Team RoboSimian: Semi-autonomous Mobile Manipulation at the 2015 DARPA Robotics Challenge Finals

Sisir Karumanchi, Kyle Edelberg, Ian Baldwin, Jeremy Nash, Jason Reid, Charles Bergh, John Leichty, Kalind Carpenter, Matthew Shekels, Matthew Gildner, David Newill-Smith, Jason Carlton, John Koehler, Tatyana Dobreva, Matthew Frost, Paul Hebert, James Borders, Jeremy Ma, Bertrand Douillard, Paul Backes, Brett Kennedy
Jet Propulsion Laboratory, California Institute of Technology, 4800 Oak Grove Drive, Pasadena, California 91109
Sisir.B.Karumanchi@jpl.nasa.gov

Brian Satzinger, Chelsea Lau, Katie Byl
University of California, Santa Barbara, California 93106

Krishna Shankar, Joel Burdick
California Institute of Technology, 1200 East California Boulevard, Pasadena, California 91125

Abstract

This paper discusses hardware and software improvements to the RoboSimian system leading up to and during the 2015 DARPA Robotics Challenge (DRC) Finals. Team RoboSimian achieved a 5th place finish by achieving 7 points in 47:59 minutes. We present an architecture that was structured to be adaptable at the lowest level and repeatable at the highest level. The low-level adaptability was achieved by leveraging tactile measurements from force torque sensors in the wrist coupled with whole body motion primitives. We use the term "behaviors" to conceptualize this low-level adaptability. Each behavior is a contact-triggered state machine that enables execution of short order manipulation and mobility tasks autonomously. At a high level, we focused on a teach-and-repeat style of development by storing executed behaviors and navigation poses in object/task frame for recall later. This enabled us to perform tasks with high repeatability on competition day while being robust to task differences from practice to execution.

1 Introduction

The DARPA Robotics Challenge (DRC) Finals showcased robotic systems that were remotely operated in degraded communications. The event simulated a disaster response scenario where the robot, as a first responder, had to perform eight human-scale mobility and manipulation tasks within an hour. The eight tasks included 1) driving a vehicle, 2) egressing from the vehicle, 3) opening and entering a door, 4) turning a valve, 5) picking up a tool and cutting a hole in a wall, 6) performing a surprise manipulation task, 7) clearing debris or traversing irregular terrain, and finally 8) climbing stairs.

The RoboSimian robot from the Jet Propulsion Laboratory (JPL) had a unique form factor among the robots at the DRC and was developed with a philosophy that a passively stable, deliberate robot would be safer, more robust, and sufficiently fast for performing disaster response tasks. At the Finals, the second iteration of the RoboSimian series (named King Louie) displayed versatility and consistency in task execution over multiple days *without requiring resets*

Figure 1: King Louie, the second iteration in the RoboSimian series, egressing from the Polaris.

or interventions. The robot minimized risk, power consumption, and maximized execution speed by adapting its form to suit different tasks. The driving and egress tasks were achieved *without any modifications to the Polaris vehicle*. The four general-purpose limbs and hands were used to self-manipulate the robot out of the tight space within the Polaris roll cage via whole body motions (see Figure 1). The two active wheels on its body and two passive omni wheels on its limbs were used to transition into energy efficient sit postures which enabled faster mobility on flat terrain while remaining highly dexterous. Team RoboSimian achieved a 5th place finish by achieving 7 points in 47:59 minutes on Day 1. On Day 2, we achieved 6 points in 54:43 minutes. Both times were highly competitive in the competition. We started both runs with 91 V on the battery and ended at 86 V in an hour with 50% remaining operational time (operational range: 94 V full charge, 78 V cutoff). The stairs task was not attempted on both days due to insufficient testing, and on Day 2, the surprise task was abandoned due to lack of time and insufficient testing.

The focus of this paper is to summarize our approach taken towards system development and testing. System development for the DRC was about the balance between autonomy and supervision and it involved answering two key questions: first, *how much supervision is necessary*, and second, *how often is supervision required*. In the competition, we had some level of autonomy for short-order manipulation and mobility tasks but not at the level of performing multiple tasks at once. One guiding principle towards achieving semi-autonomy was to let the robot do what it is good at (repeatability, strength, precision) and let the operators do what they are good at (situational awareness, context, high-level reasoning). In this paper we present an architecture where the operator was responsible for situational awareness and task specification and the robot was responsible for its body movement. For this reason, the RoboSimian system was designed with body level adaptability and task level repeatability in mind.

The body level adaptability was limited to adaptation by proprioception alone (inertial measurements, tactile measurements in the wrist 6-axis force torque sensors). We use the term "behaviors" to conceptualize this low-level adaptability. Each behavior is a contact-triggered state machine; coupling them with whole body motion primitives enabled execution of short order manipulation and mobility tasks autonomously in body/odometry frame. We deliberately did not include any exteroceptive perception data for obstacle avoidance within our motion planners as this made our system brittle and unpredictable. The behaviors formed high-level command sequences for the operator to issue.

At a higher-level, we focused on a *teach-and-repeat* style of development by storing well tested behaviors and navigation poses in object/task frame for recall later. Using operator-aided object fitting to estimate a one shot task-to-body frame transform[1] enabled us to perform tasks with high repeatability on competition day while being robust to task

[1] Behaviors are executed in body frame but are stored in task frame for recall. When a task/object is fit in world frame, we can get the task-to-body

differences from practice to execution. We characterize the above hierarchical structure as *high-level repeatable and low-level adaptable* in this paper. Operator-aided object fitting was lightweight and only required a single stereo disparity image. However, the system was capable of using multiple stereo pairs on the robot to generate a local 3D map for situational awareness (on request only).

This paper is a follow-on from Hebert et al. (2015b), which outlined the system of the first robot in the RoboSimian series (named Clyde). Clyde was the entry for the 2014 DRC Trials and lacked fully functional tetherless operation. The main contribution of this paper is a field realization of the *high-level repeatable and low-level adaptable* paradigm within the DRC context. Other key contributions of this system include 1) a unique high-dimensional motion primitive architecture (see Section 5) that enabled highly repeatable motion planning in multi-limbed systems even when all or a subset of limbs are rigidly constrained to the environment (Figure 1), 2) a novel 3 degrees of freedom (DoF) gripper design (Cam-Hand) that has the ability to function as both a foot and a hand; the design is rated for very high grip strength under actuation (0.6kN) and under reaction (3kN) (see Section 3.2), 3) an operator-aided object fitting (see Section 6.3) method that relies on lightweight stereo disparity images (as opposed to point clouds). This approach generated reliable object fits given operator annotations and transmitted efficiently across a bandwidth-limited network.

This paper is structured as follows. Section 2 discusses related work and the relevance of the contributions made in this paper. Section 3 outlines the lessons learned and key hardware improvements made to the RoboSimian system since DRC Trials. Section 4 outlines the current software architecture by discussing the different software processes running within the system and how they were distributed across the network. Section 5 discusses the control system and the approach taken to address whole body motion planning. Sections 6 outlines the localization system, the mapping system and interactive object fitting for task specification. Section 7 discusses our management of message passing over a degraded wireless link during the competition. Section 8 discusses our approach and testing progress for each of the eight tasks leading up to and during the competition. This section outlines the different strategies that were attempted during testing and discusses what did and did not work. Section 9 presents results from DRC finals and prior testing in terms of battery performance, computing resource performance, task performance and network performance. Finally, Sections 10 and 11 summarize the lessons learned and provide concluding remarks.

2 Related work

As mentioned previously, the main contribution of this paper is a field realization of the *high-level repeatable and low-level adaptable* paradigm within the DRC context. We begin this section with background and motivation to provide context for the above paradigm against past work in robotics. Following this, we compare the three individual contributions in this paper to their respective state of the art 1) a high-dimensional motion primitive architecture, 2) a novel 3 DoF end-effector design (Cam-Hand) and 3) an operator-aided object fitting scheme on lightweight stereo disparity images.

2.1 Background and motivation for *high-level repeatability and low-level adapatability*

In this sub-section, we discuss related systems-level work in robotics both in support of and in contrast to the paradigm of *high-level repeatability and low-level adaptability*[2] that we emphasize in this paper. In the 2004/2005 DARPA Grand and 2007 Urban challenges that concentrated on autonomous driving (Buehler et al., 2007, 2009), the above-mentioned paradigm was flipped. In those autonomous driving systems, the low-level decision-making modules consisted of repeatable behaviors such as following pre-defined arcs/trajectories, whereas the higher level path/route planning modules were constantly adapting to changing situations (Thrun et al., 2006; Urmson et al., 2008).

In autonomous driving, the focus was to avoid contact (collisions). However, as DARPA challenges started to shift focus from mobility to manipulation with dexterous robots, deliberate interaction with contact became the goal. Sys-

transform based on current robot state in the world frame. The stored behaviors generate body frame constraints for motion planning.

[2] At DRC finals, repeatability was at the task level which was considered high level (e.g. turn the valve with consistent trial-by-trial execution) and adaptability was at the body contact level (e.g. adapt to changing forces as the robot interacts physically).

tems had to adapt at the contact level out of necessity. Autonomous dexterous manipulation was the showcase of the DARPA Autonomous Robotic Manipulation (ARM-S) software program. The ARM-S program focused on autonomous grasping and manipulation of objects with a bimanual robot (Hackett et al., 2014) that had no base mobility. JPL was one of the top performers under the ARM-S program (Hudson et al., 2012, 2014), where the notion of contact triggered behaviors took shape to enable *high-level repeatability and low-level adaptability*.

In the DRC series of challenges, the focus shifted from immobile autonomous manipulation in the ARM-S program to semi-autonomous mobile manipulation (Hebert et al., 2015b; Stentz et al., 2015; Johnson et al., 2015; Fallon et al., 2015). JPL has a rich history of past work in supervised autonomous robotic capability for remotely controlled scientific exploration on planetary surfaces (Hayati et al., 1997; Schenker et al., 2001; Backes et al., 2005; Matthies et al., 2007; Backes et al., 2010). Past experience guided our system choices to emphasize short-order autonomy as a necessary means to deal with communication latency. The system architecture of the RoboSimian system presented in this paper is consistent with space applications, where the goal is to enable high productivity in semi-autonomous science operations with minimal remote intervention.

Even though the individual tasks under DRC were challenging, the test environment itself was largely static and structured. Hence, adaptation at a higher level was not required. Task level strategies, once trained, did not need to change drastically. In addition, the operator formed the major source of the high-level decision making. If necessary, the operator could always take control at a lower level. Having access to a well tested, discrete dictionary of strategies enabled the operator to operate with confidence, which was critical to operational success. The teach-and-repeat style of development during testing is also consistent with development for space missions, where commands are composed of well tested capabilities that are developed on earth either prior to launch or on a surrogate robot on lab (Schenker et al., 2001).

The notion of adapting at the contact level is consistent with other robots fielded in the DRC. Most had a strong focus on active balancing (Koolen et al., 2013; Tedrake et al., 2014). JPL's use of contact behaviors was not focused on active balancing but instead on performing manipulation sequences with touch triggers and adapting by feel to uneven terrain while walking. Since the DRC Trials, the mentioned paradigm has been adapted to other robotic systems (Hebert et al., 2015a) developed under funding from the Defense Threat Reduction Agency (DTRA) and U.S. Army Research Lab (ARL).

2.2 Current state of art in motion-planning for high DoF systems and relevance of the presented motion primitive architecture

There are two relevant research clusters within the sub-field of motion planning for high DoF systems. The first is *trajectory search methods*. These methods focus on free-space end effector movement for manipulation on robots with arms connected to a fixed shoulder. The left and right arms are treated independently as there is no internal articulation in the shoulder. Experimentally, they have not been applied beyond kinematic chains with at most 6-9 DoFs (Ratliff et al., 2009; Kalakrishnan et al., 2011; Zucker et al., 2013). A quadruped robot was explored in (Ratliff et al., 2009) but body trajectories were decoupled from that of swing leg trajectories ensuring the kinematic chain requirement at all times. Full whole body articulation with end effector constraints has not been considered.

A second emerging research cluster involves *sequential constrained optimization methods* that tackle whole body mobility for bipeds and quadrupeds (closed kinematic trees with constraints) where some subset of end effectors may be constrained. Sequential constrained optimization methods don't reason about the trajectory as a whole; instead, iterative solves are needed to generate a trajectory and planning times can be slower in comparison. However, these approaches have been applied to very high dimensions of up-to 34 DoFs with end-effector constraints with near real time performance ($\sim$500Hz) (Kuindersma et al., 2014). The later approach has been more popular within the DRC program (Kuindersma et al., 2015; Koolen et al., 2013; Feng et al., 2015) where active balancing of biped systems and whole body planning has been a strong focus.

Our motion primitive architecture presented in Section 5 is consistent with the sequential constrained optimization cluster. The architecture uses a hierarchy of constrained optimizers and has been adapted for admittance control with

position controlled robots such as RoboSimian. RoboSimian has unique design elements such as symmetry, reconfigurability, omni-dexterous strength and internal articulation which are leveraged in our motion primitive architecture. The architecture enabled us to treat manipulation and mobility as an identical problem; Walking, posture transitions, free space limb motions, multi-limbed motions with rigidly constrained end effectors were all achieved with a single architecture without switching algorithms for each functionality. Compared to other teams at the DRC and to the state of art, our implementation is less focused on active balancing but more on *repeatability and range of tasks*. The repeatability can be seen in our repeated task performance results during and before the competition (discussed in Section 9). At the DRC Finals, RoboSimian demonstrated motions in 34 DoFs while using multiple limbs to grab the vehicle roll cage and exit the vehicle and was the only robot to do so without modifications to the Polaris vehicle.

2.3 Current state of art in hand designs and relevance of the Cam-Hand design

Commercially available end-effectors are generally designed for "hand" functionality and are aimed at manipulation tasks alone. RoboSimian's morphology placed unique requirements on the end-effector that required them to serve as both a "hand" and a "foot" for both walking and manipulation. To our knowledge, The only other alternative end-effector design that serves as both a "foot' and a "hand" was the old RoboSimian gripper presented in Hebert et al. (2015b). In addition, whole body motions such as those required during the egress task at DRC placed high grip strength requirements for the end-effector design. The scenario of whole body self-manipulation demands that a robot be able to support its entire weight by a single end effector. As such a key requirement for our application was high reactive grip strength as opposed to dexterity. RoboSimian weighs 134 Kg and required at least 1.3 kN of passive grip force to enable whole body motions. The Cam-Hand is rated for very high grip strength of 0.6 kN under actuation and 3 kN under reaction and serves the above requirements. A comparison of Cam-Hand's strength specifications with respect to a selected set of commercially-available electric hands (Robotiq, 2016b,a; Barrett, 2016) is summarized in Table 1.

	Gripper Weight	Grip Force (at Finger Tip)
Robotiq 3-Finger Adaptive Gripper	2.3 Kg	15 to 60 N
Robotiq 2-Finger Adaptive Gripper	0.9 Kg	20 to 235 N
BarrettHand^{TM}	0.98 Kg	15 N (active) 20 N (passive)
RoboSimian's Cam-Hand	3 Kg	600 N (active) 3000 N (passive)

Table 1: A comparison of Cam-Hand's strength specifications with respect to a selected set of commercially-available electric hands (Robotiq, 2016b,a; Barrett, 2016)

2.4 Current state of art in object recognition and relevance of the operator-aided fitting scheme

Object recognition and geometric fitting of known object models is a widely explored topic in the literature (Steder et al., 2009; Marton et al., 2010; Pang et al., 2015). It has generally been focused on 3D point clouds generated from laser range finders. Although automated object recognition can be beneficial in the DRC context, it is very difficult to eliminate false positives in real world conditions. Challenges include occlusions, viewpoint dependent data that differ from models and lighting variations. We operated with the assumption that the operator has much better judgment of context in real world conditions than algorithmic methods. Instead of focusing development resources on algorithm complexity, we focused on an intuitive interface for the operator to specify/re-specify intent to the robot in a lightweight manner.

Typical approaches at the DRC involved transmitting lidar based point clouds or voxelized representations of a region of interest to the operator side for operator-guided object fitting (Stentz et al., 2015; Johnson et al., 2015; Fallon et al., 2015). This presents a tradeoff between resolution and point cloud message size and often results in lossy compression. Our approach discussed in Section 6 relies only on stereo disparity images (or range images generated from lidar) that can be compressed in a *loss-less* manner and transmitted very efficiently. Instead of lidar, the RoboSimian robot has seven stereo pairs (six used in competition) embedded in its torso that give a rich panoramic snapshot of the local surrounding. The long range lidar (velodyne) is primarily used for 3D scan matching for localization and the point

clouds are never transmitted to the operator side. We used annotations on a 2D image that enabled the operator rapidly specify orientation and position constraints for object fitting. This lightweight approach has value beyond the DRC as it generalizes to simpler user interfaces such as a touch tablet or a smartphone.

3 Hardware improvements

RoboSimian weighs 134 kg and consists of four general-purpose limbs with seven degrees of freedom (DoF) per limb. It has two active wheels on its body and two passive omni wheels. All seven joints in the limbs and the two active wheels are actuated by the same actuator design. The actuator consists of a frameless DC brushless motor that directly drives a 160:1 harmonic drive, with the output supported by a crossed-roller bearing. Each actuator is rated to produce a peak torque of 580 Nm and a peak speed of 3.4 rad/s. A power-on-to-disengage magnetic safety brake is included on the motor rotor and is able to hold a torque at the output side. When the robot is not being commanded to move, the brakes are engaged and no power is consumed by the motors from the battery. This enables the robot to be highly energy efficient in a DRC style event where there are often periods of long pauses during execution. A component overview of the RoboSimian system is shown in Figure 2a.

One of the high level goals in the system was to bridge the gap between mobility and manipulation by treating mobility as a self-manipulation problem. The RoboSimian platform was designed to be highly reconfigurable and is well suited for whole body maneuvers in complex 3D environments. It has the ability to grasp its environment with all four limbs. Since each limb consists of seven identical actuators, there are no weak joints at the end of each limb, and this enables the robot to take the same amount of load in its wrist as it would in its shoulder. When closing the kinematic loop with the world, this omni-dexterous strength from shoulder to wrist and across limbs is highly desirable.

The high maneuverability of the robot was demonstrated in the egress task at the competition, where we performed the task without making any modifications to the Polaris vehicle and leveraging whole body maneuvers by grabbing the roll cage (see Figure 1). The robot could walk on and grasp with the new Cam-Hand end effectors, which were designed to retract the fingers while walking (see Section 3.2) . In the competition, only two actuated hands were used on the front two limbs in order to have a set of spares for hardware maintenance. The limb link lengths($0.24m$) were chosen to enable ease of transport in a Pelican^{TM} 0550 transport case (see Figure 2c; six 0550 Cases fits to a 463L military air cargo pallet standard) (Pelican, 2016).

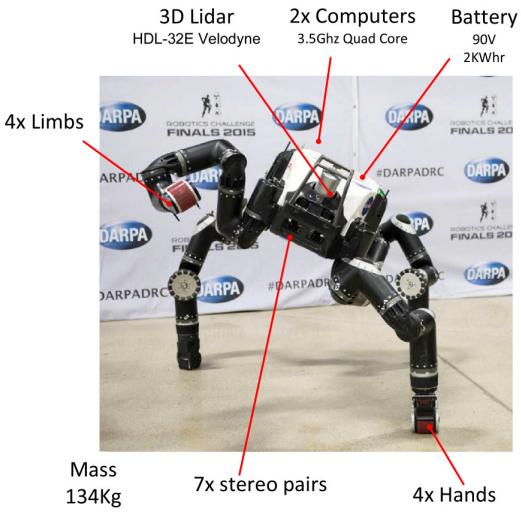
This section outlines the two key hardware improvements made to the RoboSimian system since the DRC Trials. The first improvement was a chassis redesign to incorporate a 2 kWhr battery for tetherless operation. The second was a complete redesign of the hands since the DRC Trials. No major design improvements were made to the limbs and the actuators since the Trials. Further details about the limbs and actuators are discussed in Hebert et al. (2015b) .

3.1 Chassis redesign for tetherless operation

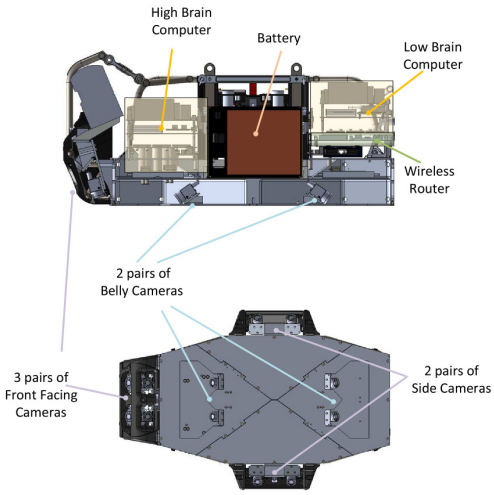
The chassis from the Trials had to be redesigned to incorporate the battery subsystem, a wireless router and a wireless emergency stop for tetherless operation. The chassis hosts the power electronics, e-stop receiver, a 2 kWhr battery, two 3.5GHz quad-core computers, seven pairs of stereo cameras, an HDL-32 Velodyne lidar scanner, and a wireless router. The new chassis packaging and the layout of the cameras is shown in Figure 2b. The robot uses multiple stereo cameras to achieve nearly 360° passive three-dimensional sensing. The lidar scanner was a new addition to the sensing suite since the Trials. The main purpose of the lidar was not for mapping but for lidar based odometry with incremental scan matching (discussed in Section 5). The cameras, laser scanner, and IMU are all synchronized to a common time frame by a microcontroller system that triggers the cameras at 10 Hz based on a pulse-per-second (PPS) signal from the Velodyne lidar scanner.

The battery is composed of lithium cells used in electric motorcycles to achieve high peak power to weight ratios. The final design weighed 15.5 kg. It had a capacity of 1.92 kWhr with continuous power of 1.9 kW, peak power of 47 kW, and a nominal pack voltage of 90 VDC. It comprised of four lithium modules (24S4P) constructed for JPL by

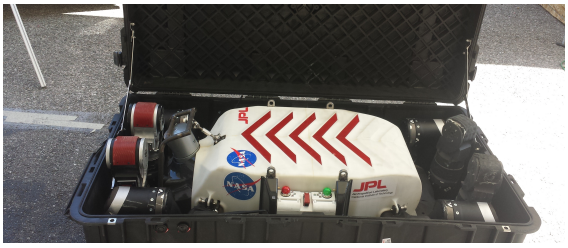
Lithiumstart LLC. The battery package consisted of an onboard battery management system (BMS), current sensor and a fuse. BMS reported the state of the battery including voltage, current, temperature, and cell status via messages on a CAN bus. The battery charger can be connected directly to the pack or on a shared DC bus with the RoboSimian load.



(a)



(b) Cross section and bottom view of the new chassis . The chassis is made out of aluminum and weighs 42 Kg with components. Its outter bounding dimensions were roughly 0.93 m(L) x 0.5 m(W) x 0.28 m(H).



(c) RoboSimian was designed to fit into a standardized military transport case for ease of deployment in a disaster scenario (Pelican, 2016). The interior dimensions of the case are 1.21m (L) X 0.61m (W) X 0.45m (H) .

Figure 2: Hardware component description and new chassis packaging. The chassis was redesigned to incorporate the battery subsystem, a wireless router and a wireless emergency stop for tetherless operation. In addition, a lidar sensor and additional stereo pairs were added to the chassis for 360 coverage of stereo mapping and lidar based localization.

3.2 Hand redesign

The previous hand design that was used in the Trials had high finger dexterity, and each finger was tendon driven (see Figure 3a). For most tasks in the DRC, dexterity was needed at the wrist level instead of the fingers. Finger-level finesse in manipulation was rarely required, except for triggering the drill, which is just a one DoF operation. The hand performance during the DRC Trials was deemed inadequate for the scale of manipulation tasks we were focused on for the DRC Finals. These inadequacies included 1) low actuated grip strength despite increasing the gear ratio, 2) low reacted finger strength (tendons could easily snap), 3) tendons snapping while walking on rough surfaces, 4) objects shifting in grip during hand movements and during interactions with the world, 5) inability to grip 2x4's on the 4" faces, 6) inability to pick up cinder blocks, and 7) the difficulty and tediousness of repairing broken tendons.

Based on the inadequacies of the previous hand design, we devised a new set of desired capabilities with a high-level

aim of achieving strength over finger dexterity. In the end, we converged on the Cam-Hand design, which has three degrees of freedom (DoFs) with four fingers. The outer two fingers are slaved and the inner two are independent (see Figure 3a). All fingers have continuous rotation capability and together with the 3 DoFs can generate many different configurations as shown in Figure 3c. The Cam-Hand is characterized by a number of non-obvious extensions of the state of the art in gripper design. The new guiding design principles included 1) the ability to use the gripper as both a "foot" and a "hand", with high grip strength under actuation and under reaction, 2) the use of both the inside and outside surfaces of the fingers for grasping, 3) continuous rotation of the fingers about the palm to both enable the use of both sides of finger and to act as a wrist joint if all fingers are moved together relative to the palm, 4) accommodation of multi-functional fingers that can be used for various types of grasping, 5) easy repair, and 6) interchangeable finger design.

The name Cam-Hand was coined as the hand was inspired by spring loaded camming devices used in rock climbing. When lodged within cracks, they convert the pulling force along the stem of the unit into outwards pressure on the rock and aid in climbing. The camming function of the hand can be seen in Figure 3d, where the hand is able to grip a cinder block from the inside via insertion and outward actuation to jam it from within.

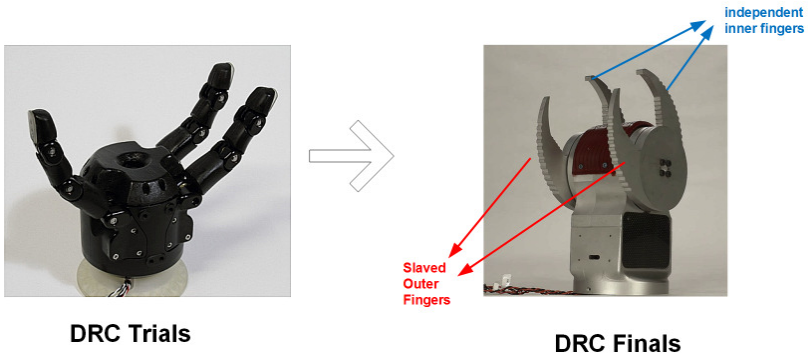
The final design of the hand weighed 3 kg and had three brushed DC motors with a planetary and spiroid gear train for a total gear ratio of 249:1. It had a grip strength of 609 N at the tip of each finger when the motors were stalled. With an applied tip force of 609 N, the hand is rated to exert $\approx$ 62 kg of load per finger via actuation. This actuated strength is sufficient to grip a concrete block on two faces and lift it (Figure 3d) or pierce through half inch drywall. In addition to actuated grip strength, the reacted tip force for each finger was around 3 kN. This parameter becomes significant when the hand is not actuating and taking a load (i.e. it would take 3 kN of force on a single fingertip to break a gear tooth - see Figure 3b). Each finger on the Cam-Hand is rated to support the entire weight of the robot individually as long as the aluminum tip does not break or deform. In practice, however, it is safer to evenly distribute the load across fingers and to use multiple grasps when possible. For further details on the Cam-Hand design the reader is referred to (Kennedy and Carpenter, 2016; Shekels et al., 2016).

4 Software architecture

This section briefly outlines the software architecture by discussing the different software processes running within the system and how they were distributed across the network. Figure 4 shows all of the software processes that were running within the system and their respective host computers. The system consisted of three computers on the robot side and one remote computer on the operator side. On the robot side, there was a low-brain control computer, a high-brain perception computer, and a field computer.

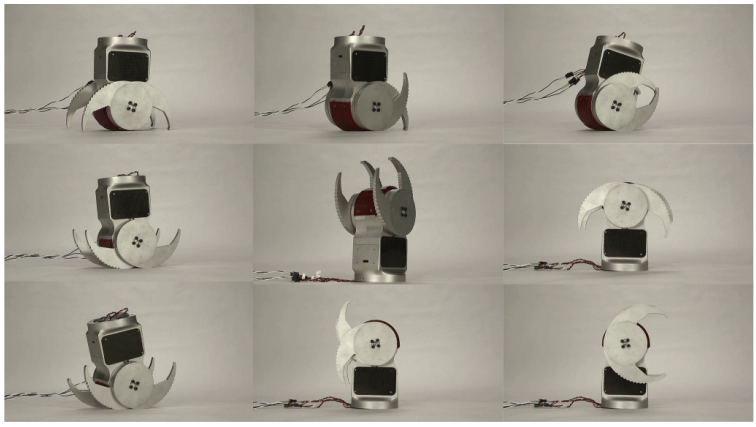
All computers were running the Ubuntu 12.04 Linux distribution. The low-brain control computer had a low-latency kernel for near real-time communications to the limbs via the EtherCAT communication protocol. The low-brain computer was the main receiver for plan requests and action commands that triggered whole-body motion planning and control. At the lowest level, there were four limb servers (each managing a chain of 7 actuators) and a differential drive server (managing 2 actuators) that were running at real-time priority. A control server mediated between the planners (mobility and manipulation) and the limb servers that required actuator level commands. The control server translated the whole-body plans into clusters of actuator level commands over time. In addition, it also sequenced force-control behaviors by closing the loop with limb-level replanning given force torque measurements in the wrist. The low-brain also had an independent battery management process that monitored the state of the battery and published messages directly to the operator interface.

The high-brain perception computer had two processes that handled the lidar and camera pipelines. The lidar process handled scan acquisition, post-processing, and 3D registration for lidar odometry. It served as a virtual sensor that provided pose deltas into a state estimator. The camera module was the workhorse of the perception system and was the primary source of imagery and maps. It handled stereo image acquisition for 7 stereo pairs at 10 Hz, generated dense stereo maps, and performed visual odometry. It also served as a state estimator which fused visual odometry with lidar odometry in an extended Kalman filter (EKF) implementation.

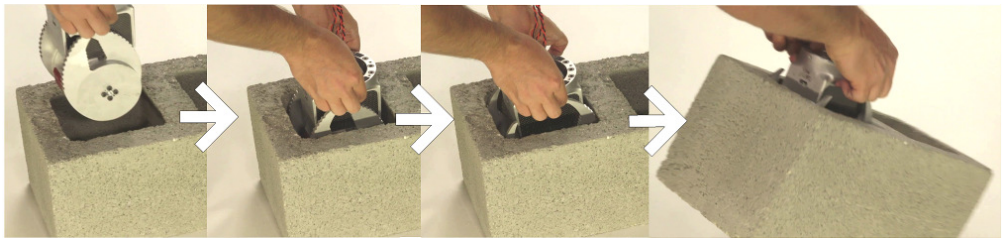


(a) Tendon-driven hand used during the DRC Trials

(b) Reacted grasp strength: one fingertip force is rated to be 3kN



(c) 3 DoFs: four fingers with one slave pair and two independent. All fingers have continuous rotation capability.



(d) Cam mode: applied one fingertip force during actuation is rated to be 609 N

Figure 3: Cam-Hand: the new hand design for the DRC Finals. The new hand was designed with a focus on strength over finger dexterity. It functions as both a "foot' and a "hand" and is rated for very high grip strength of 0.6 kN under actuation and 3 kN under reaction

Figure 5 shows the high-level communications between different computers within the network architecture via UDP or TCP packets. Further details regarding the choice of communication protocols is discussed in Section 7. All messages in the system were routed through two network managers. The robot side network manager ran on the field computer, and the operator side network manager ran on the remote computer. The network managers served as the main regulator for network traffic over the wireless links and performed message compression, decompression, network health monitoring, and associated bandwidth management.

Finally, the remote computer consisted of the Operator Control Unit (OCU), which served as the primary perceptual information hub and human robot interaction interface. Mirrors of mobility and manipulation planners were also run on the operator side.[3] These processes were identical to the ones running on the control computer and provided

[3]We made a special effort to design the mobility and manipulation planners so that the processes on the robot and OCU side would never be

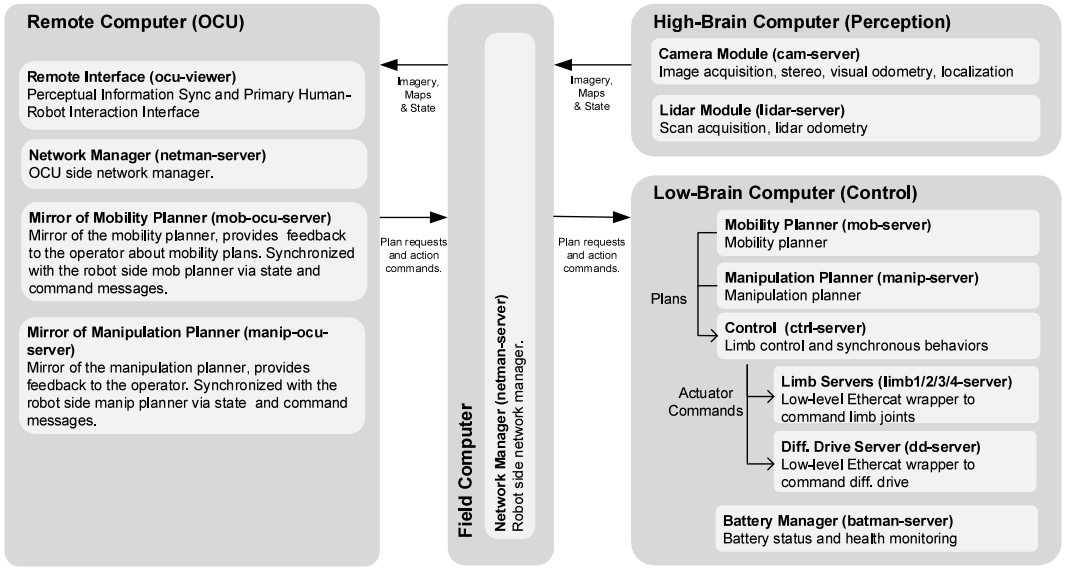


Figure 4: The RoboSimian software architecture. Each light-gray block represents a separate software module/process. Each dark-gray block represents which machine these processes run on.

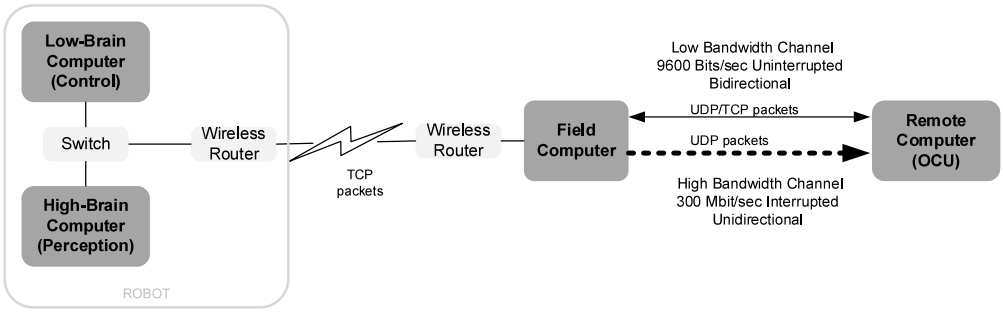


Figure 5: High-level communications between different computers within the network architecture via UDP or TCP packets. Further details regarding the choice of communication protocols is discussed in Section 7.

instant feedback about whole body plans. This approach diminished the need to transmit large plan messages over the network.

5 Planning and control

This section discusses the features and design choices of the planning and control subsystems to support the "low-level adaptability and high-level repeatability" paradigm that is outlined in this paper. This section begins with an introduction to contact behaviors which provide the base framework to enable "low-level adaptability" in task execution by reacting to contact forces at the end effector. The adaptability framework is largely confined to proprioceptive feedback such as end effector force measurements, IMU orientation, and wheel odometry. We deliberately avoided using exteroceptive perception data from stereo and lidar for obstacle avoidance within our motion planners as this made our system brittle and unpredictable.

unsynchronized. For example, the processes transmit plan requests and action requests over TCP to guarantee delivery over the wireless links and plan with the compressed robot state in order to be numerically consistent with the worst case state estimate on the remote side.

5.1 Behaviors

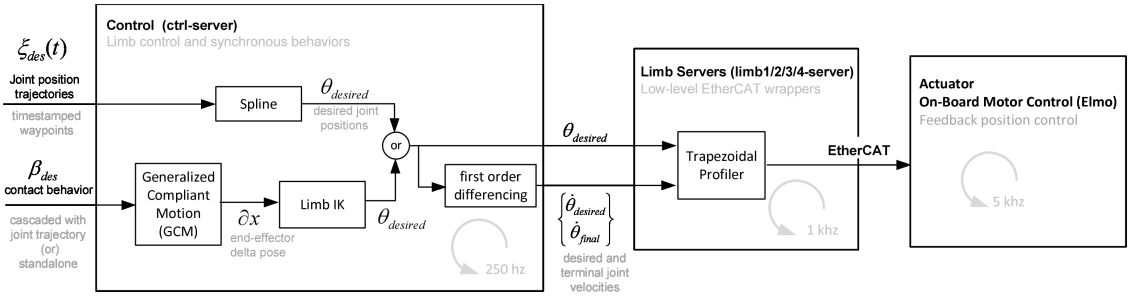


Figure 6: Control information flow between trajectory specification/following and contact-behaviors. Behaviors specify Cartesian-space motion objectives at the end effector and consist of asynchronous, contact-triggered, hierarchical state machines.

The control system on RoboSimian is a series of synchronous, cascaded loops running at different update rates. Behaviors consist of asynchronous, contact-triggered, hierarchical state machines that interface to this control system at its highest loop. The control system and behaviors are responsible for all motion execution on the RoboSimian platform.

A block diagram of this control system is given in Figure 6. Each limb has its own individual limb server process. The limb server interprets synchronous motion requests from the control module and interpolates these into desired position setpoints that get sent to the controllers via EtherCAT. Asynchronous inputs to the control module can be divided into joint position trajectories for specifying open-loop motion and parameterized behavior requests which consist of Cartesian motion objectives at the end effector. Both types of inputs can be received simultaneously in a cascaded fashion; in this situation the control module will first execute the open-loop position trajectory and then the closed-loop contact behavior. This allows for intuitive sequencing of free-space planning to get the end effector to a desired starting state and then running a contact behavior once this state has been reached.

Behaviors specify Cartesian-space motion objectives at the end effector and are data-driven hierarchical state machines. Force control set points and open-loop Cartesian moves are examples of the types of objectives that can be associated with a given behavior. Each state consists of a parameterized action, an associated set of end condition checks for monitoring action completion, and logic for transitioning. Example end condition checks are motion request complete, time reached, and measured force/torque at a certain value. Transitions are event-based and occur when an action completes or fails. The state machines for a set of behaviors used in the DRC Finals is given in Figure 7. Control uses Generalized Compliant Motion (GCM) (Backes, 1991) to translate these objectives to joint space. GCM accomplishes this by converting each objective into a perturbation in the end effector frame, then merges these perturbations together to create a unified desired transform. Control passes this transform into a Jacobian-based inverse kinematics solver to compute the desired joint angles, which are sent to the limb servers. For object manipulation behaviors (e.g. Door Open, Valve Turn, Drill Grab), part of the behavior definition is the starting pose of the end effector in the object frame. When a planner receives a behavior request from the OCU, it first plans a trajectory to this end effector pose based on the location of the object in the 3D world. It then sends the trajectory, behavior type, and behavior parameters to the control server to handle execution. From the perspective of the behavior, this ensures that the initial condition of the end effector with respect to the object is always the same.

5.2 Whole-body motion planning with motion primitives

This section discusses the tools used and the approach taken to address whole body motion planning on the RoboSimian platform. While the contact behaviors discussed in the previous section enabled low-level adaptability to force feedback, they also enabled high-level repeatability as they were defined relative to an object. Given accurate object fitting (which is discussed in Section 6.3) end effector motion objectives within our behavior framework were issued with high repeatability and little variance. In contrast, operator-reliant specification of end effector poses can

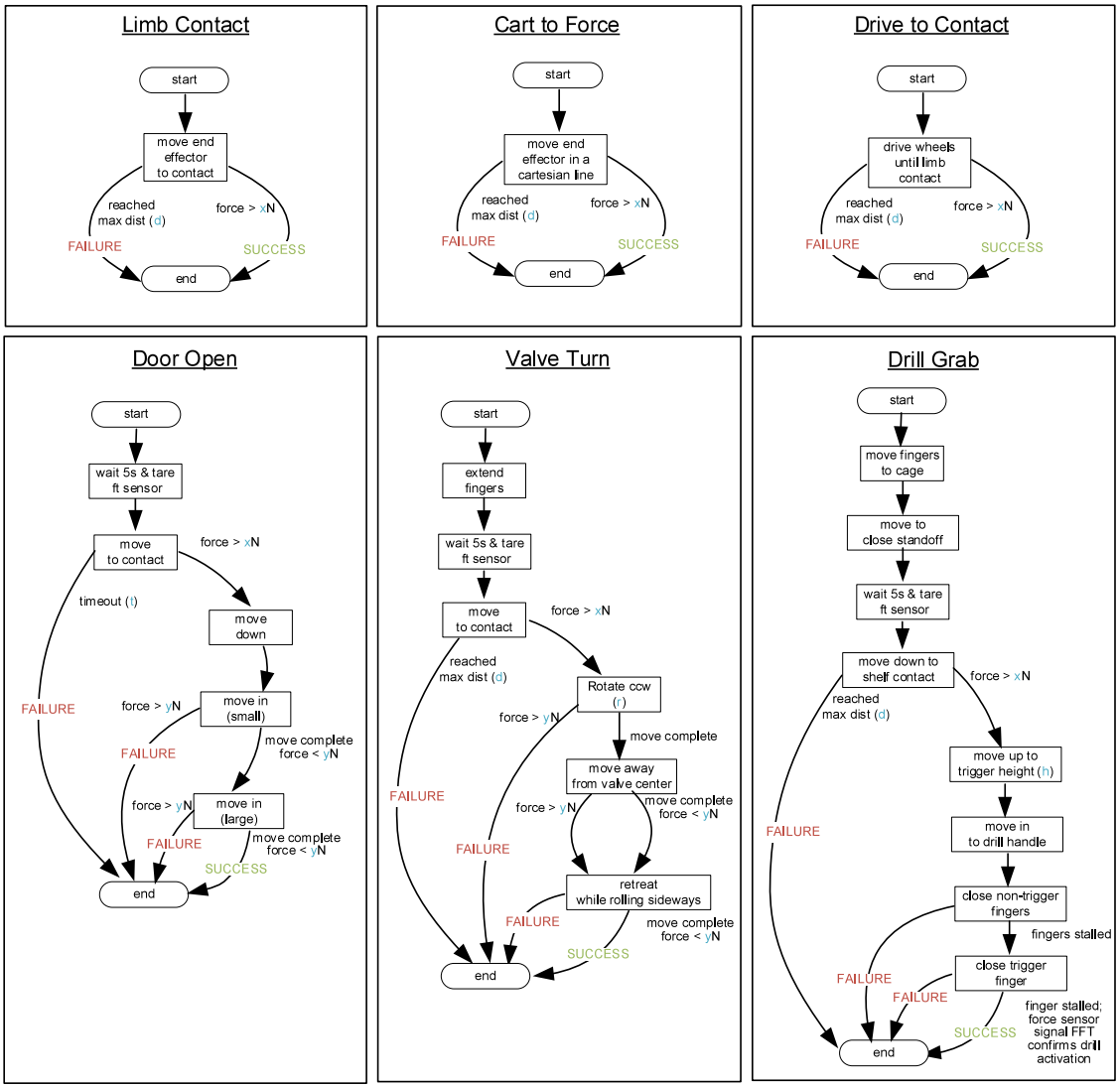


Figure 7: A subset of the behaviors (contact-triggered state machines) used in the competition. Most behaviors were parameterized by force/distance thresholds and timeouts (the parameters are shown in blue; best viewed in color).

be highly variable and can lead to mixed results. Even though the system had the ability to bypass object fitting to allow an operator to directly specify end effector goals, we avoided specification of egocentric commands as much as possible for repeatability reasons.

A block diagram of the interaction of different planners with the operator interface and control subsystem is shown in Figure 8. The core contribution of the RoboSimian motion planning subsystem is a high-dimensional motion primitive architecture that enabled highly repeatable whole body motions in dimensions up to 41 DoF. The motion primitive architecture generalizes motion planning to kinematic systems beyond kinematic chains and trees where closed loops exist. This was the case for RoboSimian when it had to perform walking, egress and various posture transitions (stand to sit, sit to stand) where some or all of the limbs were rigidly constrained to the environment. The components of this architecture, as illustrated in Figure 8, include a pre-trained motion primitive library, an online trajectory generation scheme, and an online primitive adjustment methodology that accounts for mismatches from plan to execution due to end effector behaviors. In addition to the motion primitive architecture, we also used a bidirectional RRTc planner (Kuffner and LaValle, 2000), which was sufficient for simple single limb manipulation tasks (7 DoF kinematic chain) and provided good performance.

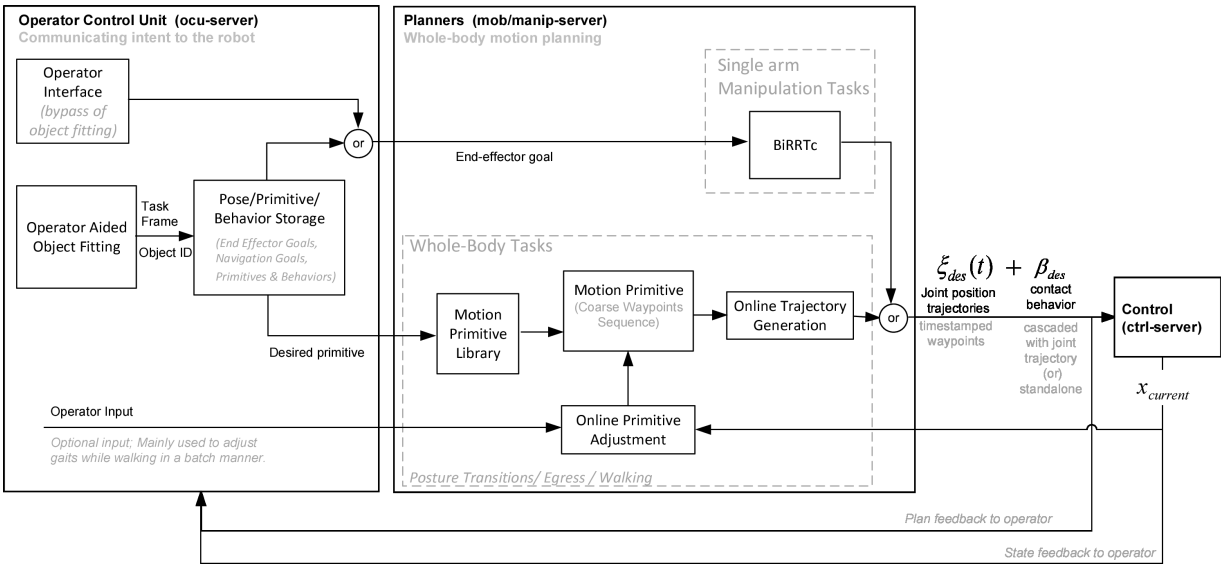


Figure 8: Information flow between planners, operator interface and control modules. Various components within the planning architecture are shown that include a pre-trained motion primitive library, an online trajectory generation scheme, and an online primitive adjustment methodology that accounts for mismatches from plan to execution due to end effector behaviors.

5.2.1 Terminology

The terminology below is used in the following subsections. We explicitly define them here for clarity:

. **posture** - a specific set of joint positions of the four limbs (7 joints x 4 limbs = 28 DoF)[4]

. **state** - the current robot posture and body pose specified in world frame.

. **waypoint** - a desired posture and the body pose[5] specified in a reference task frame (e.g. object frame).

. **motion primitive** - a sequence of coarse waypoints. Coarseness implies resolution of task specification i.e. one can think of them as key motion snapshots for a given task.

. **trajectory** - a sequence of fine waypoints. A trajectory fully specifies the task or a segment of a task.

5.2.2 Motivation and methodology

Each motion primitive is a sequence of waypoints that defines the task in a coarse manner, and each waypoint represents a desired state of the system (joint angles and body pose defined in a task frame). Planning starts and ends in the prescribed waypoints. Some waypoints terminated in contact behaviors which made them adaptable during execution. Adaptable waypoints triggered online replanning (shown as primitive adjustment in Figure 8) as a means to update current and subsequent waypoints on-the-fly. The coarse waypoints in a primitive were further grouped into *clusters*. At the end of each cluster, operator confirmation was required to proceed forward. During task development, no explicit coding was necessary in sequencing motion planning tools for different tasks; all the information required to do the task is captured in the specification of the coarse waypoints. The information regarding the waypoints, terminal behaviors and clusters were stored in text files which could be easily edited without recompiling code.

A motion primitive centric architecture was chosen for two key reasons: repeatability and nonparametric development. Posing the search as piecewise boundary value problems (BVP) (pairs of fully specified start and end states)

[4]Does not include the body pose

[5]Body pose is specified in 6 dimensions; a 3D position vector and a 3D rotation vector in angle-scaled axis form

outperformed a global initial value problem (IVP) specification in terms of repeatability.[6] This led to a decision to separate inverse kinematics (a search for boundaries) from trajectory generation (moving between the boundaries). On the subject of nonparametric development, conscious design choices were made to avoid unit-less tuning parameters in planning algorithms. All the information required for a given task was specified in terms of the task level coarse waypoints. In addition, any parameters needed to formulate the search problem had to have physical units associated with them. Some example parameters include: required end effector tolerance in IK (mm) and resolution tolerance in trajectory generation (degrees). This design choice allowed us to avoid the pitfall of circular development, where subsequent tuning and improvements to algorithms for new behaviors broke previously developed behaviors. Leading up to the competition, the notion of avoiding circular development was a key lesson learned. It was necessary to achieve effective task development during testing.

Offline generation of motion primitives: For tasks such as walking, where simple topologies of the terrain were available, primitives were auto-generated within an optimization routine. For tasks such as egress, the specification was a lot more guided with manual specification of candidate end effector locations that leveraged the intuition of the developer. Figure 9 shows interactive specification of end effector and body pose constraints in order to sequence waypoints for the egress task. In addition, we used a teach-and-repeat strategy to execute and tweak the waypoints during testing. The limb positions were finely adjusted interactively by using force control on the limbs. This made us robust to imperfect models of the robot and vehicle and unforeseen hardware/dynamic issues that are hard to account for within kinematic search of waypoints in simulation.

The following section outlines the different search problems that were used to generate motion primitives offline and generate trajectories online. Two key search problems were addressed using constrained optimization via nonlinear programming: numerical inverse kinematics addresses the question of *where and how should the robot's body be to achieve a task*, and online trajectory generation addresses the question of *how to move the body without violating constraints*.

Waypoint search via constrained nonlinear optimization: The following problem formulation (Equation 1) searches for a feasible robot body pose and posture given a desired end effector pose and/or camera gaze constraint.[7] The search is posed as a nonlinear programming problem and is solved using an open source nonlinear optimization package (IPOPT) based on the interior point method (Wächter and Biegler, 2006).

$$
\begin{aligned}
\{x^*_{torso}, \theta^*\} \quad & = \underset{x_{torso},\theta}{\arg\min} \left\{ (\theta - \theta_{nom})^T (\theta - \theta_{nom}) \right\} \qquad (1)\\
\text{s.t.} \quad & LB \leq c_{pos}(fk_{ee_i}(x_{torso}, \theta), x_{ee_i\_des}) \leq UB \mid i \in \{1,2,3,4\} \\
& LB \leq c_{orient}(fk_{ee_i}(x_{torso}, \theta), x_{ee_i\_des}) \leq UB \\
& LB \leq c_{gaze}(fk_{cam}(x_{torso}, \theta), x_{gaze\_target}) \leq UB \\
& LB \leq c_{margin}(fk_{com}(x_{torso}, \theta), sp(x_{torso}, \theta)) \leq UB
\end{aligned}
$$

where θ is joint angles, $x_{torso} \in SE(3)$ is the body-pose in world frame, θ_{nom} is a nominal posture used to bias the search, and $fk_{ee/cam/com}$ functions represent the end effector, camera and center of mass (com) forward kinematics. sp represents the support polygon formed by a set of three fixed feet. $x_{ee_i\_des} \in SE(3)$ represents the desired poses of the end effector for all limbs ($i \in \{1,2,3,4\}$). x^*_{torso} is the optimized body-pose in world frame, and θ^* represent the optimized joint angles. $c_{pos}()$[8], $c_{orient}()$[9], $c_{gaze}()$[10], $c_{margin}()$[11], are position, orientation, camera gaze and support

[6] By bounding the robot start and end states, one is able to avoid kinematic drift which can occur to due minor variations in the input parameters across multiple executions of a planner.

[7] This enforces that the manipulation hand is in a conic field of view.

[8] $c_{pos}()$ is represented by the euclidean distance metric.

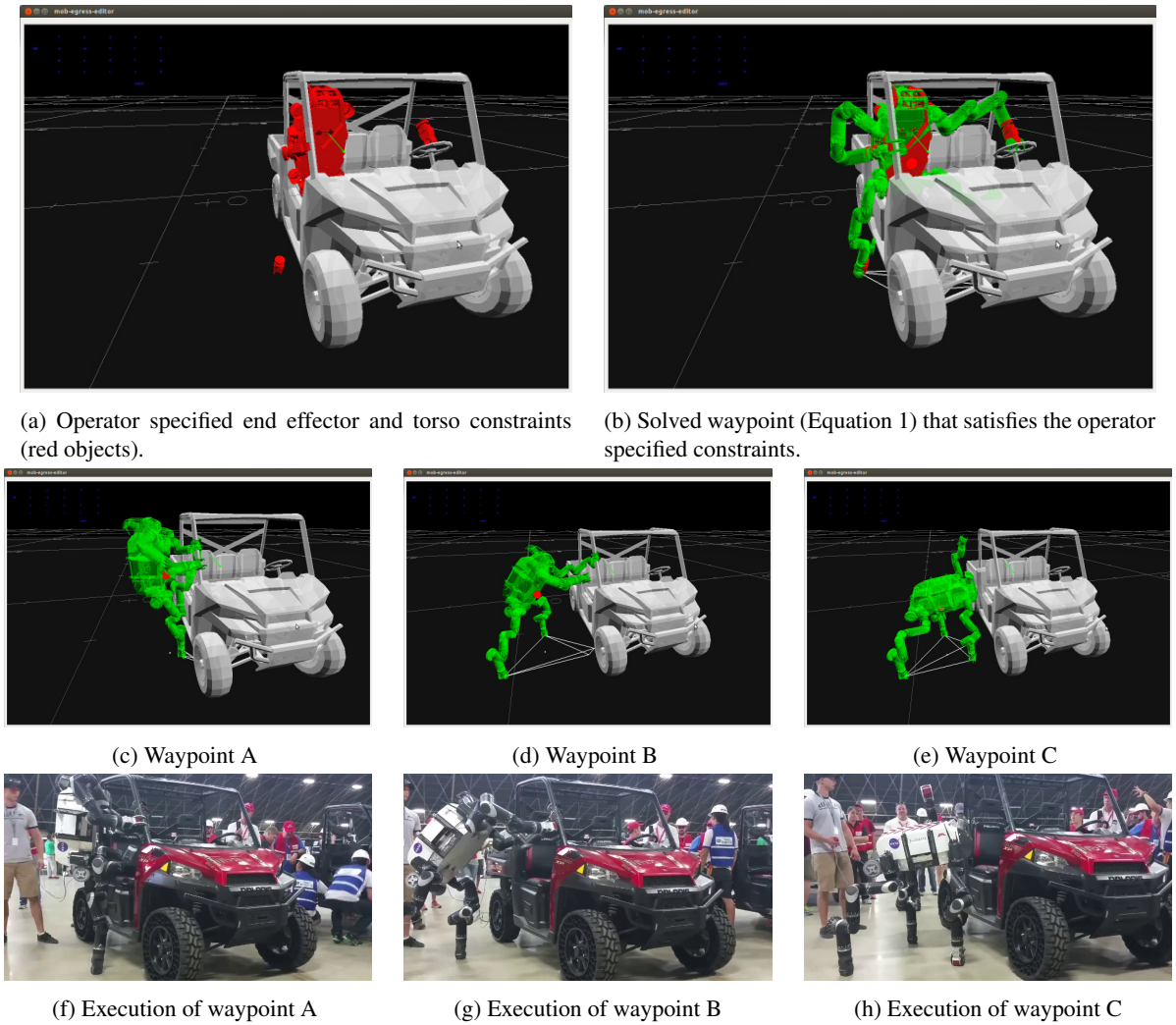
[9] $c_{orient}()$ is represented by $cos(tol) - 1 \leq 0.5 * Trace((R^W_1)^T R^W_2) - 1 \leq 0$ where R^W_1 and R^W_2 are two rotation matrices.

[10] $c_{gaze}()$ is represented by $cos(tol) - 1 \leq (\frac{p_{tar} - p_{cam}}{\|p_{tar} - p_{cam}\|})^T (R^W_{cam} u^{cam}_{axis}) - 1 \leq 0$ where p_{tar} is position of the gaze target, p_{cam} is position of the camera, R^W_{cam} is the orientation of the camera in world frame and u^{cam}_{axis} is a desired gaze axis in camera frame.

[11] $c_{margin}()$ is represented by $-\inf \leq \{-sign(n^T(p_{ee_3} - p_{ee_1})) * (n^T(p_{com} - p_{ee_2})) + margin\} \leq 0$, where $n = \frac{(p_{ee_1} - p_{ee_2}) \times n_{support\_plane}}{\|(p_{ee_1} - p_{ee_2}) \times n_{support\_plane}\|}$; p_{ee_1}, p_{ee_2}

margin constraint functions.

Each nonlinear solve took about 50 ms to generate waypoints with millimeter tolerance for the desired end effector positions and one degree tolerance for desired end effector orientation. The choice of the nominal posture (θ_{nom}) was task dependent and was essential for localizing the search problem. For walking, the nominal walk posture was used for all waypoints. For egress and posture transitions, the nominal posture for subsequent IK solves was set to the previously solved coarse waypoint to maintain kinematic consistency while allowing major postural changes.



(a) Operator specified end effector and torso constraints (red objects).

(b) Solved waypoint (Equation 1) that satisfies the operator specified constraints.

(c) Waypoint A

(d) Waypoint B

(e) Waypoint C

(f) Execution of waypoint A

(g) Execution of waypoint B

(h) Execution of waypoint C

Figure 9: Generating waypoints for egress via nonlinear waypoint search (see Section 5.2.2). A sample set of waypoints (A,B,C) from the egress sequence and their subsequent execution is shown in subfigures c-h (best viewed in color).

Augmented waypoint search: The augmented waypoint search addresses the problem of searching for waypoints over time. The current implementation of augmented search simplifies the problem by assuming that the torso does not move when the limb is moving. The gait pattern in walking follows a {torso move, limb move} cycle where the torso moves with all feet fixed and then a selected limb moves. Given a desired gait pattern and desired end effector locations, we generate waypoints for a gait via four augmented waypoint searches. Each waypoint includes the current

and p_{ee_3} are positions of the end effectors on the boundary of the support polygon, p_{ee_3} is the furthest end effector from the limb that is moving, p_{com} is the position of the center of mass (COM), and finally *margin* specifies a fixed tolerance in the directional distance of the COM from the active boundary of the support polygon.

posture of all the limbs and the future configuration of the step limb (which is treated as a phantom fifth limb within the solver). This approach provides one-step look ahead in generating waypoints for walking. The search generates two coarse waypoints that are required within a single {torso move, limb move} event of the gait cycle. The problem is formulated in a similar manner to waypoint search as described previously in Equation 1. The only exception is that two support margin constraints, which consider the torso move and future limb move postures, are required.

Online trajectory generation via recursive constrained nonlinear optimization: Given a set of coarse waypoints, the role of trajectory generation is to generate fine waypoints with a specified resolution by recursively running the optimization procedure outlined below in Equation 2.

$$
\begin{aligned}
\{x^*_{torso}, \theta^*\} \quad & = \underset{x_{torso},\theta}{\arg\min} \left\{ (\theta - \theta_{goal})^T (\theta - \theta_{goal}) \right\} \qquad (2) \\
\text{s.t.} \quad & LB \leq c_{res}(\theta, \theta_{prev}) \leq UB \\
& LB \leq c_{pos}(fk_{ee_i}(x_{torso}, \theta), x_{ee_i\text{-}fixed}) \leq UB \mid i \in \{1,2,3,4\} \\
& LB \leq c_{orient}(fk_{ee_i}(x_{torso}, \theta), x_{ee_i\text{-}fixed}) \leq UB \\
& LB \leq c_{margin}(fk_{com}(x_{torso}, \theta), sp(x_{torso}, \theta)) \leq UB
\end{aligned}
$$

where θ is joint angles, $x_{torso} \in SE(3)$ is the body-pose in world frame, θ_{goal} is the goal posture that the trajectory search is seeking, and $fk_{ee/cam/com}$ functions represent the end effector, camera, and center of mass (com) forward kinematics. sp represents the support polygon formed by a set of three fixed feet. $x_{ee_i\text{-}fixed} \in SE(3)$ represent the selected end effectors that are not moving ($i \in \{1,2,3,4\}$). x^*_{torso} is the optimized body-pose in world frame, and θ^* represent the optimized joint angles. $c_{res}()$[12], $c_{pos}()$, $c_{orient}()$, $c_{margin}()$, are resolution, position, orientation and support margin constraint functions.

The purpose is to find a kinematically feasible sequence that starts and ends in the specified coarse waypoints without violating constraints such as keeping the feet fixed and maintaining the center of mass in the support polygon (quasi-static stability). The trajectory generation is run both forwards and backwards, and the first procedure to generate a successful path is used. The speed of trajectory generation was directly proportional to the desired resolution, which enabled us to control the planning speed on a task by task level (e.g. tighter resolution was used for torso moves, and the resolution was relaxed for single limb moves where spline interpolation performs well). In order to ensure our online trajectory generation was on the order of a few seconds, we needed to generate coarse waypoints with a resolution of roughly 50 degrees between the start and end waypoints in a cluster. In the competition, trajectory generation for each waypoint cluster (a set of 2-10 waypoints) took about a few secs; we used a resolution of 5 degrees (2-norm) for a good balance between speed and kinematic consistency of constraints. On average, up to 10 fine waypoints were generated given a set of coarse waypoints in a cluster, and each nonlinear solve took about 50 ms. The fine waypoints generated via the recursive search were further resampled via joint-level splines and limb-level Jacobian pseudo-inverse corrections (Buss, 2004) for improved resolution and accuracy.

Online adjustment of motion primitives: For the egress task and posture transitions the set of coarse waypoints in the motion primitive were fixed. For walking, a subset of waypoints terminated in a detect contact behavior (see Section 5.1) to account for uncertainty in the environment, which made them adaptable. The contact behavior was appended to the end of swing trajectories to ensure safe touchdowns. Under this behavior, the swing limb stops a distance Δz above the desired stance location and slowly lowers until contact is detected through force feedback. If there is any error in the z location, future waypoints in the primitive that expect the limb to be in stance at the planned location need to be updated to reflect the true location. In those cases, we employ a Jacobian-based control method via damped least squares (Buss, 2004) to adjust the joint angles on-the-fly in the appropriate waypoints, so that the end effectors maintain their desired position. When applying the control method to adjust for contact behavior errors, we do not adjust the torso position since we are only correcting for errors in z location, which do not affect stability under the stability criterion of keeping the center of mass in the support polygon formed by the fixed feet. In addition

[12] $c_{res}()$ is represented by $(\theta - \theta_{prev})^T (\theta - \theta_{prev})$

to adjusting primitives on the fly as waypoints terminated in behaviors, the operator had the ability to adjust primitives in a batch manner interactively. Each gait adjustment triggered a replanning step that would adjust the waypoints in the motion primitive interactively. If limb-level waypoint adjustment violated stability constraints, then nonlinear waypoint search (Equation 1) was called to update the waypoint. This ability allowed the operator to adjust step distances to guide foot placement.

6 Perception

6.1 Odometry

The pose estimation on RoboSimian is primarily stereo vision based visual odometry (VO) (Howard, 2008) coupled with an INS solution for orientation provided by the IMU. The sensor data is fused together in an extended Kalman filter with zero-velocity update (ZUPT) to avoid pose drift when the robot is stationary. Visual odometry with an IMU alone was typically not reliable enough over the course of a manipulation task, in part due to the majority of manipulation tasks involving the robot arms coming into view of the stereo cameras. To avoid phantom motions from tracked features on the arms, VO masks out the regions of the image containing limbs based on a kinematic model of the robot; however, the pose estimate can still deteriorate due to the loss of trackable features. To mitigate this problem, a secondary form of pose estimation was fused with the existing visual-inertial pose estimator on RoboSimian using lidar point clouds with scan registration.

To provide the platform with accurate localization in all lighting conditions, we developed an additional lidar-based odometry system (LO). LO consists of a real-time implementation of the Iterative Closest Point (ICP) algorithm (Besl and McKay, 1992), in particular the point-to-plane version of ICP (Chen and Medioni, 1992). The implementation is single core, without hardware acceleration, and can produce a registration of two full Velodyne scans in 50 ms on average. 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. Further details on our implementation of LO can be found in (Hebert et al., 2015a).

6.2 Stereo vision

The RoboSimian vision system is comprised of seven pairs of stereo cameras that provide imagery for context and 3D data from stereoscopy. The cameras are oriented around the robot to provide complete 360-degree situational awareness, and the operators can request a combined, high resolution, colored 3D map of robot's environment from the OCU on demand. These aggregated local stereo maps also serve as the basis for object fitting in the OCU. Building global maps, or relying on a priori maps, was deliberately avoided in the competition to avoid perception error modes from accumulating and affecting the fine manipulation planners in an unpredictable way. During development, we recognized that the teach-and-repeat strategy for developing primitives implicitly captures most of the important structure in the manipulation tasks, including collision avoidance, so the high resolution map and motion previews in the OCU simply help the operators double-check the validity of the motion plans in the context of the robot's current surroundings. In a less structured environment, the robot also contains specialized voxel-maps for footstep planning and manipulation, as described in (Hebert et al., 2015a).

6.3 Human-in-the-loop interactive object fitting

This section discusses object fitting via annotations on the stereo disparity image. This ability was crucial in our system to achieve high-level task repeatability. Each fit contained all the necessary information required to perform a manipulation task repeatably. The stored information includes 1) relevant behaviors, 2) starting end effector locations, and 3) pre-manipulation, manipulation, and post manipulation navigation locations. All of the above stored information aids fast and fluid task execution and sequencing.

Figure 10 shows an example of the fitting process for the valve task. Subfigure 10a shows the raw stereo maps in 3D views. The process of manually or automatically aligning models in 3D view is challenging and time-consuming, especially given noisy point cloud data and changing illumination conditions. Instead, the process can be aided with input from the operator on the raw disparity images from stereo (see Subfigure 10b & 10c). Subfigure 10d shows the final fit in 3D maps generated from annotations on the disparity image. The following fitting examples were used in the competition to aid high-level task repeatability in the system.

door coarse: 2 clicks to get door frame. The door's orientation is assumed known relative to robot pose. The door fit had a navigation pose stored in object frame.

door fine: 3 clicks to get door frame plane, and one click to get handle relative to door. The door fit had a navigation pose and a door open behavior stored in object frame.

valve: 3 clicks to get wall normal, and two clicks to get valve position and orientation (see Figure 10). The valve fit had a navigation pose and a valve turn behavior stored in object frame.

drill: 3 clicks to get wall normal, two clicks each to get drill handle position and orientation. The drill fit had a navigation pose and drill adjust/drill pickup behaviors stored in object frame.

drywall frame: 3 clicks to get wall normal, and two clicks to get frame boundaries. The wall fit had a navigation pose stored for beginning the cut plan.

spot fit for surprise: 3 clicks to get wall normal, and one click to get starting position relative to wall. The spot fit had a set of navigation poses and a desired starting end effector pose stored in object frame.

terrain/debris/stair boundaries: 2 clicks to get terrain/debris/stair extent. The extent fit had a starting navigation pose stored in its frame.

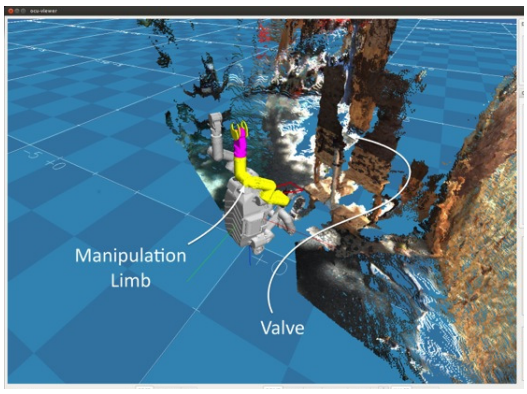
7 Network communications and management

The DRC finals imposed significant degradations to the communications between the OCU and the robot. This was implemented by placing a Degraded Communications Emulator (DCE) in between the OCU and the field computer / robot computers, which delayed and discarded packets in order to simulate degraded network conditions. A low speed (9600 bits / second, bidirectional) link was always active. A high speed downlink (300 megabits / second) was also selectively enabled during the competition. Specifically, the high speed downlink was enabled when the robot was "outside" prior to entering the door or just before beginning the stairs task. The high speed downlink was also enabled after 45 (out of 60) minutes had elapsed regardless of the robot's location. Otherwise, when the robot was "inside," the downlink was enabled for 1 second bursts according to a known randomized schedule.

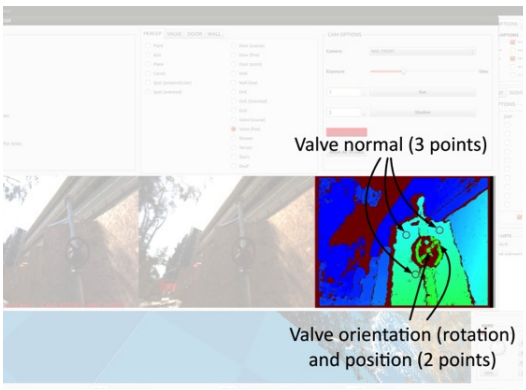
The low speed link presented architectural changes to our system. Our software is divided into 19 modules, which communicate using a proprietary interprocess communication (IPC) protocol. This system allows passing messages between modules over either TCP or UDP, along with additional metadata such as message type IDs and timestamps. Modules are configured at runtime with a directory of module names, IP addresses, and base port numbers in order to allow them to establish connections with each other. Traditionally, all modules are configured with the same IPC directory, which accurately reflects the true location of each module within the system. The modules then establish connections between each other in an all-to-all topology through a shared high speed network.

However, in the DRC degraded communications environment, this approach will not work well because it does not provide a mechanism for limiting bandwidth globally. For example, if multiple modules send messages at nearly the same time, they may easily saturate the low speed link, resulting in packets being discarded. This problem is obviated by changing the system architecture to a more centralized approach, where messages traversing the low speed link all pass through a single bandwidth-limited queue before transmission.

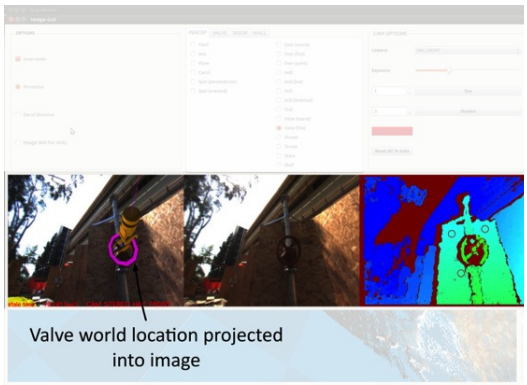
We provided this centralization without requiring any changes to our existing modules by implementing an intentional man-in-the-middle attack on our system. We wrote an additional module, named netman ("NETwork MANager") which impersonates other modules in our system. Two copies of netman are used. One is on the field computer, which



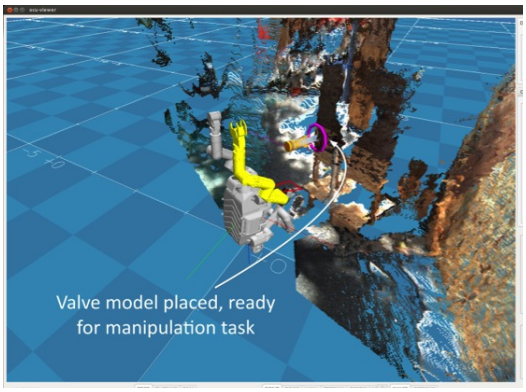
(a) Valve in 3d stereo maps



(b) Interactive perception window with stereo disparity



(c) Annotations on stereo disparity



(d) Valve fit shown in 3d map view

Figure 10: This figure shows an example of the fitting process for the valve task with input from the operator in terms annotations on raw disparity images from stereo. The input for valve consisted of 3 clicks to get wall normal, and two clicks to get valve position and orientation (10b). The valve fit had a navigation pose and a valve turn behavior stored in object frame to initiate robot motion (10d). (best viewed in color)

impersonates every module running on the OCU. The other runs on the OCU, which impersonates every module running on the robot. The two netman processes establish TCP and UDP connections between each other and provide message routing, compression, and bandwidth-limited queuing for the system. Other modules are unaware that they are not communicating directly.

A message traveling directly from the robot to the operator must first traverse a wireless link and then through the DCE. The wireless link is subject to packet loss, while the DCE effectively prevents the use of TCP for any traffic going over the high speed link (preventing automatic retransmission of lost packets). However, DARPA provided the ability to place a field computer in between the wireless link and the DCE. Running the robot-side netman module on the field computer allows communication over TCP with modules on the robot, making the system robust to packet loss.

We applied data compression to all messages sent over the low speed link. All uplink commands from the OCU netman to the robot netman (on the field computer) were sent using TCP in order to guarantee eventual in-order delivery. Repetitive telemetry messages downlinked from the robot netman to the OCU netman were sent over UDP, because losing some telemetry messages was acceptable, and because the protocol overhead is lower. Other downlink messages, such as responses to planning requests and error conditions, were sent over TCP to guarantee delivery.

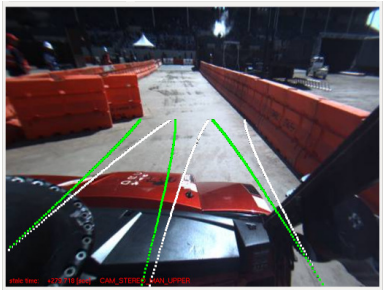
8 Approach and testing

Our approach to preparing for the DRC had a close coupling between software and hardware development with a unified focus on task completion and end-to-end testing. Both hardware (mechanical, electrical and electromechanical) subteams and software subteams were co-located in the same lab space with constant interaction and communication. We had dedicated hardware development days for upgrades, and maintenance interleaved with software development and testing. This close interaction was essential for effective closed-loop development between hardware and software and helped us manage two major hardware redesign events. The first involved introduction of new hands for manipulation seven months prior to the competition (December 2014). This event required a redesign of manipulation tasks in terms of wrist dexterity over finger dexterity. The second event involved the introduction of the new chassis and battery system for untethered operation three months prior to the competition (March 2015). The second redesign event increased the mass and size of the chassis (15.5 kg) and required retraining for previously developed behaviors (especially for egress).

The following subsections discuss our approach and testing methodology for each of the eight tasks leading up to and during the competition. Each subsection outlines the different strategies that were attempted during testing and discusses what did and did not work.

Driving

(a) RoboSimian driving a Polaris at the competition.



(b) Driving interface

Figure 11: The operator interface for the driving task shows arcs corresponding to the center and the boundaries of the vehicle. Green arcs represent the current steering position. White arcs correspond to the desired steering position.

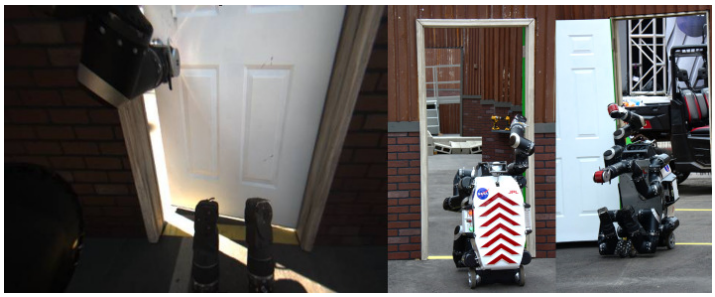
Operationally, the driving task was straightforward. Other than the logistics of testing safely with constrained resources, the technical execution was relatively simple. Development of the egress task took priority over the driving task. Our approach to driving was to let the operator select an arc in the camera image as shown in Figure 11. On approval, a steer and gas command was sent to the robot. We did a piecewise steer then gas strategy for simplicity. The operator also had the ability to specify the duration and percent engagement of throttle. No mechanical driving aids were installed in the vehicle; the robot sat in the passenger seat, grabbed the center of its steering wheel with its end effector, and used one of its limbs to throttle the gas pedal. We trained with more challenging barrier placements than what was featured in the competition. We did not rely on performing a throttle to RPM mapping prior to driving; the operators were able adjust the throttle parameters manually during operation on a need-to basis.

Egress

We made a conscious decision early on to not make any hardware modifications to the Polaris while developing the egress behavior. The reason for this decision was to showcase the versatility of the RoboSimian design and its ability to interact with complex 3D spaces via multi-limb grasping. Such self-manipulation in complex 3D spaces is an important strategic capability for JPL moving towards space applications. As such, we developed the egress behavior on the stock Polaris vehicle with the roll cage.

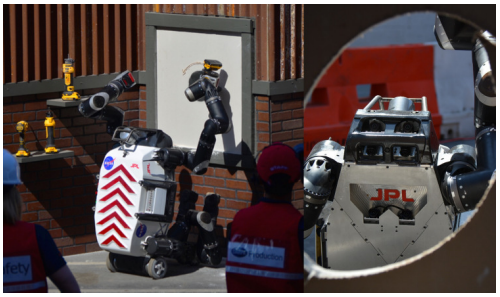
We employed a teach-and-repeat strategy as discussed in Section 5.2 to develop the egress primitive. The primitive was generated by interactively generating coarse waypoints offline and tweaking them during testing to develop the egress behavior (see Figure 9 c-h). Some of the challenges we overcame to develop the behavior include management of internal force buildup under dual arm grasping and movement with stiff position control. Multiple iterations of the hand design, especially the electronics, was required to achieve the desired performance. One issue we encountered was that the hand could not be ungrasped under heavy loads. This required a significant redesign of the hand electronics to operate the finger motors under increased load with higher current. This issue was especially challenging when we transitioned to the new chassis design with the increased weight of the 12 kg battery, which increased the loads on the fingers.

Manipulation tasks: door, valve, wall, surprise



(a) Door at the competition

(b) Valve at the competition



(c) Wall at the competition

(d) Surprise at the competition

Figure 12: Manipulation tasks (best viewed in color)

Manipulation behaviors were initiated via operator-aided object fitting with annotations on the stereo disparity image as discussed in Section 6.3. An operator-aided fit had navigation poses and manipulation behaviors stored in object frame. The operator would re-fit if pose estimation drifted. In the competition, we did a coarse fit to approach the door/valve/wall and a fine fit to perform the manipulation behavior. Figures 12a, 12b, 12c and 12d show RoboSimian performing the manipulation tasks at the competition.

The following end effector behaviors were explored to open the door: finger grasping and handle turning, hooking the handle with the fingers, and preloading the door and moving the wrist to hook the handle. Finger level grasping behaviors were explored for the Trials. They were relatively slow in the event of imperfect grasping/fitting, and they required elaborate recovery paths on failure within the behavior state machine (Figure 7). Coarse behaviors, such as wrist hook, were faster and also had simpler recovery paths in the event of failures. The state machine for the final door behavior is illustrated in the bottom left region of Figure 7.

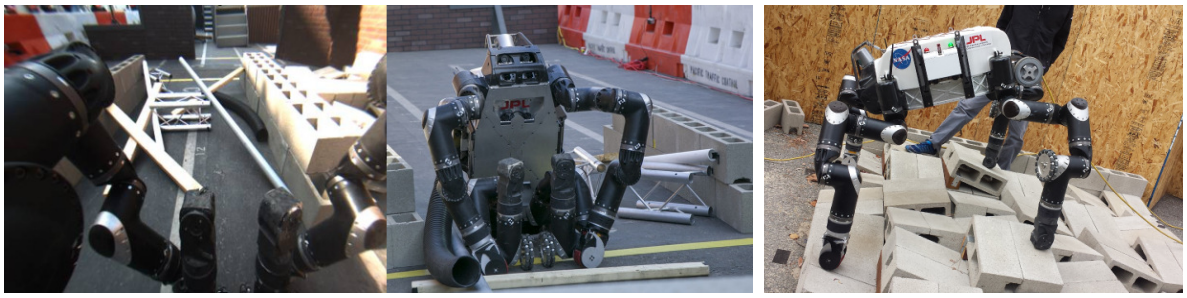
The following end effector behaviors were explored to turn the valve: grasping at the perimeter and turning, grasping the center or the perimeter of the valve and turning (for smaller valves), and hooking the center of the valve and turning. The earlier behavior of grasping the circumference was explored for the Trials mainly because the old hands did not have the finger strength to turn from the center (the tendons snapped). With the new Cam-Hand design (Section

3.2), this was not an issue; grabbing or hooking the center of the valve was faster and easier. The state machine for the final valve turn behavior is illustrated in the bottom middle region of Figure 7.

Among the manipulation behaviors, the wall task was the most challenging. The wall task was sequenced into three discrete objectives: pick up the drill, align the bit with the wall, and cut the hole. In the competition, the following set of behaviors were stored in the drill and wall fits: 1) 'drill grab' (moves hand to detect contact with the shelf and moves up a known amount), 2) 'push drill' (pushes the drill back a bit to make some space on the shelf), 3) 'drive to contact' (moves the wheels until the tip touches the wall), and 4) 'cut a circle given a wall normal.' The state machines for the 'drill grab' and 'drive to contact' behaviors are illustrated in the top right and bottom right regions of Figure 7.

Most surprise tasks we practiced were achieved by composing simple behaviors; an example sequence might look like 1) 'move to a start pose with a backoff' (cord, shower), 2) 'move until contact' (button), and 3) 'pull until a force is received/relieved' (cord, shower, kill switch). These simple behaviors were initiated via spot fits. On Day 1, the kill switch was achieved via a spot fit and a 'pull until a force is relieved' behavior. The plug task that was present on Day 2 was a new challenge for the team that we had not practiced leading up to the competition. We were able to grab the plug with a grab behavior, but we were unable to insert it into the receptacle with visual feedback and end effector Cartesian movements. It was difficult to assess the state of plug in the receptacle from the sitting viewpoint with visual feedback alone. Given time, we would have developed a contact behavior that feels for the dimensions of the receptacle and then inserts the plug.

Debris



(a) Rubble at the competition (b) Terrain (did not attempt at the competition)

Figure 13: Rubble/Terrain task (best viewed in color)

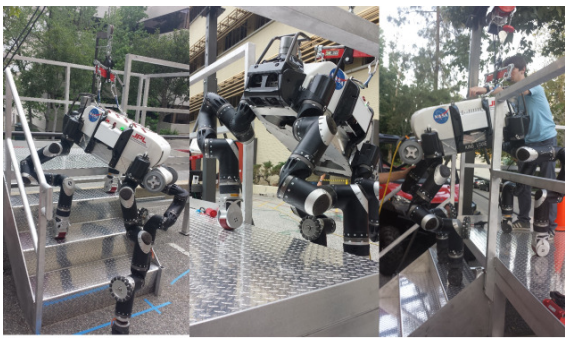


Figure 14: Stairs (not attempted at the competition; best viewed in color)

Early in testing, we investigated walking over the debris field. However, the performance of walking was unpredictable as the debris could shuffle underneath the robot. In the end, it was significantly easier to push the debris out of the way. We did not spend much time investigating removal of debris via pick and place as we found it to be a slow process that required a fair amount of operator input at the Trials. Instead, we focused on developing a transition behavior to a plow posture and worked on clearing debris instead. We developed recovery behaviors to shuffle debris by quickly

lifting both hands a bit and moving debris to one side or the other by moving the hands as we drive. These behaviors were very effective in practice. In the competition, the debris was much simpler than what we practiced. Figure 13a shows RoboSimian performing the debris tasks at the competition.

Terrain

Since the Trials, the most amount of time was spent on developing and improving the walking behaviors. Ironically, we were much faster with debris than terrain, so we chose the former during the competition. Figure 13b shows RoboSimian performing the terrain task at JPL. This particular terrain task was challenging for the RoboSimian platform for two reasons: the required amount of operator input, and the kinematics of narrow stances.

Level of operator input: RoboSimian walks by deliberation and cannot reactively recover by adjusting footsteps. A common failure case in this task was when the end effector was placed on the edges of cinderblocks; when it slipped, this would lead to violation of the quasi-static assumption. This was an issue when the end effector slipped by an amount greater than 20 cm. This issue was significantly improved with the introduction of belly cameras in the new chassis design, as we were able to see where the rear end effectors would go. The 40 cm regularity of cinderblocks required constant adjustment to avoid placing the end effectors on the edges. Individual end effector adjustment via operator input was slow as the operator had to adjust four end effectors for each gait cycle and verify that the stability margins were not violated via each adjustment. In the end, we performed batch adjustment of end effectors for each gait cycle, which turned out to be effective.

Kinematics of narrow stances: In its nominal stance, RoboSimian requires a terrain patch of 1.2 m $\times$ 1.2 m to stand on. Narrow stances severely limited the lift height due to self collisions and also affected stability margins. The structured nature of cinder blocks requires the platform to have precise alignment with the terrain edge in order to achieve 30 cm leg lifts and 40 cm leg moves while avoiding the edges.

Stairs

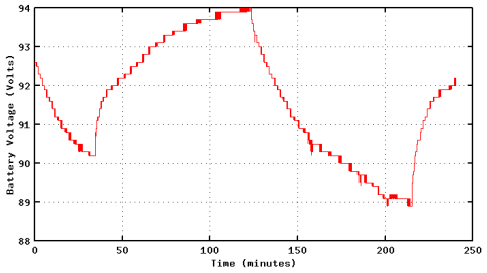
Initially, we investigated climbing strategies in simulation for vertical ladders and stairs that utilized two handrails via grasping. When the stairs in DRC finals were modified to include a single rail on the right side, we reverted to a walking strategy. Similar to the terrain task, we had a stair climb primitive and posture transition primitive that transitions into a narrow posture from the nominal walk posture.

Finding a narrow stance with the right footprint that would fit the proportions of the stairs was challenging. The narrow stance was generated by running a sequence of optimizations offline overnight. A posture solver generated stances with random nominal bias. Then, a series of augmented waypoint searches (as discussed in Section 5.2.2) were performed to generate 6 gait cycles to generate a stair climb primitive. In the end, the narrow stance approach was the most promising but was still awkward in execution, as it did not have the same stability margin as the nominal walking stance that was tested the most. Due to the reduced stability margin, the transition into the narrow posture was sensitive even to mild terrain slopes. Figure 14 shows testing of RoboSimian walking up the stairs task at JPL.

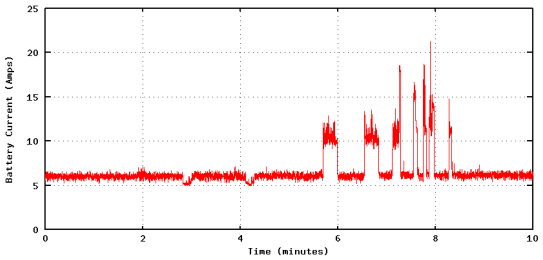
9 Results and analysis from DRC Finals

9.1 Battery performance

Typical battery voltage charging and discharging characteristics of the 2 kWhr battery are shown in Figure 15a. Nominally, we started the battery with a charge of 94 V and discharged it until $\approx$ 88 V which gave us roughly 100 minutes of operation as shown in Figure 15a. In rare cases, we discharged the battery until $\approx$ 80 V, which gave us up to 3 hours of operation time. In theory, we could have discharged the battery further to around 75 V, but operated conservatively during testing to prevent permanent damage to the lithium cells and avoid the nonlinear discharge regions where the



(a) Typical charging and discharge characteristics



(b) Current profile during the egress task

Figure 15: Performance of the 2 kWhr battery. The voltage plot shows a nominal discharge for 100 minutes of operation from 94 V to 88 V. Given a lower bound of 78 V on the battery, we estimate a max operation time of 3 hours. The current plot shows peak demand of 20 A during the egress task.

voltage drops off steeply. Since discharge below 75 V could result in permanent damage to the lithium cells, we used a lower bound of 78 V to calculate operational battery percentage, taking into account the nonlinear discharge profile of the cells.

The current profile during the most demanding task, egress, is shown in Figure 15b. In steady state, we saw a current draw of about 7 A; no current was consumed by the motors as the actuators are braked when they are not moving, and most of the energy consumption is from the two onboard computers and motor logic power. In the competition, the robot was pre-initialized at the garage and was transported to the competition arena in a 'low-power' state. We began the run with 91 V (81% operational time) and ended the run with 86 V, which corresponds to 50% remaining operational time assuming a lower bound of 78 V.

9.2 CPU and memory consumption

Table 2 shows the CPU and memory usage of the system running with processes shown in Figure 4 and discussed in Section 4. The control, perception, and remote computers were off-the-shelf 3.5GHz quad-core computers with 16 GB of RAM. Plenty of CPU and memory resources were still available with the entire system running. On average, the control computer had 73% CPU and 57% memory available, the perception computer had 46% CPU and 92% memory available, the remote computer had 78% CPU and 30% memory available, and the field computer had 98% CPU and 82% memory available. As such, there was no need for cloud computing resources to operate the system. The control computer has enough memory and CPU resources for running network management, and we did not need the field computer to aid resource management. The field computer was mainly used to manage receive TCP packets on the other end of a wireless network.

9.3 Pre-competition task performance

Task times of end-to-end runs with degraded communications in the two weeks leading up to the competition is shown in Table 3. The table outlines our end-to-end testing regiment prior to the competition and also helps explain the decisions we made during the competition. The driving task was consistently around the 5 minute mark for 100 ft driving tests (which we could only perform on the weekends). The total end-to-end run time is only shown for the runs with 100 ft driving. Egress was one of our most consistent tasks and took about 8-9 minutes on all runs. The door and valve task times were also consistent and took on average about 5 and 4 minutes respectively. The door task was a little variable on days when multiple attempts at opening the door were required. The wall task was a lot more variable, as it had many steps to achieve the task. In addition, we tested different placements and orientations of the drill on the shelf as well as different drill types. We also practiced many recovery behaviors in the event of imperfect execution. For the surprise task, we only practiced the switch, button, and shower tasks, and they were all reasonably consistent with 7 minutes on average. The terrain task, at best, took us around 14 minutes to complete. We tested the

Control Computer

Process name	CPU Usage (%)	Memory Usage (%)
ctrl-server	17	2.5
manip-server	2.25	16.9
mob-server	2.15	22.9
limb1-server	1.25	0.1
limb2-server	1.175	0.1
limb3-server	1.075	0.1
limb4-server	0.825	0.1
dd-server	1	0.1
batman-server	0.175	0.1
available	73.1	57.1

Perception Computer

Process name	CPU Usage (%)	Memory Usage (%)
cam-server	41.75	6.1
lidar-server	12	2.4
available	46.25	91.5

Remote Computer

Process name	CPU Usage (%)	Memory Usage (%)
cam-server	41.75	6.1
lidar-server	12	2.4
available	46.25	91.5

Field Computer

Process name	CPU Usage (%)	Memory Usage (%)
netman-server	1.825	17.7
available	98.175	82.3

Table 2: CPU and Memory usage. The last row in each sub-table shows the remaining CPU and memory resources on each computer. Plenty of CPU and memory resources were available on all computers. As such there was no need for cloud computing resources.

debris task under much greater difficulty than the actual setup in the competition in terms of the quantity and weight of the debris. Our approach during testing required multiple attempts at clearing and moving the debris to the side with arm motions and behaviors. Our best end-to-end run prior to the competition was a 7 point run in 50:30 minutes (with augmented difficulty in the debris task and a 100 ft drive).

Date	Drive	Egress	Door	Valve	Wall	Surprise	Terrain	Debris	Stairs	Total
5/19	2:00 (25ft)	9:20	F	6:25	F	7:05 (switch)	20:00			
5/20	1:50 (25ft)	8:00	7:05	4:15	F	5:00 (switch)	15:00			
5/23A	4:10 (100 ft)	8:10	8:30	4:10	13:20	14:00 (shower)	14:00			66:20 (7pts)
5/23B	5:40 (100 ft)	9:00	5:40	4:50	19:00	6:50 (shower)	F			51:00 (6pts)
5/24	3:40 (100 ft)	8:20	5:10	4:30	16:40	7:50 (shower)				46:10 (6pts)
5/27A			2:45	3:10	9:40	8:00 (switch)		6:40		
5/27B			3:00	4:20	11:15	3:45 (switch)		F		
5/28A	1:40 (25 ft)	8:00	5:00	3:30	18:50 (F)	3:20 (button)		7:00 (F)		
5/28B			3:20	3:10	8:45	4:15 (shower)	15:00	9:30		
5/29A	4:10 (100 ft)	8:00	8:30	3:00	13:00	4:50 (switch)	15:00	9:00		50:30D(7pts) 56:30T(7pts)
5/29B			4:45	3:40	11:21	6:15 (shower)		23:00(+)		

Table 3: Task times of end-to-end runs in the two weeks leading up to the competition (with degraded communication). 'F' represents task failure, '+' represents augmented difficulty, 'D' represents total time via debris, and 'T' represents total time via terrain.

9.4 Competition performance and events

During the competition, we achieved a 5th place finish by achieving 7 points in 47:59 minutes on Day 1. On Day 2, we achieved 6 points in 54:43 minutes[13].

[13]The timing data was derived from DARPA video casts on you tube (DARPA, 2015b,a). Day 1 run of RoboSimian on the red course starts here, and we received our 7th point here. Day 2 run of RoboSimian on the red course starts here, and we received our 6th point here.

The individual task times from Day 1 and Day 2 competition runs are shown in Table 4 and Figure 16. We performed the surprise task before the wall task on Day 1 and vice versa on Day 2. The order was chosen based on the perceived difficulty of the surprise task in relation to the wall task. This enabled us to have less blackout events during the later task. On Day 2, we did not complete the surprise task and only received 6 points in total.

Date	Drive	Egress	Door	Valve	Surprise	Wall	Debris	Stairs	Total
Day1 - 6/5	4:26	8:42	5:02	3:17	7:17 (switch)	15:31	3:44		47:59 (7pts)
Date	Drive	Egress	Door	Valve	Wall	Surprise	Debris	Stairs	Total
Day2 - 6/6	4:11	8:18	3:38	4:50	22:13	7:31* (plug)	4:02		54:43 (6pts)

Table 4: Task times in the competition (DARPA, 2015b,a). '*' represents task not completed. The surprise task was performed before the wall task on Day 1 and vice versa on Day 2.

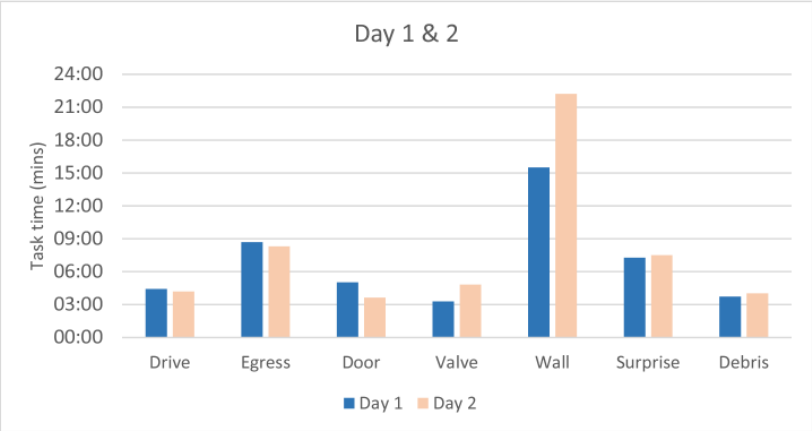


Figure 16: Day 1 and Day 2 Task Time Comparison (DARPA, 2015b,a). The times show highly repeatable task execution between the two days with the exception of the wall task which required multiple attempts on Day 2.

The following descriptions summarize the events and reasons that prevented us from scoring three additional points over the two days of competition. We analyze these issues in hindsight and describe potential approaches to their solution.

Day 1&2: Stairs: not attempted. ***Cause:*** We agreed as a team to only attempt stairs on Day 1 if we had 15 minutes left on the clock. The plan was to attempt it on Day 2 if time permitted. The stairs were our least tested behavior. In its nominal walking stance, RoboSimian is too wide for the stairs. The system is well suited for utilizing two handrails to climb stairs via grasping. With only a single rail on the left side, we reverted to a walking strategy. Finding a narrow stance with the right footprint that would fit on the proportions of the stairs was challenging. In the end, the narrow stance we developed was awkward in that did not have the same stability margin as the nominal walking stance that was tested the most. On Day 2, we attempted the transition into the narrow posture; however, the behavior was sensitive even to mild terrain slopes due to reduced support margins and a high center of mass. The field lead e-stopped the robot to terminate the behavior during the final limb move as the opposite support leg came out of contact. During our limited testing, the terrain was either flat or inclined in the opposite direction, so this issue did not occur in testing. ***Solution:*** We could have developed a climbing strategy using the handrail with sufficient testing or performed extended testing and development of the current walking strategy.

Day 2: Drill: bad cut through circle. ***Cause:*** The cause was bad initial positioning of the drill relative to the circle. Typically, the operators would have waited for camera images in the next comms window to verify initial positioning prior to cutting. The operators made a deliberate call to proceed without the verification/correction step to save time, as we were trying to be bold on Day 2. The second cut also took longer than usual. The robot runs a drive to contact behavior to detect initial wall contact with the drill. After the first cut was made, there was a bug in our code where

this drive to contact behavior was still executing, causing the robot to drive in and push against the wall with the drill chuck instead of the tip, resulting in a loss of alignment with the drill tip and the wall. Finally, it was challenging to see the remaining small black piece of the circle against the black background of the first hole, so we decided to punch the wall in the end. ***Solution:*** We should have waited for the next comms window and verified the alignment in the images.

Day 2: Surprise: failed to insert the plug into the hole; abandoned task due to lack of time. ***Cause:*** The operators deliberately opened the hand to abandon the task as we agreed beforehand as a team to attempt debris when there was 10 minutes left on the clock. The team did not practice the plug task leading up to the competition. The robot's arm occluded the view of the plug and receptacle in the cameras, and it was difficult for the operators to perform the task with visual feedback and end effector Cartesian movements alone. ***Solution:*** We should have developed a contact behavior that feels for the dimensions of the receptacle with the plug and achieves the task without the need for visual feedback.

9.5 Network performance

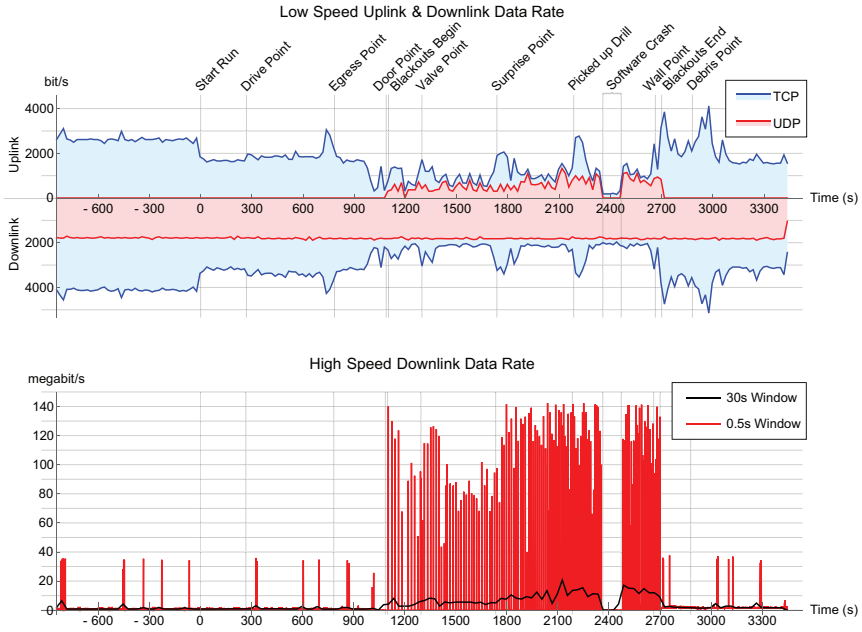


Figure 17: (Top) Network data rate to (downlink) and from (uplink) the OCU over the low speed link during the DRC Finals competition run on June 5th 2015. TCP and UDP rates are stacked and include all Ethernet, IP and TCP/UDP overhead. Data rate was computed by binning packets into 20 second long non-overlapping windows. The approximate times of events during the run are labeled for context. The link speed was limited by DARPA to 1200 bytes/s (bidirectional). (Bottom) Network data rate on the high speed downlink during the DRC Finals competition run on June 5th 2015. All traffic is UDP (due to unidirectionality) and includes all Ethernet, IP, and UDP overhead. Data rate is computed by binning packets into non-overlapping windows of length 30 seconds (to show overall average rate) and 0.5 seconds (to show burst rate).

Low Speed Uplink: During our DRC Finals competition run on June 5th, 2015, we sent 6374 command messages during the 2879 seconds between the beginning of the run and when we scored our 7th and final point. Each message includes a variable sized data payload and a 4 byte header containing message routing and type metadata. The payload and 4 byte header are compressed together, and transmitted with an uncompressed 2-byte long message length field to allow the receiver to separate the byte stream into individual messages.

Figure 17 also shows UDP uplink traffic while we were inside. These messages are sent to acknowledge that the high

speed link is currently active and will not be included in the following discussion.

We sent 167402 uncompressed payload bytes (including the payload 4 byte header). These were compressed to 55646 bytes before transmission (a ratio of 0.33). Each message had an additional uncompressed overhead of 2 bytes for the message length field, giving another 12748 bytes of overhead. The total content of the TCP uplink steam during the 2879 seconds of our run was therefore 68394 bytes, with an average uplink rate of 23.8 bytes / second (this does not include Ethernet, IP, or TCP overhead, which is included in Figure 17).

Of these 6374 command messages, 4661 were automatic image request messages sent at several Hz from the OCU to the robot from the beginning of the run until going inside, and then again after communications blackouts ended 45 minutes into the run. These messages made up a significant amount of uplink traffic and were not strictly necessary, since their function could have been replaced by turning a flag on and off in the camera server to enable or disable automatic image streaming. However, such a direct request/reply architecture was used in order to avoid making major architectural changes to the camera server and because it was not necessary to reduce uplink bandwidth further.

The 4661 automatic image request messages each had an uncompressed size of 16 bytes (12 bytes of data, and 4 bytes of metadata). Therefore the total uncompressed payload size for these messages was 74576 bytes (44.5% of the total). We observed that nearly all of these messages were compressed to 8 bytes. Including the 2-byte message length fields, these automated streaming image request messages therefore represent approximately 46610 bytes out of the 68394 bytes in the entire TCP uplink stream. The remaining data volume comprised of all other messages (21784 bytes) and corresponding data rate (7.6 bytes / second) are more representative of the information provided by the human operators to the robot in order to allow it to perform the DRC tasks.

Reed and Durlach (1998) provide data on the effective information transfer rates of human communication. They cite a range of 25 bits / second (typical) to 60 bits / second (maximal) as the effective information transfer rate of spoken English. Our estimate of the information rate from the above paragraph (7.6 bytes / second) is equivalent to 60.8 bits / second – only slightly higher than the reported maximal information transfer rate of spoken English.

Low Speed Downlink: We sent robot state (joint positions, body pose estimates, etc) and telemetry data (battery voltage, current, etc) over the low speed link downlink using UDP at a constant rate. Figure 17 shows this traffic (including Ethernet, IP and UDP header overhead). During our DRC competition run on Friday, June 5th, 2015, the robot transmitted 6575 state and telemetry packets to the operator station. These were compressed using zlib set to its maximum compression level. The total uncompressed payload size was 520970 bytes, which was compressed to 376046 bytes (a ratio of 0.72, and a final data rate of 130.6 bytes / second). The additional UDP downlink bandwidth shown in Figure 17 reflects overhead.

The TCP portion of the low speed downlink shown in Figure 17 is almost entirely TCP protocol overhead involved in acknowledging uplink data, explaining the near-symmetry to the TCP portion of the low speed uplink. In addition to this overhead, some critical message traffic (such as status after planning or behavior execution) was downlinked using TCP in order to guarantee eventual delivery.

High Speed Downlink: A high speed downlink was available between the robot (including the field computer) and the OCU. This was limited to 300 megabits / second, unidirectionally, and was selectively enabled during the competition run. When the robot was ”outside” (prior to entering the door, and prior to attempting the stairs) the link was always on. After 45 minutes the link was also always on. Otherwise, when the robot was inside, the link experienced a series of scheduled blackouts with 1 second long communications windows between the blackouts.

Counter-intuitively, our average downlink rate was lower when the high speed downlink was continuously active. This is because, without blackouts, we were able to downlink images and maps at any time at the operator’s request. There was no reason to request images or maps that were not currently needed. However, when blackouts were occurring, it was imperative to downlink as much data as possible while it was possible in order to provide situational awareness for the operator through the next blackout period.

Figure 17 (bottom) shows this downlink traffic. During ”outdoor” operations, camera images are streamed at a few Hz

(typically using under 1 megabit / second). The 3D map downlinks are responsible for the data bursts visible prior to entering the door around 1000 seconds into the run (and after the end of blackouts 2700 seconds into the run). While inside, heartbeating at 100 Hz from the field computer to the OCU (over the high speed link) was used to discover when the high speed link was active. When these heartbeat messages were received, heartbeat acknowledgment messages were sent from the OCU to the field computer. These heartbeat acknowledgment messages are responsible for the UDP uplink traffic shown in the Figure 17 (top).

10 Lessons learned

This section summarizes some of the highlights of our platform, discusses the lessons learned, and justifies the core assumptions we made in design, testing and operations:

Adaptability:
Risk vs reward. RoboSimian minimized risk, power consumption, and maximized execution speed by adapting its form to suit different tasks. RoboSimian did not require any resets or interventions in the competition. It was able to perform the driving and egress tasks without any modifications to the Polaris vehicle.

Repeatability:
Non-parametric planning to avoid circular development. Conscious design decisions were made to avoid unitless tuning parameters in planning algorithms. All of the information required for a given task was specified in terms of the task level coarse waypoints. In addition, any parameters needed to formulate the search problem had to have physical units associated with them, e.g. end effector tolerance in IK (mm) and resolution tolerance in trajectory generation (degrees). This design choice allowed us to avoid circular development, where subsequent tuning/improvements to algorithms for new behaviors would break previously developed behaviors.

Proprioceptive feedback instead of exteroceptive feedback for repeatability. Tight integration of stereo point clouds for obstacle avoidance with whole body motion planning in 3D (beyond 2.5D with voxel maps) is a challenging problem. Perception error modes, however infrequent, can affect planner behavior in an unpredictable way and is detrimental to repeatable performance.

Locality:
No global maps. We deliberately avoided relying on a priori maps or building global maps through SLAM and instead focused on 360-degree local situational awareness. This approach prevents perception errors from accumulating and affecting the manipulation planners in an unpredictable way.

Redundant vs. persistent task specification. Specify the task cheaply, quickly, and often instead of tracking objects. For example, the operators specify a coarse fit of the door on approach followed by a fine fit to manipulate. This approach of specifying the task multiple times makes the system robust to imperfect localization and tracking.

Energy efficiency for endurance:
Power-on-to-disengage brakes within actuators. When the robot is not being commanded to move, the brakes in the actuators are engaged and no power is consumed by the motors from the battery. This enabled the robot to be highly energy efficient in a DRC style event, where there are often periods of long pauses between commands.

Energy consumption of onboard computers. The most significant energy consumption in RoboSimian was due to the two onboard computers and motor logic power. On average, there were plenty of memory and CPU resources available in the control and perception computers. Lower power choices (such as fewer cores) could have sufficed for the tasks.

Energy consumption of off-the-shelf motor controllers. The motor controllers were industrial grade, off-the-shelf components that were unoptimized for energy consumption. There is significant room for improvement if custom motor controllers were designed in house.

Hand dexterity vs. strength:
"It was all in the wrist." For most tasks in the DRC, dexterity was needed at the wrist level instead of the fingers. Finger level finesse in manipulation was rarely required (except for triggering the drill, which is a one DoF operation).

Strength of the weakest link. In order to perform whole body maneuvers such as the egress task, the weakest link in the chain is a significant limiting factor, and these are usually the fingers. Therefore, the ability to have high finger strength is essential for robust operation in challenging 3D environments with high reactive loads.

Maintenance:
Repeated design elements. RoboSimian consisted of one actuator design repeated 30 times. This made hardware maintenance and spare inventory manageable.

Operations:
Fieldability. During move-in day, it only took the team 30 minutes to unbox the robot and initialize it from the time it arrived at the Fairplex. In addition, only one operations computer was used for ease of deployment.

Interpersonal communications among operators. We preferred a single computer setup for operations over a multi-operator setup. We had a dedicated operator for the entire event with two supporting co-operators. The members of the operations team were chosen because they communicated the best amongst themselves (and not because they were the fastest). Over time, their speed increased as they trained together with clear communication. We streamlined decision-making but encouraged discussion among the operators. Under stressful circumstances, the discussion provided structure to relieve stress and learn from mistakes later on.

Structured decision making. The second co-operator had an exhaustive operations checklist for consistent operation from run to run, which reduced the mental burden on the primary operator. Strategic decisions, such as time limits to abandon a task, and recovery strategies, were predetermined by the team.

Fast training of new behaviors:
Training under stress. The teach-and-repeat architecture we built in our system was effective in composing new behaviors prior to the competition. On Day 2, however, we found it challenging to develop new behaviors under stressful circumstances. The ability to rapidly train and test new behaviors is critical to generalize our system to new environments. It is also important that this process of adding new behaviors is quick and easily modifiable.

Generalizability. Simpler behaviors were more generalizable than complicated behaviors which were very specific. This tradeoff suggests that compound behaviors composed out of simpler behaviors are more likely to be effective. We relied heavily on force sensing for our behaviors, which provided low-level adaptability for repeatable behavior execution.

11 Conclusions and future work

Team JPL's journey in the DRC has been a memorable experience. Team JPL achieved a fifth place finish in all three stages of the DRC. In the beginning, JPL had funded entries to both Track A and Track B series of the DRC. As the hardware team designed and built the RoboSimian robot, the software subteam competed in the VRC as a Track B entry and received a fifth place finish. We were awarded both a Boston Dynamics Atlas robot and funding to compete in the DRC Trials. However, JPL decided to forfeit their Atlas robot and to pursue the development of RoboSimian as a single unit. The tight coupling between software and hardware teams and its associated iterative development was essential for a wide range of out-of-the-box innovations in our approach to the problems at hand.

Moving forward, JPL is investigating a number of improvements to the current system. First, JPL is exploring current control in its limbs. This opens up a spectrum of possibilities on the hardware, including the ability to achieve variable compliance by controlling current limits in software. In the competition, the limbs were exclusively operated with stiff position control with high control bandwidth of several kHz on the onboard motor controllers. Compliance was simulated with adjustment of position commands with force measurements. Second, imperfect localization is a major bottleneck that needs to be tackled for multi-task autonomy. Improvements to lidar odometry and the incorporation of a navigation or tactical grade IMU are being considered. Third, an improved pipeline during the training process for contact behaviors and fitting strategies will speed up our ability to generate new behaviors on the platform. Finally, we are investigating the design space for faster, lighter, stronger actuators in the limbs.

Acknowledgments

The research described in this publication was carried out at the Jet Propulsion Laboratory, California Institute of Technology, with funding from the DARPA Robotics Challenge Track A program through an agreement with NASA with contributions from the Army Research Lab's RCTA program.

References

Backes, P., Diaz-Calderon, A., Robinson, M., Bajracharya, M., and Helmick, D. (2005). Automated rover positioning and instrument placement. In *Aerospace Conference, 2005 IEEE*, pages 60–71. IEEE.

Backes, P., Lindemann, R., Collins, C., and Younse, P. (2010). An integrated coring and caching concept. In *Aerospace Conference, 2010 IEEE*, pages 1–7. IEEE.

Backes, P. G. (1991). Generalized compliant motion with sensor fusion. In *Advanced Robotics, 1991.'Robots in Unstructured Environments', 91 ICAR., Fifth International Conference on*, pages 1281–1286. IEEE.

Barrett (2016). BarrettHand^{TM} Gripper Spec-sheet. http://www.barrett.com/products-hand-specifications.htm (Last accessed - 7/11/16).

Besl, P. J. and McKay, N. D. (1992). Method for registration of 3-d shapes. In *Robotics-DL tentative*. Int. Society for Optics and Photonics.

Buehler, M., Iagnemma, K., and Singh, S. (2007). *The 2005 DARPA grand challenge: the great robot race*, volume 36. Springer Science & Business Media.

Buehler, M., Iagnemma, K., and Singh, S. (2009). *The DARPA Urban Challenge: Autonomous Vehicles in City Traffic*. Berlin, Heidelberg: Springer-Verlag Berlin Heidelberg.

Buss, S. R. (2004). Introduction to inverse kinematics with jacobian transpose, pseudoinverse and damped least squares methods. *IEEE Journal of Robotics and Automation*, 17(1-19).

Chen, Y. and Medioni, G. (1992). Object modelling by registration of multiple range images. *Image and vision computing*.

DARPA (2015a). Video cast of the Blue Course from Day 2 of the 2015 DARPA Robotics Challenge Finals. https://youtu.be/s6ZdC_ZJXK8?t=35864 (Last accessed - 8/20/15).

DARPA (2015b). Video cast of the Red Course from Day 1 of the 2015 DARPA Robotics Challenge Finals. https://youtu.be/vgt6FPWU2Lc?t=17932 (Last accessed - 8/20/15).

Fallon, M., Kuindersma, S., Karumanchi, S., Antone, M., Schneider, T., Dai, H., D'Arpino, C. P., Deits, R., DiCicco, M., Fourie, D., et al. (2015). An architecture for online affordance-based perception and whole-body planning. *Journal of Field Robotics*, 32(2):229–254.

Feng, S., Whitman, E., Xinjilefu, X., and Atkeson, C. G. (2015). Optimization-based full body control for the darpa robotics challenge. *Journal of Field Robotics*, 32(2):293–312.

Hackett, D., Pippine, J., Watson, A., Sullivan, C., and Pratt, G. (2014). Foreword to the special issue on autonomous grasping and manipulation: The darpa autonomous robotic manipulation (arm) program: a synopsis. *Autonomous robots*, 36(1-2):5–9.

Hayati, S., Volpe, R., Backes, P., Balaram, J., Welch, R., Ivlev, R., Tharp, G., Peters, S., Ohm, T., Petras, R., et al. (1997). The rocky 7 rover: A mars sciencecraft prototype. In *Robotics and Automation, 1997. Proceedings., 1997 IEEE International Conference on*, volume 3, pages 2458–2464. IEEE.

Hebert, P., Aydemir, A., Borders, J., Bajracharya, M., Hudson, N., Shankar, K., Karumanchi, S., Douillard, B., and Burdick, J. W. (2015a). Supervised remote robot with guided autonomy and teleoperation (surrogate): A framework for whole-body manipulation. In *IEEE International Conference on Robotics and Automation (ICRA)*.

Hebert, P., Bajracharya, M., Ma, J., Hudson, N., Aydemir, A., Reid, J., Bergh, C., Borders, J., Frost, M., Hagman, M., et al. (2015b). Mobile Manipulation and Mobility as Manipulation–Design and Algorithms of RoboSimian. In *Journal of Field Robotics*.

Howard, A. (2008). Real-time stereo visual odometry for autonomous ground vehicles. In *Int. Conference on Robots and Systems (IROS)*.

Hudson, N., Howard, T., Ma, J., Jain, A., Bajracharya, M., Myint, S., Kuo, C., Matthies, L., Backes, P., Hebert, P., Fuchs, T., and Burdick, J. (2012). End-to-end dexterous manipulation with deliberate interactive estimation. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 2371 –2378.

Hudson, N., Ma, J., Hebert, P., Jain, A., Bajracharya, M., Allen, T., Sharan, R., Horowitz, M., Kuo, C., Howard, T., et al. (2014). Model-based autonomous system for performing dexterous, human-level manipulation tasks. *Autonomous Robots*, 36(1-2):31–49.

Johnson, M., Shrewsbury, B., Bertrand, S., Wu, T., Duran, D., Floyd, M., Abeles, P., Stephen, D., Mertins, N., Lesman, A., Carff, J., Rifenburgh, W., Kaveti, P., Straatman, W., Smith, J., Griffioen, M., Layton, B., de Boer, T., Koolen, T., Neuhaus, P., and Pratt, J. (2015). Team ihmc's lessons learned from the darpa robotics challenge trials. *Journal of Field Robotics*, 32(2):192–208.

Kalakrishnan, M., Chitta, S., Theodorou, E., Pastor, P., and Schaal, S. (2011). Stomp: Stochastic trajectory optimization for motion planning. In *Robotics and Automation (ICRA), 2011 IEEE International Conference on*, pages 4569–4574. IEEE.

Kennedy, B. and Carpenter, K. (2016). Cam-Hand: This robust gripper design has applicability to both robots and as a prosthetic for the physically challenged. NASA Tech Briefs, NPO 49607, http://www.techbriefs.com/component/content/article/ntb/tech-briefs/machinery-and-automation/24809 (Last accessed - 7/11/16).

Koolen, T., Smith, J., Thomas, G., Bertrand, S., Carff, J., Mertins, N., Stephen, D., Abeles, P., Englsberger, J., Mccrory, S., et al. (2013). Summary of team ihmc's virtual robotics challenge entry. In *Humanoid Robots (Humanoids), 2013 13th IEEE-RAS International Conference on*, pages 307–314. IEEE.

Kuffner, J.J., J. and LaValle, S. (2000). RRT-connect: An efficient approach to single-query path planning. In *IEEE International Conference on Robotics and Automation*, volume 2, pages 995 –1001 vol.2.

Kuindersma, S., Deits, R., Fallon, M., Valenzuela, A., Dai, H., Permenter, F., Koolen, T., Marion, P., and Tedrake, R. (2015). Optimization-based locomotion planning, estimation, and control design for the atlas humanoid robot. *Autonomous Robots*, pages 1–27.

Kuindersma, S., Permenter, F., and Tedrake, R. (2014). An efficiently solvable quadratic program for stabilizing dynamic locomotion. In *2014 IEEE International Conference on Robotics and Automation (ICRA)*, pages 2589–2594. IEEE.

Marton, Z.-C., Pangercic, D., Rusu, R. B., Holzbach, A., and Beetz, M. (2010). Hierarchical object geometric categorization and appearance classification for mobile manipulation. In *2010 10th IEEE-RAS International Conference on Humanoid Robots*, pages 365–370. IEEE.

Matthies, L., Maimone, M., Johnson, A., Cheng, Y., Willson, R., Villalpando, C., Goldberg, S., Huertas, A., Stein, A., and Angelova, A. (2007). Computer vision on mars. *International Journal of Computer Vision*, 75(1):67–92.

Pang, G., Qiu, R., Huang, J., You, S., and Neumann, U. (2015). Automatic 3d industrial point cloud modeling and recognition. In *Machine Vision Applications (MVA), 2015 14th IAPR International Conference on*, pages 22–25. IEEE.

Pelican (2016). Pelican 0550 transport case. http://www.thepelicanstore.com/pelican-0550-transport-case-1187.aspx (Last accessed - 7/11/16).

Ratliff, N., Zucker, M., Bagnell, J. A., and Srinivasa, S. (2009). Chomp: Gradient optimization techniques for efficient motion planning. In *Robotics and Automation, 2009. ICRA'09. IEEE International Conference on*, pages 489–494. IEEE.

Reed, C. M. and Durlach, N. I. (1998). Note on information transfer rates in human communication. *Presence: Teleoperators and Virtual Environments*, 7(5):509–518.

Robotiq (2016a). Robotiq 2-Finger Adaptive Gripper Spec-sheet. http://robotiq.com/wp-content/uploads/2015/12/specsheet-2finger85UR-14apr2016-V2-web.pdf (Last accessed - 7/11/16).

Robotiq (2016b). Robotiq 3-Finger Adaptive Gripper Spec-sheet. http://robotiq.com/wp-content/uploads/2014/08/Robotiq-3-Finger-Adaptive-Gripper-Specifications-ES.pdf (Last accessed - 7/11/16).

Schenker, P. S., Huntsberger, T. L., Pirjanian, P., Baumgartner, E. T., Aghazarian, H., Trebi-Ollennu, A., Leger, P. C., Cheng, Y., Backes, P. G., Tunstel, E., et al. (2001). Robotic automation for space: planetary surface exploration, terrain-adaptive mobility, and multirobot cooperative tasks. In *Intelligent Systems and Advanced Manufacturing*, pages 12–28. International Society for Optics and Photonics.

Shekels, M., Carpenter, K., and Kennedy, B. (2016). Cam-Hand: A robust gripper design for manipulation in positive and negative spaces. *Manuscript in preparation*.

Steder, B., Grisetti, G., Van Loock, M., and Burgard, W. (2009). Robust on-line model-based object detection from range images. In *2009 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 4739–4744. IEEE.

Stentz, A. et al. (2015). CHIMP, the CMU Highly Intelligent Mobile Platform. *Journal of Field Robotics*, 32(2):209–228.

Tedrake, R., Fallon, M., Karumanchi, S., Kuindersma, S., Antone, M., Schneider, T., Howard, T., Walter, M., Dai, H., Deits, R., et al. (2014). A summary of team mits approach to the virtual robotics challenge. In *Int. Conf. Robotics and Automation*. IEEE.

Thrun, S. et al. (2006). Stanley: The robot that won the darpa grand challenge. *Journal of Field Robotics*, 23(9):661–692.

Urmson, C. et al. (2008). Autonomous driving in urban environments: Boss and the urban challenge. *Journal of Field Robotics*, 25(8):425–466.

Wächter, A. and Biegler, L. T. (2006). On the implementation of an interior-point filter line-search algorithm for large-scale nonlinear programming. *Mathematical programming*, 106(1):25–57.

Zucker, M., Jun, Y., Killen, B., Kim, T.-G., and Oh, P. (2013). Continuous trajectory optimization for autonomous humanoid door opening. In *Technologies for Practical Robot Applications (TePRA), 2013 IEEE International Conference on*, pages 1–5. IEEE.