned actions to complete a particular behavior sequence. The top image of Fig. 5 shows the Engineering OCU as it is currently used, with the 3D world occupying a majority of the display, a 2D live image feed on the top left corner, and a number of widgets with buttons and sliders on the right side. Through the use of behaviors, entire sequences of actions can be specified easily through the OCU by chaining multiple behaviors together in sequence. A majority of the behavior chaining is done for the user via button presses and sliders in various widgets available through the main GUI display, greatly reducing unnecessary operator interaction time. One fairly useful tool developed in our interface is the ability to fly a free-floating hand into the environment. This “virtual-hand” (as seen in Fig. 4b)
allows the operator specify a desired behavior on an object in
the environment potentially outside the reachable workspace
of the robot and plan an entire sequence of mobility moves
followed by manipulation behavior sequences all in one request.

2) Tablet OCU: The Tablet OCU is written in C/C++
using the QT library for defining a number of widgets
that simplify behavior specification. The interface itself is
divided into various tabs allowing the user to observe process
health, live robot image feeds, and 3D plan previews of robot
behaviors prior to execution. The interface has been ported
and tested on a Panasonic ToughPad FZ-G1 using the touch
interface to identify objects, specify behaviors, and preview
and execute robot actions. The bottom image of Fig. 5 shows
one tab of the tablet interface as used by an operator to
control the robot.

IV. IMPLEMENTATION

The SURROGATE robot designed and built at JPL is a
mobile manipulation platform. Three 7-DOF limbs following
the limb design used on the JPL RoboSimian platform [5]
were used to make up the two arms and torso of SURRO-
GATE. The left and right limbs are connected via a chest-
plate that sits on the end-effector of the torso. The base of the
torso is mounted to a Qinetiq Talon tracked-wheel base. This
particular design for manipulation and mobility increases the
manipulation workspace of the robot, giving it a particular
advantage in reach and manipulability in comparison to other
robots such as the DARPA Atlas or the PR2 by Willow
Garage.

Each right and left limb have interchangeable end-
effectors: a Robotiq, Inc. three-fingered gripper or a rubber
stub. The gripper allows for grasping and actuation of man-
made objects (single-handed or dual handed manipulation
tasks) whereas the stub allows for bracing and support of
the robot when traversing over difficult terrain or reaching
further than the allowable balance polygon. All limbs, torso,
and tracks are driven by Elmo motor controllers communici-
ating over EtherCAT and RS-485 networks. An embedded
computer system is installed on the base of the Qinetiq Talon
base to control all actuation of the robot. The “control”
computer runs a real-time Linux kernel with an Intel Core
i7 processor.

The perception head of the robot is a stand-alone per-
ception system developed by JPL. It consists of two Im-
perx BobCat-B2520 color cameras (2448 × 2050 resolution)
connected via Camera-Link cables and arranged as a stereo
pair. The stereo-camera pair is connected to a pan-tilt motor-
actuated base to allow for user specified “look-at” behaviors
for improved situational awareness. Mounted on top of the
tilt motor is a Velodyne, Inc. HDL-32E laser scanner
spinning at a rate of 10 revolutions per second with up to
700,000 range points per second covering a 1−100m
range. This sensor provides long-range obstacle detection
and map building as discussed in Section III-A. Lastly,
an inertial-measurement-unit (IMU) exists on the tilt plate
providing angular rate and accelerations of the perception
head. The cameras, laser-scanner, and IMU data stream are
all synchronized into a common time frame via a micor-
controller unit (MCU). The MCU sends a trigger signal to
synchronize the camera capture between both color cameras
at 10Hz. The trigger signal is based off a pulse-per-second
(PPS) signal that comes from the Velodyne laser scanner.
This signal allows for the synchronization of all laser packets
into the MCU timeframe as well. The 200Hz IMU data
stream is also synchronized off the same MCU trigger
signal to send 20 packets of gyro and acceleration data
for every sync pulse. This allows camera data, laser data,
and IMU data to be represented all in the MCU timeframe,
allowing for synchronous data signals. Finally, an embedded
small form-factor computer is mounted to the underside
of the tilt plate, running an Intel Core i7 processor. This
“perception” computer runs a standard 64-bit Ubuntu-Linux
12.04 install and is responsible for running the algorithms for
stereo vision, visual odometry, lidar odometry, short-range
manipulation mapping, and long-range obstacle mapping –
running nominally at 7.5Hz. Both the “control” computer
and “perception” computer communicate over a dedicated
gigabit network.

V. EXPERIMENTAL RESULTS

Towards the development of the SURROGATE platform
we conducted a series of three unique manipulation tests and
a series of repetitive mobility tests to highlight navigation.
Of the three manipulation tests, the first consisted of an in-
strument panel with the goal to push a button that was on the
order of 1cm in diameter. The second test consisted of testing
mobility to navigate a short distance and to apply coarse
manipulation of turning a valve. The third test consisted of
fine bimanual manipulation of a chemical analysis tool. Of
the mobility tests, each test had the robot manually driven
away from the start location and autonomously commanded
to navigate back to the origin using the mobility framework
described earlier.

A. Manipulation: Instrument Panel Button Press

The first test (Figure 6) consisted of a stationary instrument
panel that was part of a battery pack. One of the goals
of this test was to exercise the perception capability of
the system. A considerable drawback to the DRC system
used in RoboSimian was the lack of perception autonomy in pose estimation of objects. The SURROGATE system still depends on a user to inform of the particle object type and the rough region where the SURROGATE system can expect to find it as demonstrated in Figure 5.

Another goal of the system was to exercise visual servoing of the end-effector as the 1cm diameter button was too small to achieve open-loop, considering the accumulated error of the system from the cameras to the end effector. The perception module detects fiducials at the end effector to provide an offset between the forward kinematic pose and the pose detected in the camera frame. This offset is then used to translate the end effector towards the goal that was also detected in the camera frame.

A third goal was to exercise whole-body planning. Given different possible locations of the panel, using each limb independently would not always guarantee a solution capable of pressing the button. In addition, as the fiducials are attached to the end effector, the ability to point the cameras at the button/end effector throughout the button press action became crucial to do so.

Once the robot is in front of the panel, the user can select a region in the camera image where the panel is located at which point the system automatically detects and places the model of the battery pack into the world. The user can select an appropriate behavior to conduct on the instrument panel. The push behavior is used which consists of linear motion into the button until a certain force level is experienced. Once the appropriate button is selected, the behavior planner decides how to orient the end effector above the button and the starting joint angles to ensure the linear motion is feasible. The manipulation planner then plans a trajectory of the joints to these starting joint angles. This trajectory and end behavior is sent back to the user to validate the execution. Once the execution is approved, the robot then executes the trajectory to the starting behavior joint angles. At the start of the behavior, the visual servoing provides one additional correction for the robot to execute. At this point the robot executes the behavior to push the button.

B. Manipulation: Valve Turn

The second experiment (Figure 7) had the objective to turn a valve that was some distance away from the robot. For this test, the user first had to provide a directive with the goal that was not in manipulation workspace to the robot. As such, end state planning was used to provide a goal base pose of the robot. The goal robot is then used by the mobility planner to plan a path towards. The end state planner requires the requested behavior sequence (grasp and then rotate), which ultimately provides a starting end effector pose of the behavior sequence to the end state search. The mobility planner then provides a global path to the controller to follow.

C. Manipulation: J-CAD Actuation

The third test (Figure 8) consisted of bimanual manipulation of a handheld chemical analysis tool called a J-CAD. In order to turn on this device, a rain cap is required to be twisted off. The perception capability is very similar to Sections V-A and V-B in that the user selects a region in the camera image in order for the perception module to detect and localize the J-CAD object.

Once the robot has traversed to the robot goal pose, the robot pauses and waits for human guided perception region-of-interest selection. Since the valve is in the workspace of the robot, the human can now select a region in the camera image for detecting the valve very similarly to Section V-A. Once the valve has been fit with the perception module, the user can now request the behavior sequence (grasp and then rotate). The behavior planner check feasibility and determines the best end effector pose and joint angles for the behavior. The manipulation planner then plans a whole body trajectory. The trajectory and behavior sequence is set back to the user to verify and validate. The user then requests the robot to execute. At the end of the behavior, the user can then request another behavior (release) to open the hand and backoff from the valve.

Figure 7. The robot during mid-valve turn after having navigated towards the valve.

Video of the instrument panel test can be found at the following URL: http://www-robotics.jpl.nasa.gov/tasks/taskVideo.cfm?TaskID=260&tdaID=700064&Video=152

5Video of the whole-body valve turn test can be found at the following URL: http://www-robotics.jpl.nasa.gov/tasks/taskVideo.cfm?TaskID=260&tdaID=700064&Video=152

6Video of the whole-body valve turn test can be found at the following URL: http://www-robotics.jpl.nasa.gov/tasks/taskVideo.cfm?TaskID=260&tdaID=700064&Video=152
results of the five runs resulted in displacements of [12 cm 10 cm 8 cm 9 cm 13 cm] from the origin when the robot was commanded to return to the origin, with an average displacement of 10 cm. Note that the threshold for the mobility planner on reaching the goal was 5 cm and the grid size used in the map for planning was 5 cm as well; so in actuality the pose accuracy was likely much better than 10 cm. The paths taken were approximately each 5 m long. While these navigation tests were rather simple exercises, these results capture system level navigation capabilities since several modules were run to produce these results, including track control, D-star navigation, perception, mobility, manipulation, and pose estimation.

VI. ACKNOWLEDGMENTS

The research described in this paper was carried out by the Jet Propulsion Laboratory, California Institute of Technology. Copyright 2015 California Institute of Technology. U.S. Government sponsorship acknowledged.

REFERENCES


