CHIMP, the CMU Highly Intelligent Mobile Platform

Anthony Stentz, Herman Herman, Alonzo Kelly, Eric Meyhofer, G. Clark Haynes, David Stager, Brian Zajac, J. Andrew Bagnell, Jordan Brindza, Christopher Dellin, Michael George, Jose Gonzalez-Mora, Sean Hyde, Morgan Jones, Michel Laverne, Maxim Likhachev, Levi Lister, Matt Powers, Oscar Ramos*, Justin Ray, David Rice, Justin Scheifflee, Raumi Sidki†, Siddhartha Srinivasa, Kyle Strabala, Jean-Philippe Tardif, Jean-Sebastien Valois, J. Michael Vande Weghe, Michael Wagner, and Carl Wellington

National Robotics Engineering Center, Carnegie Mellon University, 10 40th Street, Pittsburgh, Pennsylvania 15201

Received 9 March 2014; accepted 23 November 2014

We have developed the CHIMP (CMU Highly Intelligent Mobile Platform) robot as a platform for executing complex tasks in dangerous, degraded, human-engineered environments. CHIMP has a near-human form factor, work-envelope, strength, and dexterity to work effectively in these environments. It avoids the need for complex control by maintaining static rather than dynamic stability. Utilizing various sensors embedded in the robot's head, CHIMP generates full three-dimensional representations of its environment and transmits these models to a human operator to achieve latency-free situational awareness. This awareness is used to visualize the robot within its environment and preview candidate free-space motions. Operators using CHIMP are able to select between task, workspace, and joint space control modes to trade between speed and generality. Thus, they are able to perform remote tasks quickly, confidently, and reliably, due to the overall design of the robot and software. CHIMP's hardware was designed, built, and tested over 15 months leading up to the DARPA Robotics Challenge. The software was developed in parallel using surrogate hardware and simulation tools. Over a six-week span prior to the DRC Trials, the software was ported to the robot, the system was debugged, and the tasks were practiced continuously. Given the aggressive schedule leading to the DRC Trials, development of CHIMP focused primarily on manipulation tasks. Nonetheless, our team finished 3rd out of 16. With an upcoming year to develop new software for CHIMP, we look forward to improving the robot's capability and increasing its speed to compete in the DRC Finals.

1. INTRODUCTION

The Defense Advanced Research Projects Agency (DARPA) has a long history of sponsoring competitions that advance the state of the art in robotics. For instance, in the two Grand Challenges (2004, 2005) and the Urban Challenge (2007), the robotics field made immense progress in the development of autonomous vehicles. Alarmed by the 2011 Fukushima Daiichi nuclear disaster in Japan, DARPA announced a similar competition, the DARPA Robotics Challenge (DRC), to accelerate the development of robots capable of responding to natural and manmade disasters.

As a Track A participant in the DARPA Robotics Challenge Trials, the Tartan Rescue team developed CHIMP, the CMU Highly Intelligent Mobile Platform, for executing complex tasks in dangerous environments engineered for humans to perform work. CHIMP is roughly human in size and form, so it fits into the same spaces and requires the same workspace as a human. The robot possesses both strength and dexterity to work effectively in these environments, but it avoids the need for complex control by maintaining static rather than dynamic stability. Bipedal machines, such as classic humanoid robots, must balance as they walk about (Grizzle, Chevallereau, Ames, & Sinnet, 2010; Yi, Zhang, Hong, & Lee, 2011). If the terrain is uneven or shifts under the robot's feet, the humanoid runs the risk of toppling over and breaking, thus becoming part of the problem rather than the solution. Although roughly anthropomorphic (see Figure 1), CHIMP includes motorized tracks on all four limbs to provide stable mobility. When performing work, CHIMP rests on the tracks in its legs. This provides a broad, stable base for CHIMP to roll forward, in reverse, or turn in place to position its arms and hands to grasp objects in its environment. To move over uneven terrain, CHIMP changes posture to drive on all four tracks. This transformation lowers CHIMP's center of gravity, provides a wider base of support, and engages more motors while driving. For work tasks, CHIMP's high degree-of-freedom arms allow it to reach into constricted areas. Its electric drive provides both strength and precision. CHIMP was designed to operate with either an external tethered power supply or a battery.

Human operators controlling a robot in a dangerous environment may need to do so over a degraded

*Visiting researcher from LAAS-CNRS
†NREC intern participating on the DRC project
Direct correspondence to Anthony Stentz: axs@nrec.ri.cmu.edu

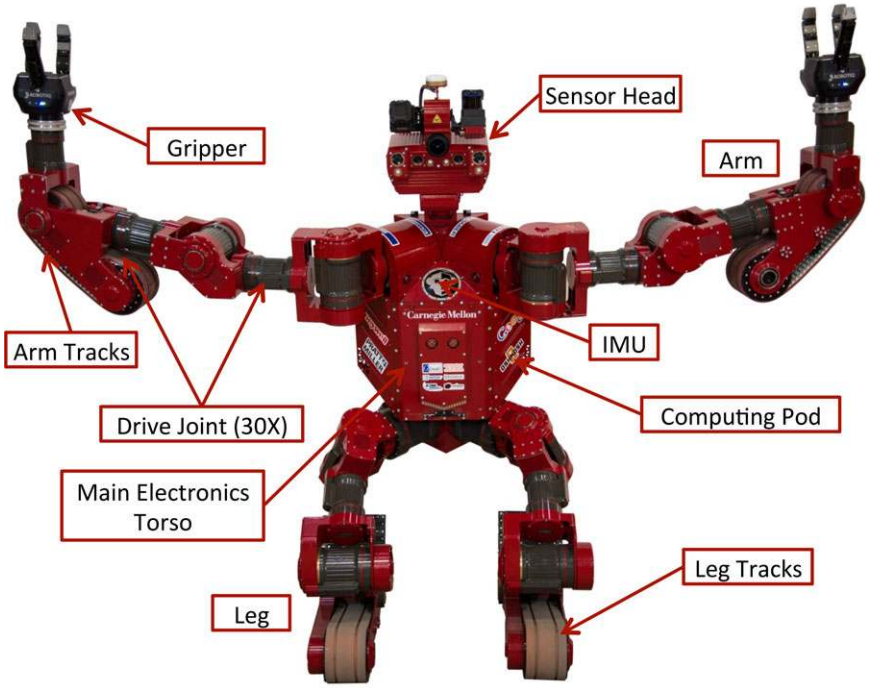

Figure 1. System diagram of CHIMP standing.

communications link, especially in a disaster response scenario when the deployment is unexpected and the infrastructure support is inadequate. The operator may lack line-of-sight to the robot and the link may exhibit latency that complicates real-time control with bandwidth limitations that preclude streaming video. To address these problems, CHIMP leverages techniques from telerobotics (Sheridan 1992), including a virtualized/augmented reality interface (Kanade, Rander, & Narayanan, 1997; Tzafestas, 2006) that is incrementally updated in real time (Kelly et al., 2011). To provide situational awareness, CHIMP uses cameras and laser rangefinders (see Figure 8) to construct a texture-mapped, three-dimensional model of its environment. CHIMP uses an accurate position estimation system based on an inertial measurement unit (IMU), visual odometry, and kinematic motion estimates to incrementally assemble the model as it moves about. CHIMP transmits the model to the remote operator. To minimize the effects of latency, the operator plans robot actions, previews them in the model, and sends them to the robot for execution once properly vetted. During the preview, the operator is able to observe the planned action from any viewpoint and check robot stability, collision avoidance, and grasp approach. To minimize the effects of bandwidth limitations, CHIMP sends just the model changes to update the remote operator's model.

In an ideal world, CHIMP would be fully autonomous, but the state of the art does not support it. Instead, we blend manual and autonomous control in different ways (Kortenkamp, Burridge, Bonasso, Schrenkenghost, & Hudson, 1999) depending on the circumstances, leveraging the strengths of both human and robot. The overall approach was influenced by our efforts in autonomous manipulation (Bagnell et al., 2012). CHIMP supports three operator control modes. In task mode, the operator selects objects for the robot to pick up, indicates which grasping strategy to use, and specifies how devices such as a valve are able to move. The software autonomously plans the robot motion that accomplishes the task, and then it sends the result to CHIMP for execution. Task mode is faster than the other modes, but it may not be applicable in all situations. In workspace mode, the operator controls the position and orientation of the hands and feet. CHIMP automatically calculates how to coordinate the movement of its joints. Workspace mode is slower than task mode but handles a wider variety of situations. In joint mode, the operator directly controls the individual joints for extra precision or to recover if the other modes fail. To overcome latency problems, CHIMP achieves these various behaviors using different methods of control.

The DRC was not only a technical challenge but also a schedule challenge. CHIMP was just a paper concept 15 months before the DRC Trials. The hardware team designed, built, and tested the robot in parallel with the software development. The software team made use of simulators and surrogate robot arms to develop and test until

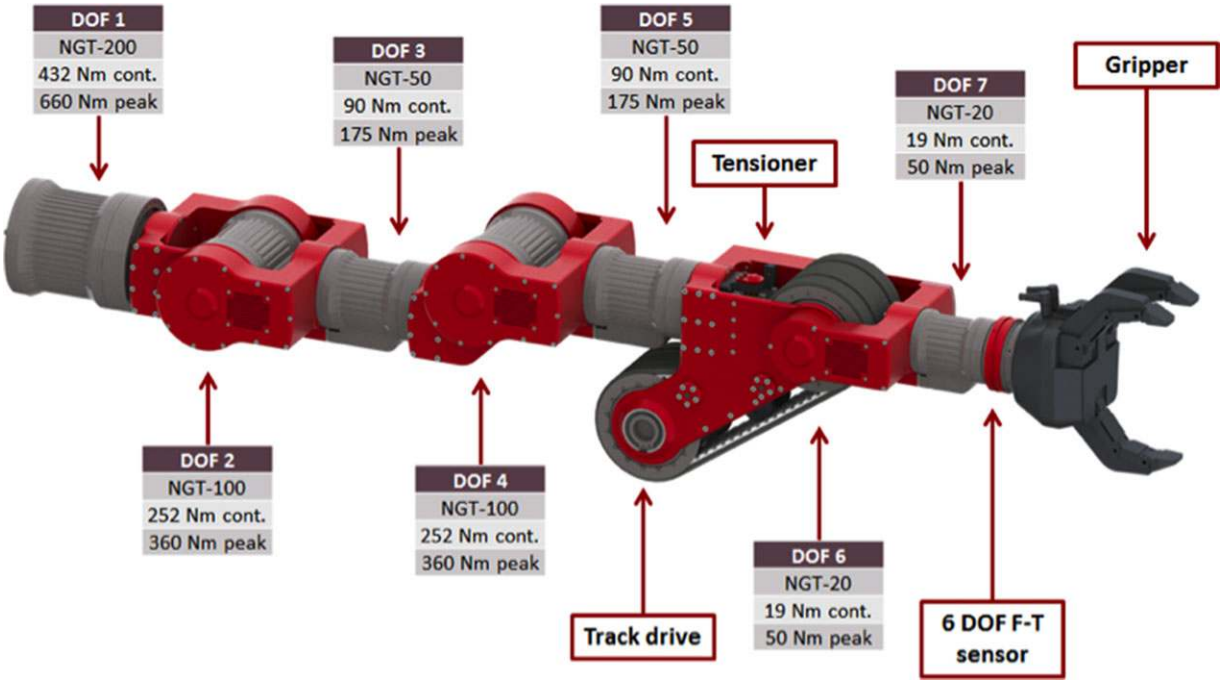

Figure 2. CHIMP limb.

the actual robot was ready. The software was used to evaluate candidate hardware configurations and to put finishing touches on the design. The CHIMP hardware was completed six weeks before the robot was shipped to the DRC Trials. Over these six weeks, the Tartan Rescue team ported, integrated, and debugged the software on the robot, and it practiced the tasks under realistic competition conditions. CHIMP was well prepared to perform the five manipulation tasks but not the three mobility tasks, due to a lack of schedule time to develop software for driving a vehicle, climbing a ladder, and moving in four-limb mode over large obstacles. Nonetheless, the robot performed well enough to capture 3rd place out of 16 teams.

In this paper, we describe CHIMP's hardware and software in detail, including its mechanical design, power, safety, computing, sensing, perception, pose estimation, planning, control, operator interface, and communications subsystems. We also describe CHIMP's performance at the DRC Trials, and we draw conclusions from the endeavor.

2. CHIMP ROBOT DESIGN

2.1. Robot Design Overview

CHIMP is a 39 degree-of-freedom (DOF) robot designed to perform a variety of tasks in human-engineered environments. We selected a design philosophy to minimize the number of engineering challenges. We were able to largely ignore dynamic stability issues by driving on tracks rather than walking with legs. The track modules give CHIMP the ability to drive on all four appendages like a tank or on its knees in a standing position, as shown in Figure 1. This approach means that CHIMP is inherently stable, and it makes many tasks simpler to accomplish. To maximize reach and maneuverability in tight spaces, we designed the arms to incorporate a traditional 3-1-3 kinematic architecture, as seen in Figure 2. Three joints are used at both the shoulder and wrist to create spherical DOFs, while an additional joint at the elbow provides one DOF of redundancy.

Of the 39 DOFs found on CHIMP, 26 are used to control the robot's limbs, four control track velocities, eight adjust gripper finger positions (four for each gripper), and one is used to actively control the spinning of the LIDAR units atop its head. To minimize complexity, we designed just four custom drive joints in different sizes that are used for 30 of the DOFs. CHIMP makes extensive use of modular designs, weighs a total of 400 lb, includes 10,600 individual parts, and is designed for all-electric operation. CHIMP additionally includes full onboard sensing and computation necessary for autonomous mobile manipulation tasks.

CHIMP's limbs provide all actuation requirements and are connected through modular structures containing the necessary support components. CHIMP's head contains a variety of sensor payloads and computing. Its torso houses the IMU, additional computing, power distribution, as well as a dedicated and embedded safety system that continuously monitors CHIMP's operations to guarantee overall system safety. In this section, we provide greater

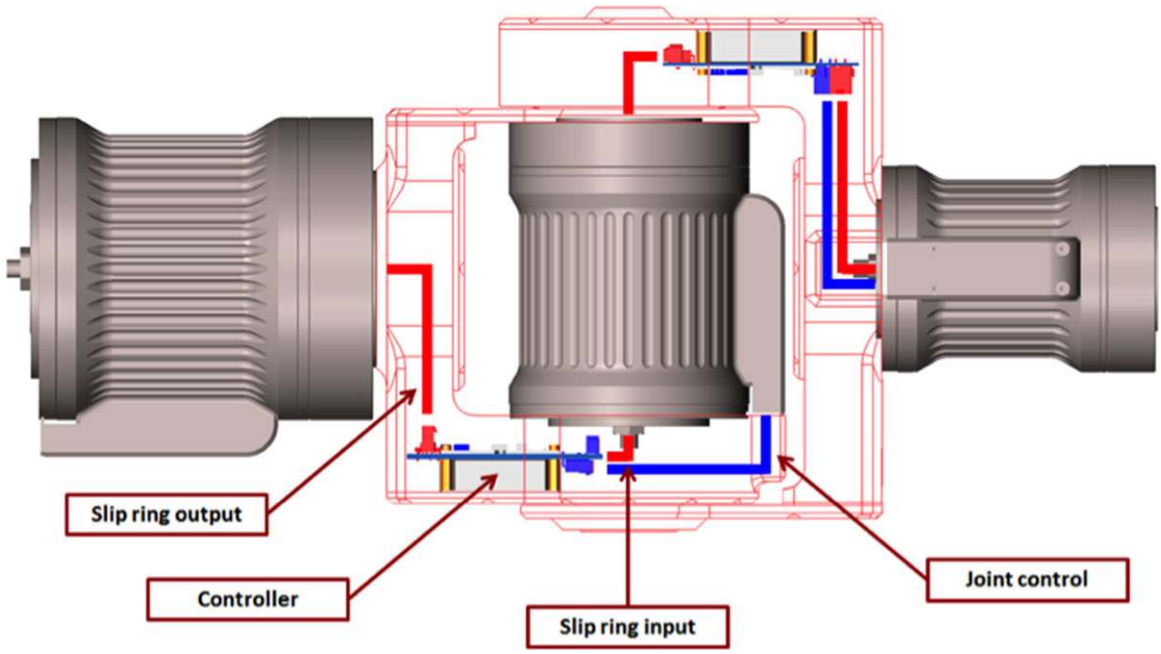

Figure 3. Actuator interconnect structure and wiring detail.

details on CHIMP's mechanical design, drive joints, power, control, safety, computing, and sensor systems, prior to describing the software used to operate CHIMP.

2.2. Mechanical Design

CHIMP is comprised of four modular limbs, a rigid torso, and a fixed sensor head. The limbs are nearly identical and use common drive joints and structural links. The arms and legs differ only in the most distal drive joint, end-effector, and track length. The torso contains all primary networking, computing, positioning, and power electronics. It is also designed to accommodate a hot-swappable battery for tether-free operation. The sensor head integrates the laser scanning mechanism, primary perception sensors, and sensor-related computing.

CHIMP's limbs are comprised of common, self-contained drive joints of four different sizes. Each joint is identical in feature set and general design, but the size and strength are scaled to optimize system weight and power. To size each joint, we analyzed worst-case static and dynamic point cases. Several DRC tasks and load cases were studied, but the ladder climb dominated joint torque requirements and the 2-to-4 limb transition drove stability requirements.

CHIMP's arms contain seven degrees of freedom to maximize reachable manipulation workspace. To prevent self-collisions and allow continuous rotation of the wrist, the arm tracks are only 250 mm long. The end-effector consists of a Barrett 6-DOF force-torque sensor and Robotiq three-finger adaptive gripper. Working with the manufacturer, we made custom hardware and firmware upgrades to the gripper to significantly increase its payload capacity and grasping force.

Our task analysis showed that CHIMP's legs needed only six degrees of freedom. The sixth and seventh drive joints and an end-effector were replaced with a simple 1-DOF foot that is utilized for ladder climbing and vehicle pedal actuation. The foot can rotate to a stowed position when the tracks are in use or rotate to a load-bearing hard stop for supporting the full weight of the robot when standing. To maximize two-limbed driving stability, the leg tracks are 400 mm long but can only rotate 90° before colliding with the limb structure.

The arm and leg track assemblies are identical in design but vary in length for stability and workspace optimization. The track is a custom 100 mm wide, polyurethane, ATN 12.7 toothed timing belt with a thick vulcanized nitrile backing for traction. A custom, self-contained drive joint drives the belt. It includes an incremental encoder, brushless dc motor, custom planetary gearbox, and custom electromechanical parking brake. Unlike joints that use harmonic drives, a planetary gearbox is used in the track drive to better handle the shock and impact loads from offroad mobility. The track is housed inside the limb structure and includes a lead-screw driven tensioner and suspended idler pulley for increased ground contact area.

CHIMP's limbs are modular, serial chains of actuators with controllers integrated into the connecting structure, as

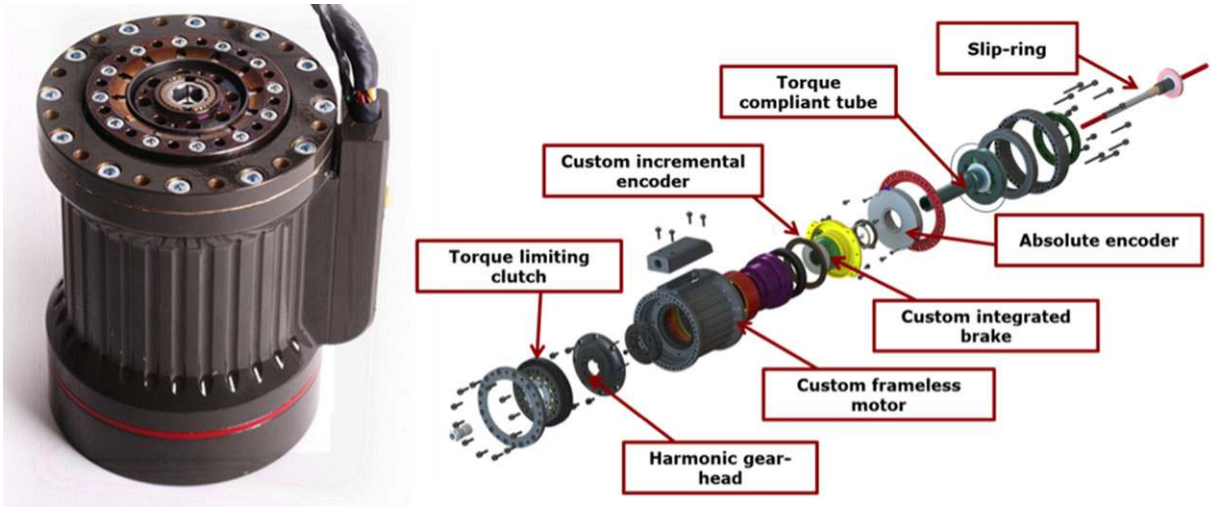

Figure 4. Drive joint (left) and component internals (right).

in Figure 3. Each drive joint features an internal slip ring that passes a shared dc bus for power and a CAN-bus for communication and control. Signal and power for the end-effector are also passed through the entire chain. The integrated motor controllers connect the drive joints electrically. Each joint is controlled using an Elmo Whistle servo drive with a custom carrier board that adds additional inputs, outputs, and signal processing. The carrier board has connectors for input from the previous joint's slip ring, output to the next joint's slip ring, and power and control circuits for one actuator. Through the use of slip rings connecting all joints, each joint is capable of full continuous rotation when not kinematically constrained.

We expended significant effort maximizing the rigidity of the limb and torso structure. The location of the end-effector is primarily calculated using the 18-bit absolute encoders in each drive joint. Any displacement due to structural flex is unmeasured and directly contributes to unknown positioning error. Finite-element analysis of a fully extended arm supporting a 5 kg payload estimated displacement at the gripper to be about 9 mm. Testing of the actual arm showed a total, average error of about 4 mm. This high accuracy positioning contributes to CHIMP's precision control and dexterity.

CHIMP's torso and neck are rigid with no degrees of freedom. The sensor head provides 360° of three-dimensional (3D) LIDAR and color camera data; therefore, there is little benefit to articulating the neck. Kinematic analysis showed that a virtual waist can be created through coordinated motion of the leg joins, allowing CHIMP to bend and twist its upper body despite a rigid torso. Therefore, we made the decision not to include flexibility in the body to maximize electronics and battery volume. The limbs mount to the torso at a 45° angle to maximize the virtual waist's range of motion, minimize shoulder and hip width for door clearance, and increase the workspace for dual-handed manipulation.

2.3. CHIMP Drive Joints

Given that CHIMP required a large number of degrees of freedom, it became clear that to increase our probability of success, we would need a series of actuators that combined power and torque densities that were not commercially available. In addition, we chose to incorporate a feature set that would allow for a more robust and durable design and aid in software development.

Modularity was a key requirement, resulting in compact drive joints that are easily installed or replaced. Torsional compliance built into each joint, in the form of a torque compliant tube running the length of the drive joint, provides a safety mechanism to protect the joint from large impact loads while also simplifying operation when performing force control. By measuring the deflection on each compliant element using input and output encoders, torque sensing is possible on each joint. To further protect the drive joint assembly, a mechanical clutch installed on the output flange provides torque limiting by causing the joint to slip rather than damage internal components, while the use of a magnetically actuated parking brake provides an integrated solution for drive joints to hold position when powered off with the robot statically stable. To provide highly accurate and unique behaviors, each joint provides continuous rotation (when kinematically feasible), has zero backlash, and uses 18-bit absolute encoders to allow for sub-centimeter accuracy at the end of each end-effector. See Figure 4 for an annotated view of a drive joint.

To determine sizing requirements for CHIMP, we developed a function based on output power and torque versus mass, and we ran simulations on various orientations

Table I. Drive joint specifications.

	NGT-20	NGT-50	NGT-100	NGT-200
Continuous Motor Torque (Nm)	19	90	252	432
Peak Torque (Nm)	50	175	360	660
Continuous RPM	30.4	27.1	14.8	10.9
Mass (kg)	1.0	2.2	3.0	5.2
Length (mm)	90.5	113.5	130.5	135.0
Diameter (mm)	77.0	94.5	111.5	140.0

and scenarios. These simulations helped us size and group the actuator development into four common sizes, with resultant specifications shown in Table I.

2.4. Power, Control, and Safety Systems

2.4.1. Power Systems

We designed CHIMP to operate with either a battery or an external tethered power supply. In selecting the main voltage, the voltage tolerance of readily available components was balanced against the desire to minimize the conductor cross section. In general, higher voltages allow more power to be transferred over a given conductor size. However, commercial off-the-shelf (COTS) components such as dc-dc converters and motor controllers tend to cluster at lower voltages. Ultimately a bus voltage of around 48 V was desired, as this is an industry standard with a wide component selection. For battery operation, the full voltage range of the battery was specified to be 66 V when fully charged and 48 V when depleted. For tethered operation, we selected an operating voltage of 52 V to allow for voltage drop along the length of the tether.

Although CHIMP was operated exclusively from a tether for the DRC Trials in 2013, we designed it from the beginning to be operable via a battery. Due to the practical size and weight restrictions for a humanoid robot, the chemistry selection was effectively limited to Li-ion, which has the highest energy density among common chemistries. The specific battery chosen was an increased capacity version of the BB-2590 battery, a standardized rugged battery module designed for hostile environments. BB-2590 battery modules have several built-in protection mechanisms that allow the pack-level circuitry to be greatly simplified without sacrificing safety. Each module has a capacity of 10 Ah at a nominal voltage of roughly 30 V. A single CHIMP battery pack consists of eight modules arranged in four parallel stacks of two, for a total energy of 2400 Wh.

In addition to the main battery or tether power supply, CHIMP also supports a secondary offboard supply designated the auxiliary power unit (APU). The APU can be either a small battery pack or a wall-powered ac-dc converter. APU power is routed to the main dc-dc converters via an ideal diode circuit that prevents it from feeding into the main battery or onto the motor bus. By providing APU power, all electronics (but not motors) on CHIMP can remain powered while the battery is changed, thus precluding the need for a full system restart.

Main power is distributed directly to each limb via independent fuses for fault isolation. The limbs use this power circuit for driving the motors as well as actuating the integrated brakes. Main bus voltage also powers two dc-dc converters to generate 12 and 24 V busses. These lower voltage busses power the various electronic components on CHIMP, such as the computers and sensors. As with the main bus, each component is individually fused for fault isolation.

2.4.2. Motor Control

For CHIMPs motor controller design, our goal was to minimize size and risk while maximizing power and capability. Our custom motor design specified the power requirements for each joint. We selected CAN bus control (as opposed to Ethernet, Serial, or EtherCAT) because it offered a good combination of high speed and low conductor count. After reviewing options for custom controllers as well as COTS parts, we selected the Whistle controller by Elmo Motion Controls. The Whistle had by far the greatest power density of the COTS parts, and we had experience using it. Furthermore, with slightly different arrangements, it was possible to use the Whistle for every drive joint in the robot, which provided a single software and electrical interface.

Because the Whistle is designed to be board-mounted, we used this opportunity to add to the feature set by designing a custom carrier board for CHIMP. Although the Whistle had many of the functions that the robot required, we wanted to use encoders on the output of the gearbox that were not compatible with the motor controller. We mounted an ARM Coretex M4 onto the custom carrier board to interface with these encoders and communicate on the same CAN bus as the Whistle. This controller also interfaced with a microelectromechanical system (MEM) IMU and served as a heartbeat watchdog to shut off its accompanying Whistle if there was a lapse in commands from the control computer.

To simplify wiring and repair, both signals and power pass through the carrier board, allowing the joints to be

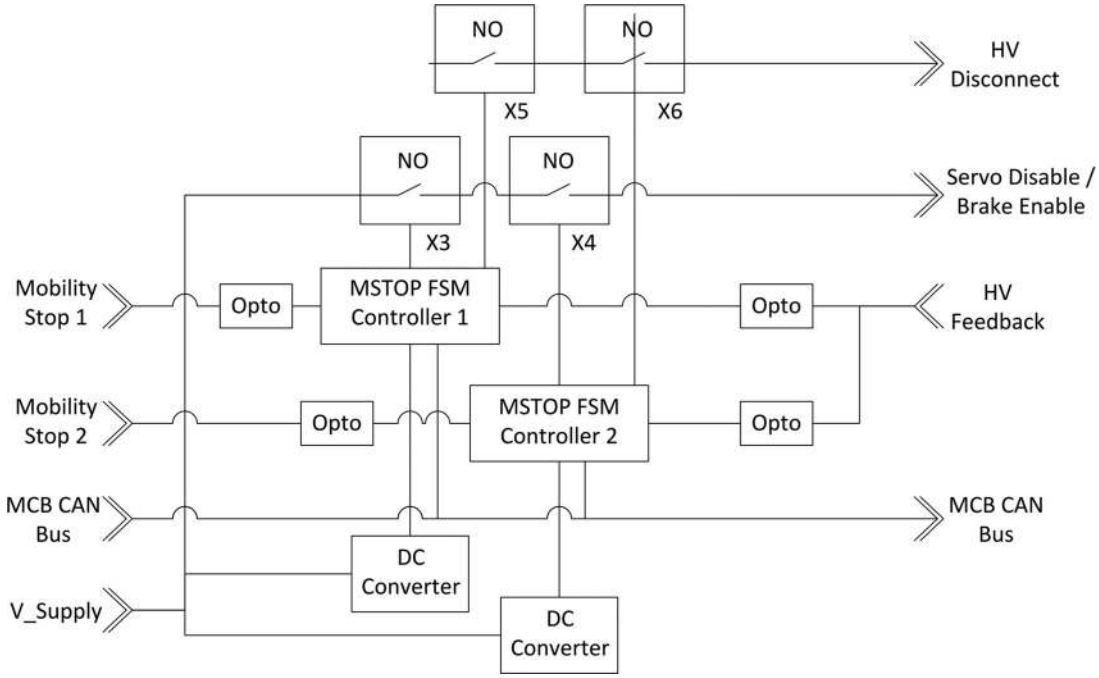

Figure 5. The architecture of CHIMP's MSTOP Controller.

daisy-chained together. This meant using separate, thick power planes as well as separate signal planes in the board to support the pass-through functionality. We created three variants of the same board to accommodate the physical structure of CHIMP and allow easy maintenance and replacement of motor/controller pairs.

We were concerned about the amount of power required to hold the brakes in the disengaged position if the entire robot was moving because of the number of motors in CHIMP. To address this, we designed a brake driver circuit into the carrier board that varies the amount of current provided to the brake so that the "holding" current was much less than the "take off" current.

2.4.3. Safety Systems

Strong, mobile robots such as CHIMP can pose significant safety risks that cannot be mitigated completely by adopting industrial-robot safety standards such as ISO-10218, especially because it is difficult to establish fixed keep-out zones for mobile robots. Other relevant safety standards, such as ISO-26262, can be difficult to apply in the context of complex, autonomous systems (Koopman & Wagner, 2014). Furthermore, our fast-paced development schedule made it infeasible to freeze development of CHIMP's control software in order to analyze and mitigate hazards throughout the system architecture. In response to all of these challenges, we adopted an approach inspired by our earlier work (Wagner, Koopman, Bares, & Ostrowski, 2009) that isolates safety-critical function to an independent subcomponent that can override and disable the more complex control systems.

On CHIMP, a preliminary hazard analysis identified that the consequences of many of the hazards listed in ISO-10218 could be mitigated by maintaining a safe standoff distance between CHIMP and personnel while the high-voltage (HV) motor bus was enabled. When CHIMP is operated, a work area and standoff zone are established. A watchful human operator is given responsibility for identifying unexpected hazards caused by CHIMP exiting the work area or personnel entering the standoff area. The operator also monitors CHIMP for unexpected behavior that may damage the equipment in the test environment or CHIMP itself. If any hazard is identified, the operator can reliably disable CHIMP using the subsystem we call the Mobility-Stop (MSTOP) Controller.

The purpose of the MSTOP Controller subsystem is to prevent CHIMP's joint motors from actuating if a red "MSTOP button" is depressed or if a properly formed heartbeat message is not received periodically (via CAN bus) from CHIMP's control software. The MSTOP Controller has three mechanisms to prevent motion: 1) engaging fail-safe brakes on the motors, 2) sending an inhibit command to limb-motor controllers, and 3) deenergizing CHIMP's motor bus after a brief delay.

The architecture of the MSTOP Controller is shown in Figure 5. This architecture avoids single points of failure by providing redundant microcontrollers, either of which may

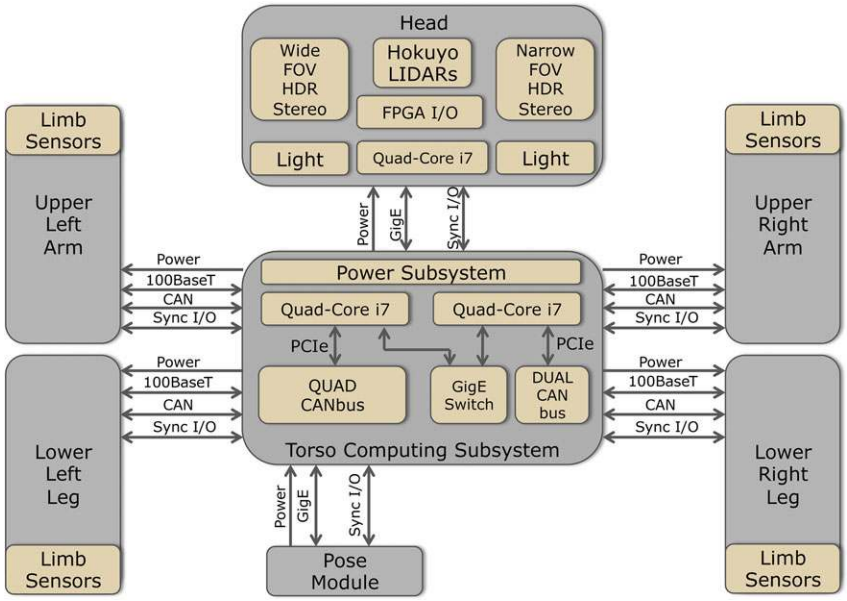

Figure 6. System diagram of CHIMP's electronic components.

disable the system through series-connected solid-state relays (X3-X6). The two Mobility Stop inputs are connected through the tether to two sets of independent contacts in a safety-rated, mushroom-style button. For wireless operation, the Mobility Stop inputs are connected to the independent outputs of a safety-rated wireless receiver. Due to volumetric constraints, the HV Disconnect output is connected to a single high-voltage contactor (not shown) to disconnect power from the motor bus. The risk that the high-voltage contactor fails to open is mitigated in two respects. First, the limb-motor brakes are engaged during an MSTOP, regardless of whether the high-voltage relay has removed power from the motor bus. Secondly, brakes remain engaged and controllers remain inhibited if any HV feedback input indicates that voltage is erroneously present on the motor bus. The inputs and outputs to each microcontroller are optically isolated, each microcontroller has its own voltage regulator, and voltage levels have been chosen to facilitate fail-stop behavior.

2.5. Embedded Computing Systems

CHIMP's embedded computing systems provide enough processing power to support semiautonomous operation. When an operator sends high-level commands to the robot, CHIMP is capable of executing and monitoring all tasks locally while continuing to process large amounts of sensor data. To support the various functions needed for these capabilities, we equipped CHIMP with three Quad Core Intel i7–3820QM CPUs. One of the processors is located in the sensor head and is dedicated to performing sensor processing, such as visual odometry, LIDAR processing, and sensor data logging. The other two processors are mounted in the torso. They control all the limbs, handle the communication interface, and they are responsible for the main data logging. The interconnections between the three processors are shown in Figure 6.

In addition to the main embedded processors, CHIMP is equipped with several field-programmable gate array (FPGA) systems to serve as sensor interfaces and to process data in the sensor head and the torso of the robot. One important hardware feature built into all of CHIMP's embedded computers is the ability to synchronize all data to a common time base. This synchronization allows CHIMP to accurately merge data from various sensors while recording the positions of joints precisely in time.

2.6. Sensor System Design

CHIMP's sensor subsystem must provide accurate data for use in navigation (e.g., pose estimation and obstacle detection), situational awareness (e.g., for the robot and human operator to understand the scene), and manipulation (e.g., for the robot to grasp objects). This led to the selection of a variety of sensors embedded in the robot's head and torso.

As shown in Figure 7, we selected five main sensing modalities that are designed to satisfy CHIMP's navigation, situational awareness, and manipulation requirements. Two LIDAR scanners capture 360° of geometric data surrounding the robot. Similarly, two cameras outfitted with panomorphic fisheye lenses provide video texture data for the geometric data. A pair of high dynamic range (HDR) stereo cameras is configured with a wide field of view and baseline to provide imagery for visual odometry and

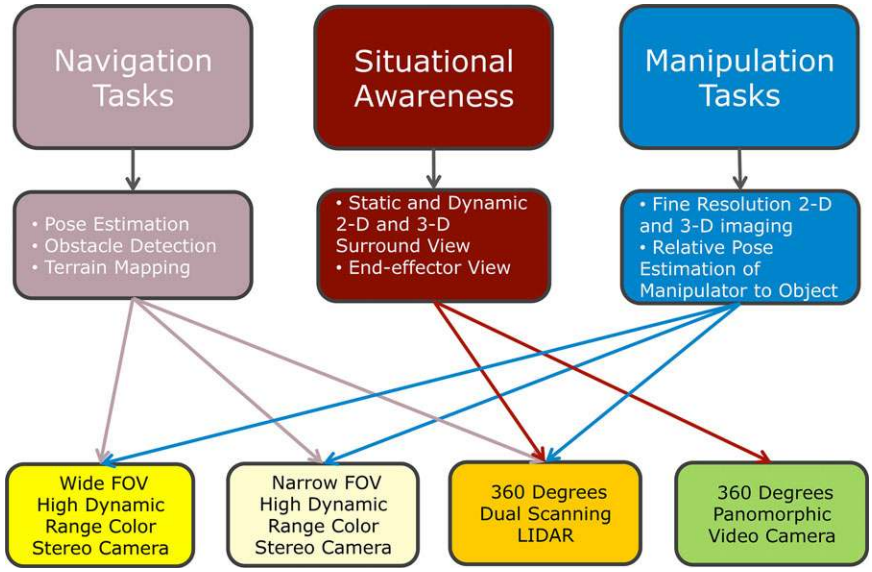

Figure 7. CHIMP's tasks and corresponding sensing modalities.

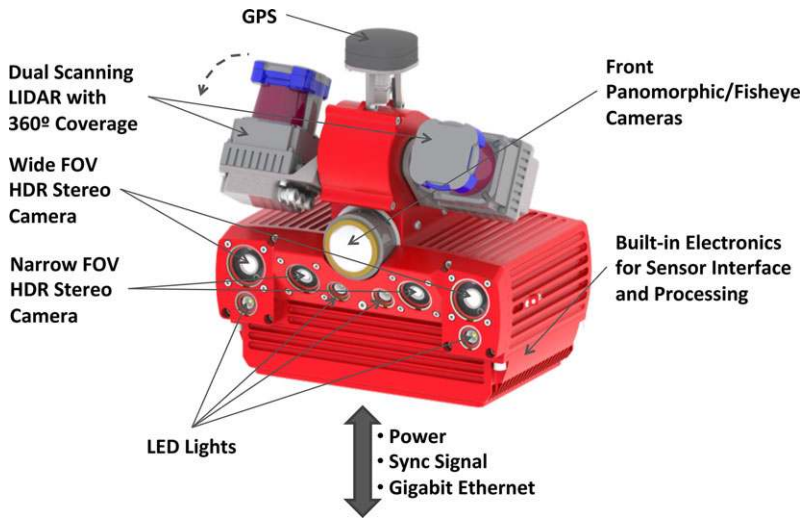

Figure 8. CHIMP sensor head.

obstacle detection. Similarly, a second pair of HDR stereo cameras is configured with a narrow field of view and baseline to provide detailed range information for manipulation tasks.

In addition to the sensing components, the sensor head (Figure 8) contains all embedded processors necessary for the LIDAR and camera imagery processing tasks, including a quad core Intel i7–3820QM, two custom Xilinx Spartan 6 FPGA units, and an Arm Cortex M4. These elements are interconnected using Gigabit Ethernet and USB, and all sensor data are time-stamped using a common time base. Figure 9 describes the architecture of components within the sensor head.

3. SOFTWARE ARCHITECTURE

CHIMP's software system interfaces with the various sensors and actuators onboard the robot to accomplish tasks, while human operators interact with the robot through intuitive user interfaces designed to guide the robot through these tasks. The overall software architecture includes sensor interfaces, perception algorithms, a positioning system, motion-planning algorithms, controls, network and communications management, as well as overall user interaction. All systems are designed to work in parallel, utilizing the limited processing and bandwidth available while providing all necessary capabilities for mobile

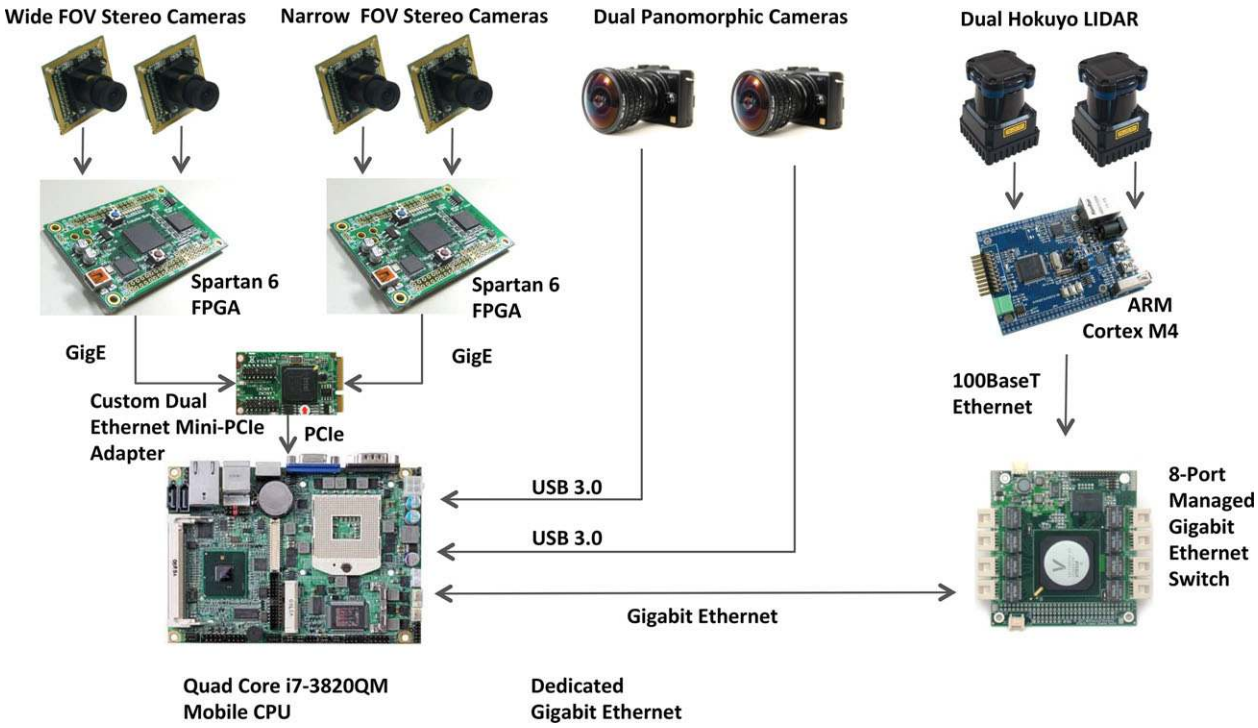

Figure 9. System diagram of sensor head components.

manipulation. Figure 10 displays the overall architecture of the software systems used with CHIMP.

Sensor interfaces retrieve data from the hardware devices found on CHIMP's head, limbs, and torso to be used by the perception system. The perception system has two primary roles. The first and most important is modeling the environment around the robot for motion-planning purposes. The second is providing situational awareness to the human operator. To assist with modeling, a positioning system makes use of a variety of algorithms to provide odometry and pose to the overall system.

The operator control interface is used to display the modeled environment, interact with 3D visualizations of the robot and its surroundings, and place planning fixtures within the environment. Task-specific wizards guide the user through the creation of these fixtures. The operator control interface interacts closely with planning algorithms that run on the operator side computing. These planning algorithms use fixtures to specify constraints in order to generate collision-free motion paths that perform tasks.

These paths are transmitted to the robot where the controls system generates fully timed trajectories. These trajectories incorporate additional calculations of static and dynamic torques to move limbs along the collision-free paths. The trajectories are executed using a 500 Hz real-time controller.

The overall goal of the CHIMP software architecture is to allow a human operator to quickly and easily specify tasks for the robot to accomplish, leveraging all of the subsystems while guaranteeing task execution. Throughout this section, we will provide details for these various components of CHIMP's software systems.

3.1. Operator Control Interface

Human operators interact with CHIMP via an operator control unit (OCU). This user interface provides the operator with situational awareness of the robot and its surroundings, allowing the user to be immersed in the remote environment while making informed decisions about tasks the robot must accomplish. These tasks are accomplished through a series of task-specific wizards that guide the user through the placement of planning fixtures, virtual objects that provide semantic information about what the robot must do, after which planning algorithms are used to preview and validate collision-free motion paths for the robot to execute. The operator interacts with the OCU to accomplish three different levels of control. In task space control, the operator uses the wizards and planning fixtures described earlier to leverage the respective advantages of humans and robots. Humans excel at scene understanding and can easily place fixtures that define the semantics of how objects work; robots excel at the high-precision

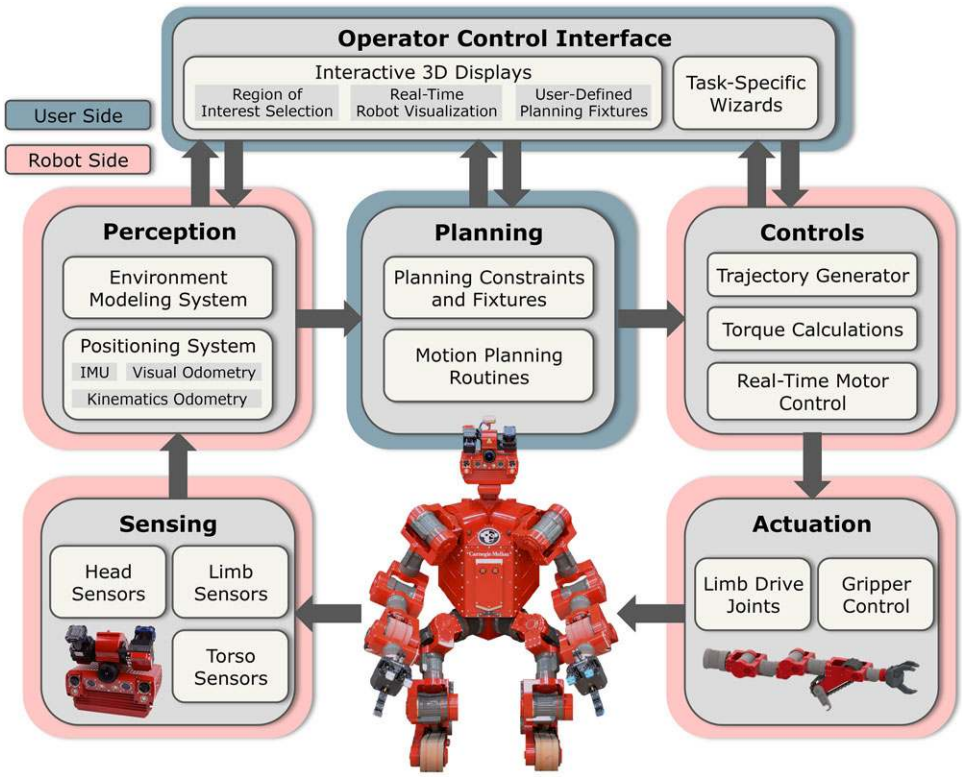

Figure 10. CHIMP robot software architecture.

motions necessary to execute these tasks. In workspace control, a human specifies end-effector motions while the robot manages their execution. This mode is useful when the robot must execute something for which we do not have a proper "task" encoding. Joint space control can be used in a pinch to perform actions on a joint-by-joint basis.

With an accurate model of the environment surrounding the CHIMP robot, the human operator requests tasks using a planning system that generates robot motion trajectories that satisfy all goals and constraints. All tasks for the planning system are presented to the algorithms using the concept of planning fixtures (Figure 11), geometric constraints that describe the steps of a task in detail to the robotic system. Planning goals include target joint configurations, end-effector locations, and task-specific motions (such as moving objects in straight or curved paths). Fixtures are placed directly into the 3D scene of the environment and define semantics about tasks in the environment. For instance, in the wall-cutting task, fixtures describe how the robot must grasp the handheld drill, where the drill's cutting bit is located, and how to trace a path along a wall in order to perform a cut. Rather than being derived from perception data, fixtures are placed directly by a human operator studying the OCU displays for situational awareness.

Task-level wizards simplify robot execution of common tasks. For example, during the valve task the user must label the valve (axis of rotation, where to grasp, and how far to turn), query the planning system, preview the motion plans, and monitor execution. A single wizard, tailored to this specific task, guides the user through these steps. These wizards make use of a modular and customizable architecture, allowing the OCU to display robot data in various configurations to maximize the operator's effectiveness. When necessary, however, the operator may also access lower-level displays that provide greater detail on underlying systems (i.e., health and safety monitoring, raw sensor data, and diagnostics) while using extra bandwidth only as needed.

These modular displays are built using the Qt framework for layout and the ROS libRViz framework for 3D visualization and interaction. We leverage many existing pieces of RViz functionality to provide 3D visualization of different types of sensor data and object displays, such as natively displaying sensor point clouds, transform frame locations, and robot state. The OCU additionally provides interaction with cameras and video streams transmitted by the robot. Using a plugin architecture built atop the Qt framework, individual plugins are allowed to add 3D objects, toolbar and menu items, and create new methods for

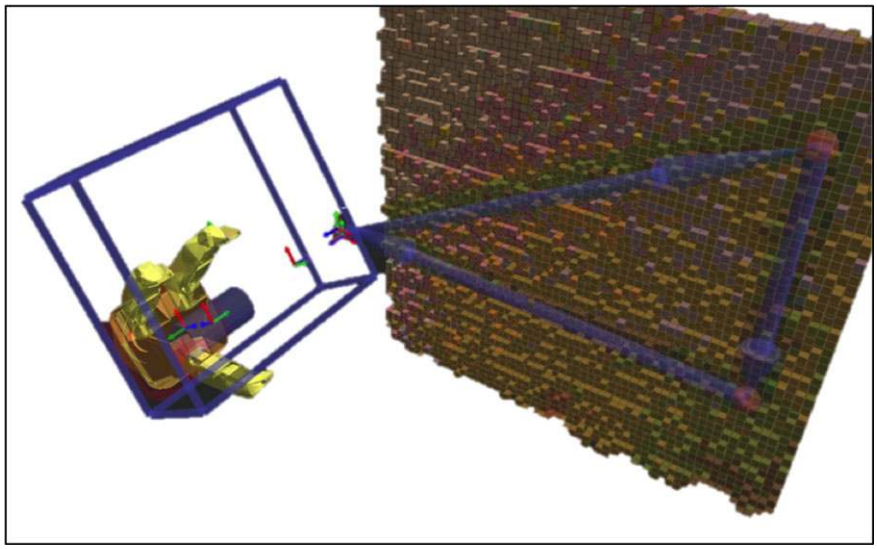

Figure 11. Example fixtures created by the operator to annotate the wall cutting task, including a virtual floating gripper to represent how the robot holds the drill, a volumetric bounding box for the drill, the location of the drill's cutting bit, and a planar cutting path along the wall's surface.

user interaction. All plugin panels can be undocked as new windows and laid out according to each user's preference.

In addition to controlling tasks, the OCU's responsibilities include providing intuitive 3D representations of the perceived environment, robot state, and tasks, interacting with the planning system, and providing modal displays that report robot health, robot communications status, and more. These must all be performed while using as little bandwidth on the communications channel as possible, while still providing the operator with complete control of the robot.

Examples of OCU displays and plugins are shown in Figure 12. Visible in the center is the 3D display of the environment. The OCU is capable of simultaneously displaying multiple views into the 3D world, allowing user interaction in any of these views. Camera views on the left-hand side allow the user to manage camera video feeds and select regions of interest on demand. Additional plugins are used to interact with the communications system (providing real-time diagnostics on its state as well as allowing modifications to data priorities) and monitor robot health. A Vitruvian CHIMP diagram (lower left-hand side) displays detailed diagnostics from the robot's hardware and safety systems, warning users of critical events as they occur.

Finally, the OCU allows the user to fall back on direct teleoperation of CHIMP when necessary. Utilizing a generic gaming joystick, the operator can control CHIMP by specifying workspace velocity control of arbitrary points on the robot. The most common use is to remotely control one of CHIMP's end-effectors for grasping or manipulating an object. It is also used for controlling CHIMP's elbow or tracks and for driving on two or four limbs. Joint space control is possible as well, with the operator exercising complete joint-by-joint control at very fine resolutions when necessary.

3.2. Perception for Environment Modeling

Given no line of sight between operator and robot, the primary goal of the perception system is to provide situational awareness for performing mobility and manipulation tasks. Due to occlusions, however, CHIMP's sensors cannot see the entire environment from a single vantage point, thus the system fuses together pieces from multiple viewpoints when generating a 3D model. These models are geocentric so that they remain fixed as the robot moves about. To improve situational awareness, models are texture mapped with camera data and updated incrementally over time. Furthermore, the OCU allows these models to be rendered from arbitrary viewpoints, allowing a user to easily perceive the robot's surroundings. Due to the limited bandwidth available, the models are sparse and are represented at different resolutions based upon each task, while model changes are transmitted and reconstructed at the OCU to preserve bandwidth. In addition to providing situational awareness, these models are used by motion-planning routines to guarantee safe, collision-free paths when determining robot motions.

The perception system utilizes sensor data to determine the occupancy of environmental models. LIDAR data from CHIMP's sensor head, colorized using the two

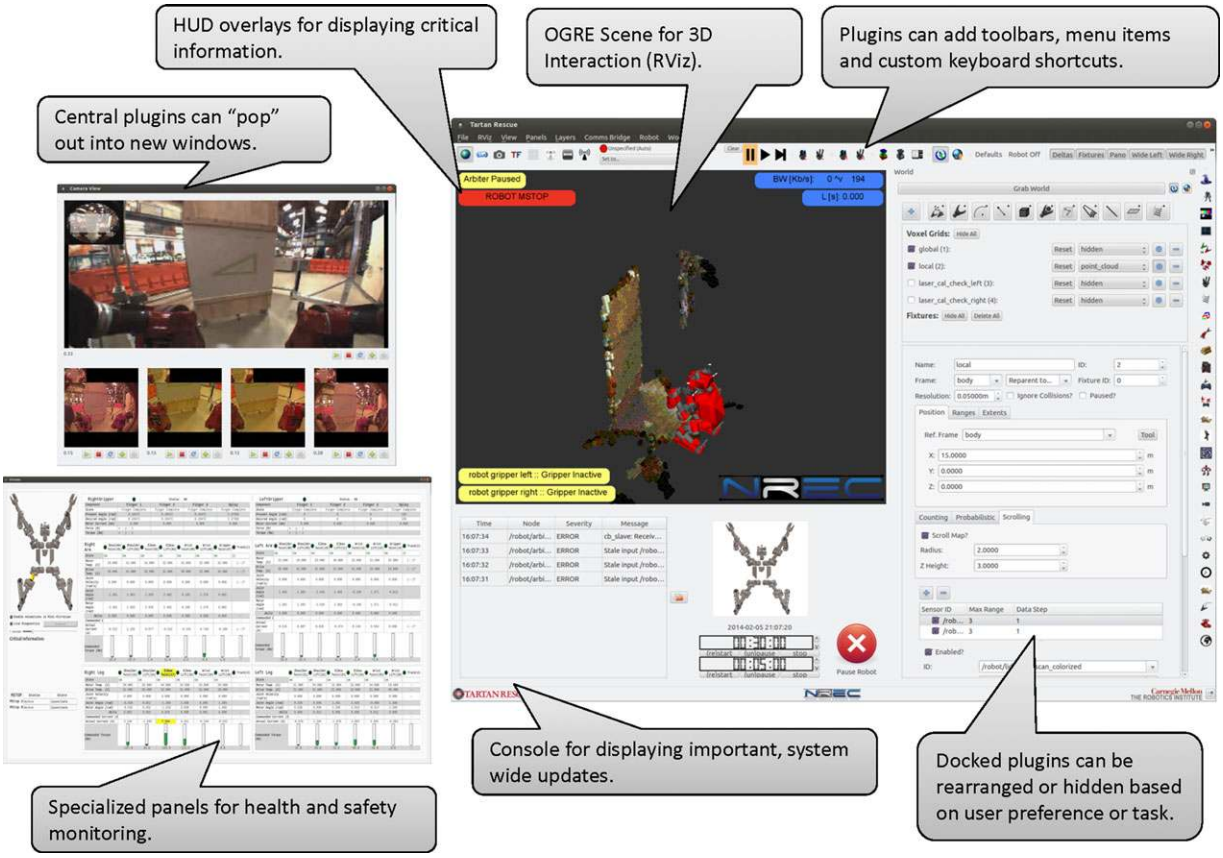

Figure 12. CHIMP operator control unit.

panomorphic fisheye cameras and transformed into geocentric coordinates using the positioning system, are the primary range input to the perception system, however colorized range data computed from stereo disparity are also supported.

For all world modeling purposes, CHIMP uses collections of voxels grids (Figure 13). These grids contain 3D sets of voxels, each containing occupancy and color tagging information. Grids are created with different extents and resolutions based upon the requirements of individual tasks, balancing high-resolution world modeling with bandwidth and computational constraints. The user can view the robot within a coarse grid of voxels (0.5 m resolution) out to the full 30 m sensor range for situational awareness. Higher-resolution models capture the local environment immediately surrounding the robot (0.05 m resolution). On-demand regions-of-interest, typically placed by a user at a specific location, provide information at the 1 cm level and are used when creating planning fixtures for objects in the environment. Through plugin panels available on the OCU, a robot operator can actively modify the settings for each voxel grid, determine which grids to display at a given time, and view the scene from arbitrary vantage points using the 3D user interface.

Voxel grids are implemented using the Octomap open-source library, utilizing octree data structures to efficiently describe occupancy of voxels (Hornung, Wurm, Bennewitz, Stachniss, & Burgard, 2013). Additional features have been added for use on CHIMP, including color tagging individual voxels using textures from registered imagery data and additional models of occupancy for use with the LIDAR data created by CHIMP's sensor head. Based upon the time history of hits and pass-throughs and algorithms counting these values, individual sensor measurements note the likelihood of whether an individual cell is occupied.

As voxel grids are used extensively throughout CHIMP's software, having an efficient means of transfer over a limited bandwidth link is a critical component in our approach. To do so, voxel deltas are computed, determining the set of all voxels changed in a given amount of time. These voxel deltas synchronize distributed models of the world, thus allowing the operator to see the same data as the robot,

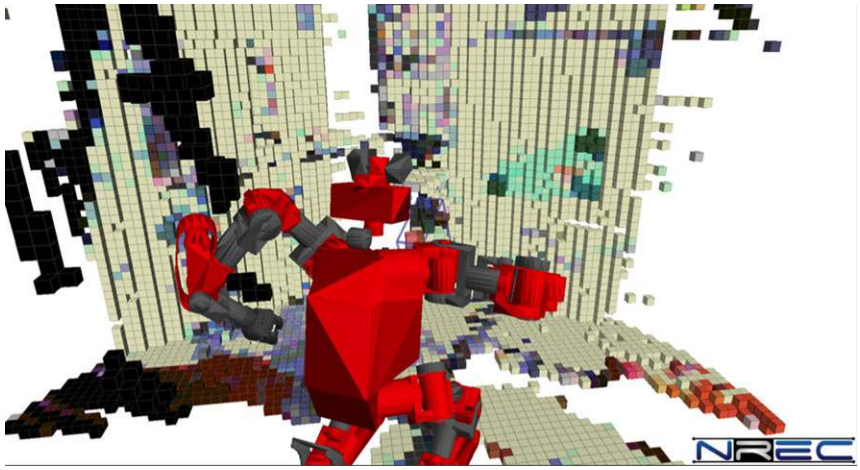

Figure 13. Example of rendered voxel grids during a manipulation task.

while only sending a small amount of data at a time. In nondynamic environments, the voxels do not change very rapidly, thus a huge bandwidth savings is possible through the use of voxel deltas.

In addition to providing situational awareness allowing a human operator to perceive the environment, the voxel models are used when performing motion planning of actions. Using grid representations of voxels, robot motions are tested for collision, thus ensuring the motions generated by the planning routines are collision-free and do not attempt to move the robot limbs through obstacles.

3.3. Positioning System

Highly accurate positioning is valuable for many purposes. First and foremost, maintaining models of the environment (both for situational awareness and for robot autonomy) requires that these models be globally consistent, despite being generated from multiple vantage points as the robot moves about the world. Furthermore, as CHIMP executes tasks, accurate positioning is required to engage with objects (e.g., pick up an object at a specific location) and to avoid collision. Lastly, if the perception models are misregistered over time, the robot will incur a bandwidth cost to update the models using voxel deltas. For these reasons, CHIMP utilizes a positioning system that strives for high accuracy and precision through redundant sensor modalities fused in a Kalman filter.

A variety of sensors provide accurate positioning for perception and mobility tasks. They can operate indoors, outdoors, and in the unstructured environments typical of disaster scenarios. Designed primarily for reliability and redundancy during motion, CHIMP's positioning system prioritizes locally consistent position information so that perceived environment models remain useful throughout robot operation.

At its core, the positioning system uses a navigation grade Honeywell IMU, selected for its accuracy in the worst-case scenario when all other aiding systems fail. This system provides measurements accurate to within 10 m over several minutes at a time, as well as subdegree orientation accuracy for several hours. Using cameras on its sensor head, CHIMP incorporates a visual odometry system (Nister, Naroditsky, & Bergen, 2006) to provide estimates of pose changes over time. This system produces solutions based upon incremental structure from motion estimates while using key frame selection and sparse local bundle adjustment (Engels, Stewenius, & Nister, 2006) to refine the results. An additional input to the positioning system comes from kinematic odometry measurements, calculated using the high-precision encoders found on each of CHIMP's drive joints. Odometry is calculated using a skid steer model of CHIMP's track drive mechanisms and mapped to the varying relative location of the IMU (with respect to the tracks) using joint measurements.

All three sources of measurements—visual, inertial, and kinematic—are blended in a modified extended Kalman filter. The filter uses the common indirect inertial navigation state model, which decouples subsystems and allows the inertial navigation, visual odometry, and kinematic odometry to continue operating normally in the event of filter failure. Furthermore, the filter is modified to handle the relative position measurements of the sensors and to deal with significant latency of some measurements, such as the visual odometry data (George, Tardif, & Kelly, 2013).

CHIMP's positioning system runs on an embedded 400 MHz processor that interfaces directly with the IMU and a GPS receiver. It receives kinematic odometry and visual odometry information over a network link and can account for latency in these measurements. A schematic diagram of the positioning system is shown in Figure 14.

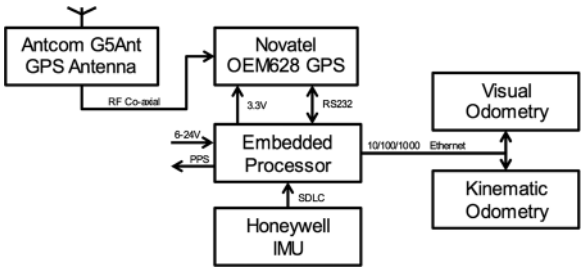

Figure 14. Positioning system architecture.

3.4. Planning Systems

CHIMP's planning system is designed to support task-level execution. This begins with the human operator annotating the environment using planning fixtures, semantically describing how the robot must grasp, manipulate, and move objects. These fixtures define constraints that the planning system must respect when interacting with specific objects (for instance, a valve must be turned precisely on its axis of rotation). Lastly, the planning system must support planning in very high-dimensional spaces while checking for collisions and fixture constraints, to produce full free-space motions that make use of the robot's high degree-of-freedom limbs.

Fixtures that the operator annotates in the scene define task-space geometric constraints on the poses of other objects (either robot links or other fixtures). For example, a grasping strategy fixture may be configured to constrain the right gripper to a particular pose relative to a grabbed object, or a single-DOF axis fixture may be configured to constrain a door handle fixture between a closed and an opened angle. Once these annotations are added to the scene, the planning system is invoked to produce a robot trajectory that moves in accordance with these constraints. The request may specify a desired end-point constraint (i.e., constrain the trajectory end point only), and/or a trajectory-wide constraint (i.e., constrain every trajectory waypoint).

The geometric constraints are specified in Cartesian space in order to be task-relevant and straightforward for the operator to reason about. We chose the task space region (TSR) parametrization (Berenson, Srinivasa, & Kuffner, 2011) due to its combination of simplicity, expressiveness, and desirable properties for planning (e.g., well-defined metric and projection operator). To produce robot trajectories to, between, and along these constraints, we used the Constrained Bi-directional Rapidly-exploring Random Tree (CBiRRT) algorithm (Berenson, Srinivasa, Ferguson, & Kuffner, 2009), represented in Figure 15.

When desired by the operator, the OCU generates a planning request message that includes a snapshot of the full world state (including all voxel data and fixtures), the current robot pose and joint configuration, and the fixture identifiers to be used to constrain the desired robot trajectory, if any. This planning request is then passed to the planning system. This encapsulation allows for planning requests to be logged and later reproduced for efficient implementation and development of planners.

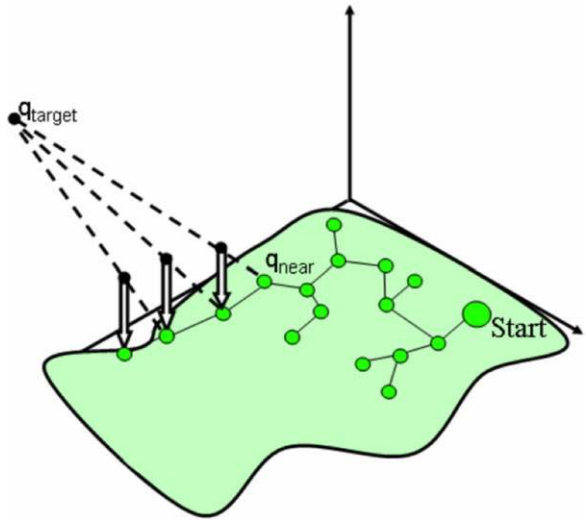

Figure 15. A visualization of the CBiRRT algorithm; during tree extension to the sampled target point q_{target} from the nearest node q_{near}, each new node is projected onto the constraint surface (green). Reprinted from Berenson et al. (2011).

The planning architecture is a multiprocess system consisting of a number of planning nodes, each implementing a common request/response interface using ROS's actionlib. We used a number of different planning nodes, including a CBiRRT planner, a constrained path shortcutter, a cached trajectory generator, and a full-body posture change planner. For tasks that required a predetermined sequence of subplans (e.g., debris clearing, valve turning, and wall cutting), we implemented task planner nodes that intelligently sequenced subplanners and concatenated their results. We also implemented a simple fan-out dispatcher node, which received requests (e.g., from the OCU), and simultaneously broadcasted them to all known planning nodes, aggregating the result in a first-to-succeed manner.

3.5. Control System

CHIMP requires precise motor control to accurately position its limbs and end-effectors to grasp objects and avoid obstacles. Furthermore, it requires accurate models of torque and manipulation forces to support forceful manipulation without damaging objects within the environment (e.g., accidentally breaking a door handle). CHIMP's control system is designed to support these operations while also guaranteeing safety, performance, and reliability in addition to precision and strength. When performing a given task, all control is

performed directly and autonomously on the robot to avoid any delay due to bandwidth restrictions or latency.

All control of CHIMP's limb and gripper motors (a total of 38 degrees of freedom) is performed by a single process running on one of the Core-i7 processors in the torso. By controlling all of the drive joints from a central process, we are able to achieve better synchronization and thus higher end-point accuracy than if we relegated the position control to the distributed Elmo motion controllers. Furthermore, centralized control allows us to add torque terms for each joint that compensate for the static and dynamic effects created by the rest of the robot, improving the stability and accuracy during motion.

The controller consists of three primary components: a high-level trajectory generator that allows the planners to command CHIMP via position or velocity sequences across a robot operating system (ROS) interface; a midlevel control system that calculates correction torques for each of the joints in order to keep them on target; and a low-level hardware interface that manages the CAN bus interfaces to gather feedback from and send commands to each of the hardware nodes.

The trajectory generator accepts sequences of multi-DOF position waypoints and calculates timed joint trajectories that observe position, velocity, acceleration, and jerk limits for each of the joints. It also provides a velocity interface for direct control of the track speeds and for real-time servoing of the limb joints during user teleoperation.

The joint control component performs all calculations in joint units (positions, velocities, torques), leaving it to the hardware interface to translate the values into motor units. Individual joint torques are calculated via standard proportional-integral-derivative (PID) controllers, which can be modified at runtime to provide specific control behaviors such as compliance in specified workspace directions. The control component also calculates feedforward torques to proactively compensate for gravity, coriolis, centripetal, and acceleration forces on each of the kinematic links. PID gains are manually tuned by analyzing the response to single-joint step inputs, and all control values from each cycle can be optionally logged to disk for offline analysis of transient behaviors.

Trajectories are executed in real time, controlling the movement of all joints simultaneously while also monitoring overall joint speeds and torques. By comparing the actual torque of a given joint against the calculated feedforward torque, the control software notes when a joint experiences torque far greater than expected, such as when a limb encounters an unexpected obstacle. In this scenario, the software safely degrades by temporarily halting the control system, engaging the parking brakes on all robot joints, and waiting for the robot operator to reactivate and initiate robot control.

The hardware interface component maintains communication with 64 CAN bus nodes across four 1Mb CAN buses (one per limb). During each control cycle, it calculates motor currents necessary to achieve the specified joint torques, sends the currents to the Elmo motion controllers, and gathers actual motor and joint encoder values from the Elmos and the ARM Coretex modules. Additionally, the hardware interface gathers ancillary sensor data (motor and drive temperatures, drive status, MEMS IMU readings) to protect the hardware from damage. Because the overall control cycle frequency is limited by the speed of the CAN bus communications, substantial effort was spent on developing protocols to minimize overhead and maximize throughput. As a result, we are able to achieve a control frequency close to 500 Hz. As with the control component, the hardware interface component also has an optional logging facility that allows it to record every CAN bus message on all four busses for offline analysis.

To achieve the stringent timing requirements, the controller runs on two dedicated cores of the processor and elevates the Unix scheduler priority of the thread performing the CAN bus communication and joint control calculations. Data flow into and out of the control thread is handled by lower priority threads using circular ring buffers and minimal use of mutexes. The hardware interface abstracts away specific details so that the controller can be run on different hardware or in a simulation mode without any changes to the higher-level ROS clients. Finally, most of the hardware configuration is specified via runtime parameters, which allows us to exchange hardware and maintain calibration values without modifying any source code.

3.6. Communications and Network Bridge

The major requirement of CHIMP's communications system is to manage all communications between the OCU and the robot while ensuring the system stays within the constraints of the network connecting the two. These communications include receiving updates to the 3D models of the environment surrounding the robot, receiving information regarding the current state of the robot, and sending control commands for the robot to execute tasks. Given the severe bandwidth and latency restrictions, we have developed a scheme to constantly compress data and prioritize communications based upon what the robot and operator require at any given time.

At the DRC Trials, DARPA tested the teams' robustness to poor communications channels by introducing artificial impairments to the network between the robot and operators. They used a Mini Maxwell network emulator from InterWorkings Labs to affect both the available bandwidth and latency of the network. During the actual competition, the network alternated between two states: "good" and "bad." During "good" communications, the bandwidth was limited to 1 Mbps and an induced, round-trip latency of 0.1 s round trip. Under "bad" communications, the bandwidth was limited to 100 Kbps and an induced, round-trip latency

of 1.0 s. Given the constant changes in network degradation, it is essential to have a software system constantly monitoring and managing the flow of data between the OCU and the robot to prevent overloading the communications link.

To achieve this with the ROS architecture, we split our system into two ROS networks. One ROS network is for the computers onboard CHIMP; the other is for the OCU computers. Each ROS network has its own ROS master. We created a Communications Bridge (CommsBridge) module to handle communication between the two networks. This provides a single point of data transfer from CHIMP to the operator.

The CommsBridge software is a custom library that provides a point-to-point communications link while monitoring and reporting the health of the link. This includes present bandwidth utilization and latency observed from both sides of link. It provides the ability to send messages using either lossy or lossless style guarantees. This can be configured per data pipe. A lossy guarantee means the CommsBridge will send the message once and forget about it if the link drops it. A lossless transmission will ensure that a message is received by the other side even if this blocks newer data that have arrived in the outgoing queue. Each data channel also has a configurable queue size that can be used to throttle messages and ensure only the most up-to-date information is available. Lastly, each data channel is assigned a unique, rank-order priority. The CommsBridge software uses these priorities when determining what data to send. The CommsBridge will always send data from high-priority channels before those from low-priority channels.

Finally, we created a ROS interface for the CommsBridge library. The wrapper handles the ROS communications on each side of the two networks. It is responsible for subscribing to and queuing ROS messages and republishing them on the other end. This solution allows us to run our software with or without the CommsBridge without any additional changes to the rest of the system. The wrapper also provides additional functionality for controlling each message channel. Each message channel has the option to turn on compression of messages. The operator can decide at runtime to configure a channel to send all data received, disable the channel altogether, or specify that it should send n messages and then stop. The wrapper allows configuration of throttling to a specific rate on a per channel basis.

The Communications Bridge is essential in maximizing the utilization of the poor network link between the operator and CHIMP, and it allowed the operator to concentrate on completing a task and not actively monitor or care about the state of the network.

4. CHIMP TESTING AND TRIALS

We now describe Tartan Rescue's strategy to prepare for the DRC Trials in December 2013 as well as CHIMP's overall performance in the trials. As with any large systems engineering project, the principal challenge in preparing CHIMP for the trials was the massive systems integration required to field a complete hardware and software system. Due to the compressed schedule of the DRC, systems had to be designed and built in parallel; thus we relied upon simulations, surrogate hardware, and testing throughout. An enormous effort was required by the entire team to both finalize the robot's construction as well as thoroughly test all software in time for competition at the DRC Trials.

Despite the disadvantage of not having an available robot to test our entire approach until the final weeks leading up to the DRC Trials, our strategy was a successful one, with CHIMP placing 3rd in the trials, and Tartan Rescue qualifying for additional funding from DARPA to prepare for the upcoming DRC Finals.

4.1. CHIMP Testing

Unlike other teams with access to existing humanoid robots at the start of the DRC program, our CHIMP robot was a mere concept on paper. To design and build the robot from scratch over the course of 15 months, we used a variety of simulations and surrogate hardware systems to begin software testing and systems integration early in the schedule. When CHIMP was fully constructed, only six weeks prior to shipping for the DRC Trials, we transitioned from subsystems verification to full system testing, including end-to-end tests on most DRC Trials task events.

We utilized simulation heavily to guide our design choices. By developing multiple simulations early in the process, we validated design decisions using various methods, thus allowing for rapid design iterations. With an initial design for the robot's degrees of freedom and kinematic arrangement, we used dynamic simulations (created by a team sponsor using commercial software) to validate drive joint sizing and ensure the robot's limbs would be strong and fast enough for typical tasks. Furthermore, we used simulations of the robot's kinematics and statics to test the manipulation workspace of the limbs, validate CHIMP's approach for static stability, and test visibility constraints of sensor placement on the robot's head.

An even more critical component for software development, however, was the use of surrogate hardware to test early and often. The software team used a 7-DOF arm that was designed and built for a previous project for early controls and manipulation testing. While kinematically similar to the eventual CHIMP limb designs, this surrogate arm exhibited significant error in end-effector accuracy, different from CHIMP's design requirement of sub-centimeter accuracy. To combat this, we used a commercial motion capture solution to measure and correct for any inaccuracy, thus simulating the absolute precision of CHIMP's limbs well before they were designed and built. We used similar approaches to test sensors and perception algorithms by mounting CHIMP's various sensors and other components

Figure 16. An operations trailer, containing isolated power, network, and computing, used to control the CHIMP robot. Inset: exterior view of the trailer at the DRC Trials.

into a fully integrated system before finalizing the design and construction of the sensor head. We placed this surrogate head alongside the surrogate arm during manipulation testing, and we also used it with a rolling base (including IMU) to test the positioning system software early in the project.

Using a surrogate sensor head and a fixed base surrogate arm, our software development efforts focused on manipulation tasks. When the CHIMP robot was fully constructed on November 1, 2013, its design with static stability and precision motion control allowed our software to be directly ported between platforms with very few modifications. Further development was required to integrate software that could not be tested using either simulations or surrogate hardware, such as mobility using CHIMP's track drive systems and kinematic odometry input to the positioning system.

To realistically test bandwidth constraints and the effect upon remote operation, we performed sequestered operation over the network-disrupting MiniMaxwell device, identical to the setup used at the DRC Trials. An operations trailer (Figure 16) was outfitted with desks, isolated power, networking, and computing, providing a setup that was used both in NREC's Pittsburgh facility and at the DRC Trials. In the days and weeks prior to the DRC Trials, we performed testing on mocked-up challenge events in configurations that were as close as possible to the actual configurations at the trials.

4.2. DARPA Robotics Challenge Trials Performance

The DRC Trials were held on December 20–21, 2013, with CHIMP being one of 16 robot competitors attending. While CHIMP was designed to have humanlike dexterity and manipulation capabilities, its track-drive mechanisms for stable locomotion and precision manipulator arms to perform tasks made it one of the most unique robot designs. The supervised autonomy approach taken to control CHIMP, in which a human operator places fixtures to annotate the environment and guide task execution, was successful throughout the challenge events. Furthermore, the use of a statically stable robot that was designed for both strength and precision led to CHIMP's overall reliability at accomplishing the tasks required.

CHIMP's performance at the DRC Trials focused heavily on manipulation tasks, the result of a conscious decision months before to focus development efforts on critical manipulation capabilities developed using surrogate hardware. Despite the robot's unique mobility design, the short time frame available with the robot required prioritization, thus curtailing development on the three mobility tasks. At the trials, CHIMP scored only 2 of 12 possible mobility points. In comparison, CHIMP was a top contender on manipulation tasks, scoring 16 of 20 possible points. Overall, CHIMP placed 3rd among the field of 16 competitors, a strong standing for a robot that was a mere concept 15 months prior. Table II shows the individual points awarded to CHIMP during the DRC Trials.

Our approach for supervised autonomy was also successful at the DRC Trials. Our software system utilized planning algorithms to determine CHIMP's own motions for most tasks. It performed collision checking against itself and environment models while feeding back the state of the robot and environment at all times to the human operators, rather than relying heavily upon teleoperation approaches. An example of our planning approach is the

Table II. DRC Trials results for the CHIMP robot.

Event	Time	Interventions	Points
Vehicle	N/A	N/A	0
Terrain	30	0	1
Ladder	30	0	1
Debris	29	0	4
Door	30	0	2
Wall	27	0	4
Valve	21	0	4
Hose	30	0	2
TOTAL	**197**	**0**	**18**

wall cutting task (Figure 17), in which CHIMP executed perfectly straight cuts to remove a piece of dry wall while using a handheld drill. Furthermore, once a human operator annotated the wall cut desired, the software system computed and executed the necessary motions with very little human intervention. In all, CHIMP's stability, precision, and strength, coupled with unique methods for operator control, were all great successes at the DRC Trials.

5. CONCLUSIONS

CHIMP's performance at the DRC Trials validated its design. The robot was never in danger of falling down; in fact, it was the only robot that did not incur an intervention to arrest a fall. The electric drive limbs demonstrated both power and precision in picking up a heavy tool and cutting a straight line in the wall. The high-DOF limbs enabled CHIMP to maneuver in constrained environments when removing boards from a truss and opening a door.

The operator interface was an excellent tool for the remote operator to plan, preview, and evaluate candidate operations before committing CHIMP to execution. The incremental model construction and updating made such efficient use of the limited bandwidth that operators performed the same whether communications were degraded or not. The operators made use of all modes of control, including task mode for removing debris, workspace mode for grasping the rail and stepping on the ladder, and joint mode for adjusting the legs to balance the torques in CHIMP's knees.

In the end, CHIMP's performance was limited only by a lack of sufficient development time. We started the DRC Trials with the software capability to compete for 22 of the 32 possible points. In the end, Tartan Rescue captured 18 of those points for 3rd place overall. On the road to the DRC

Figure 17. CHIMP at the conclusion of the "Wall" task event.

Finals, we will develop the remaining software necessary to capture all points. We will also work on increasing the level of autonomy and reducing the time required to perform the tasks. We believe that the CHIMP robot hardware is an excellent design and we will develop software that makes maximal use of its inherent capabilities.

ACKNOWLEDGMENTS

Development of the CHIMP robot has been supported by DARPA/SPAWAR under Contract No. N65236–12-C-3886. This work would not be possible without the extreme dedication of the Tartan Rescue team as well as the entirety of the National Robotics Engineering Center at Carnegie Mellon University. Additional team sponsors have provided generous support, including Accurate Gear and Machine, Brentronics, Eclipse Metal Fabrication, Elmo Motion Control, Faulhaber, Glenair, Google, Harmonic Drive, Honeywell, Kollmorgen, Micromo, Oshkosh/JLG, Pratt & Miller, Robotiq, Sepac, and THK.

REFERENCES

Bagnell, J. A., Cavalcanti, F., Cui, L., Galluzzo, T., Hebert, M., Kazemi, M., Klingensmith, M., Libby, J., Liu, T. Y., Pollard, N., Pivtoraiko, M., Valois, J.-S., & Zhu, R. (2012). An integrated system for autonomous robotics manipulation. In IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS'12).

Berenson, D., Srinivasa, S., Ferguson, D., & Kuffner, J. (2009). Manipulation planning on constraint manifolds. In Proceedings of the IEEE International Conference on Robotics and Automation (ICRA '09).

Berenson, D., Srinivasa, S., & Kuffner, J. (2011). Task space regions: A framework for pose-constrained manipulation planning. International Journal of Robotics Research, 30(12), 1435–1460.

Engels, C., Stewenius, H., & Nister, D. (2006). Bundle adjustment rules. Photogrammetric Computer Vision 2.

George, M., Tardif, J.-P., & Kelly, A. (2013). Visual and inertial odometry for a disaster recovery humanoid. International Conference on Field and Service Robotics (FSR '13).

Grizzle, J. W., Chevallereau, C., Ames, A. D., & Sinnet, R. W. (2010). 3D bipedal robotic walking: Models, feedback control, and open problems. 8th IFAC Symposium on Nonlinear Control Systems (IFAC '10).

Hornung, A., Wurm, K. M., Bennewitz, M., Stachniss, C., & Burgard, W. (2013). OctoMap: An efficient probabilistic 3D mapping framework based on octrees. Autonomous Robots, 34, 189–206.

Kanade, T., Rander, P., & Narayanan, P. J. (1997). Virtualized reality: Constructing virtual worlds from real scenes. In IEEE MultiMedia Magazine 4, 1 (IEEE '97).

Kelly, A., Chan, N., Herman, H., Huber, D., Meyers, R., Rander, P., Warner, R., Ziglar, J., & Capstick, E. (2011). Real-time photorealistic virtualized reality interface for remote mobile robot control. International Journal of Robotics Research, 30(3), 384–404.

Koopman, P., & Wagner, M. (2014). Transportation CPS safety challenges. NSF Workshop on Transportation Cyber Physical Systems.

Kortenkamp, D., Burridge, R., Bonasso, P., Schrenkenghost, D., & Hudson, M. (1999). An intelligent software architecture for semiautonomous robot control. In Autonomy Control Software Workshop (Autonomous Agents '99), 99, 36–43.

Nister, D., Naroditsky, O., & Bergen, J. (2006). Visual odometry for ground vehicle applications. Journal of Field Robotics, 23(1), 3–20.

Sheridan, T. B. (1992). Telerobotics, automation, and human supervisory control. Cambridge, MA: MIT Press.

Tzafestas, C. S. (2006). Virtual and mixed reality in telerobotics: A survey, Industrial Robotics: Programming, Simulation, and Applications. In Low Kin Huat (E d.), ISBN: 3-86611-286-6, InTech, DOI: 10.5772/4911. Available from: http://www.intechopen.com/books/industrial_robotics_programming_simulation_and_applications/virtual_and_mixed_reality_in_telerobotics__a_survey.

Wagner, M., Koopman, P., Bares, J., & Ostrowski, C. (2009) Building safer UGVs with run-time safety invariants. National Defense Industrial Association Systems Engineering Conference.

Yi, S.-J., Zhang, B.-T., Hong, D., & Lee, D. D. (2011). Practical bipedal walking control on uneven terrain using surface learning and push recovery. IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS '11).