Robot System of DRC-HUBO+ and Control Strategy of Team KAIST in DARPA Robotics Challenge Finals

Jeongsoo Lim, Hyoin Bae, Jaesung Oh, Inho Lee, Inwook Shim, Hyobin Jung, Hyun Min Joe, Okkee Sim, Taejin Jung, Seunghak Shin, Kyungdon Joo, Mingeuk Kim, Kangkyu Lee, Yunsu Bok, Dong-Geol Choi, Buyoun Cho, Sungwoo Kim, Jungwoo Heo, Inhyeok Kim, Jungho Lee, In So Kwon and Jun-Ho Oh

1 Introduction

This paper presents the development of both the hardware and software, the integration of these two elements to build up the whole robotic system, and the control strategies of Team KAIST at DARPA Robotics Challenge (DRC). In DRC, each team had to solve eight tasks under restricted communication conditions, as can be seen in Fig. 1. The solutions of each of the eight tasks are also presented with explanations of the key features and methods for accomplishing the task, including vision algorithms.

2 System of DRC-HUBO+

2.1 Robot Platform

The humanoid robot platform, DRC-HUBO+, was developed for the DRC Finals by Rainbow Robotics and the HuboLab. It contains all the technologies developed since the first generation of HUBO, KHR-1, was developed (Kim et al. 2002; Park et al. 2005). It especially was designed based on what Team KAIST learned and

A version of this article was previously published in the Journal of Field Robotics, vol. 34, issue 4, pp. 802–829, © Wiley 2017.

J. Lim (✉) · H. Bae · J. Oh · I. Lee · I. Shim · H. Jung · H. M. Joe
O. Sim · T. Jung · S. Shin · K. Joo · M. Kim · K. Lee · Y. Bok · D.-G. Choi
B. Cho · S. Kim · J. Heo · I. Kim · J. Lee · I. S. Kwon · J.-H. Oh
KAIST, 291 Daehak-Ro, Yuseong-Gu, Daejeon 34141, South Korea
e-mail: yjs0497@kaist.ac.kr

M. Spenko et al. (eds.), *The DARPA Robotics Challenge Finals: Humanoid Robots To The Rescue*, Springer Tracts in Advanced Robotics 121,
https://doi.org/10.1007/978-3-319-74666-1_2

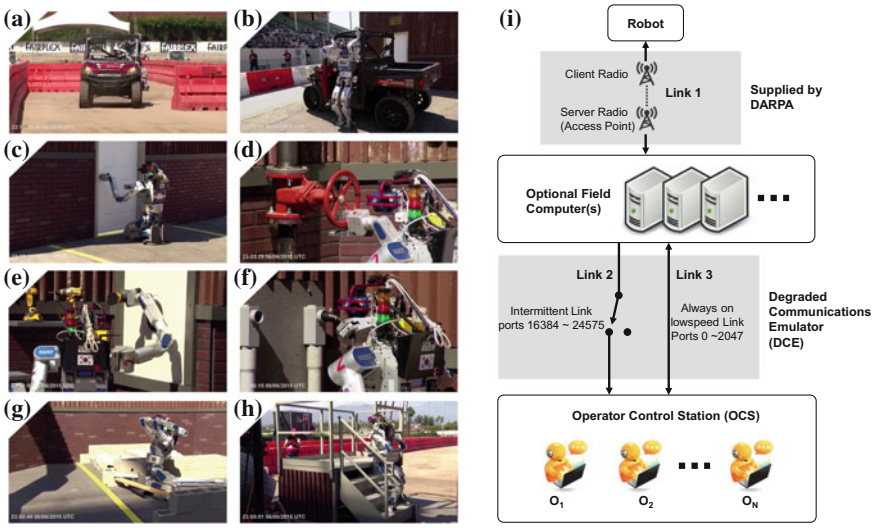


Fig. 1 Tasks given at the DRC: **a** Drive **b** Egress, **c** Door, **d** Valve, **e** Wall, **f** Surprise, **g** Rubble, and **h** Stair tasks. **i** Simplified logical diagram of key communication links. There is a wireless link between robot and field; DCE exists between field and OCS

experienced at the previous competition, the DRC Trials (Lim and Oh 2015; Oh and Oh 2015). In the following subsections, the key features of DRC-HUBO+'s hardware are briefly described.

2.1.1 DRC-HUBO+ Robot Platform

As can be seen in Fig. 2, DRC-HUBO+ has been designed as a humanoid because the given tasks are all concerned with human environments and, consequently, human morphology will bring an advantage to the tasks. At the DRC Trials, it was found that a wheel based robot had an advantage in conducting indoor tasks on flat ground. To take advantage of this situation, DRC-HUBO+ can select two types of mobility mode by transforming the posture of its legs (see Sect. 4.1.1). It can travel on flat land using wheels attached to the knees; it can also walk and traverse rubble and stairs using its two legs. As can be seen in Fig. 2b, this robot has an air-cooled heat dissipation system of actuators at pelvis pitch, knee pitch, and ankle pitch that can generate enough power for the robot to walk or change mobility mode. This system can endure 1.7 times the maximum-continuous-current which is specified in the motor pecification-sheet.

To secure enough rigidity to protect against deflection due to the robot's weight, the DRC-HUBO+ designers used an exoskeletal structure and tried to avoid cantilever forms. This design strategy also reduced the thickness and weight of each limb. For the convenience of motion planning, this robot arm has 7 degrees of freedom (DOFs).

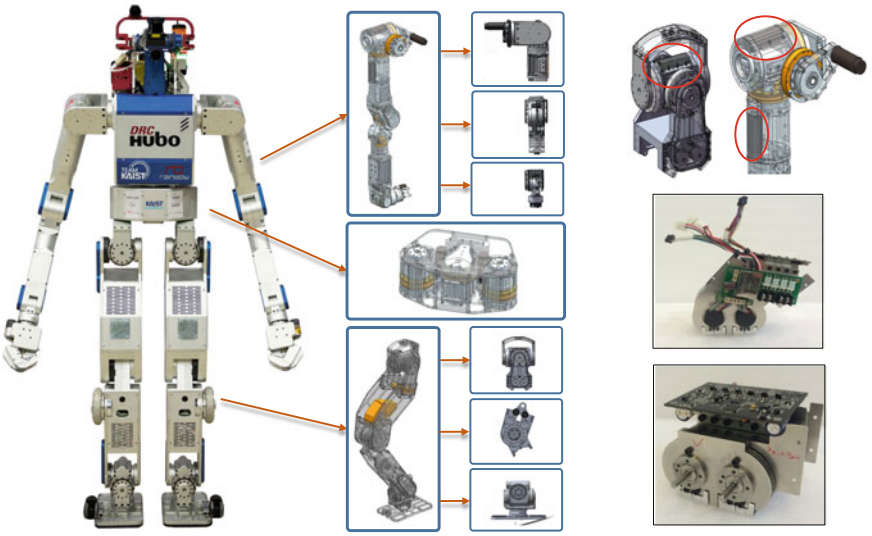


Fig. 2 DRC-HUBO+ robot platform. **a** Modular design of the robot. Each joint of the robot is easily changeable. **b** Air cooling design and conduction of motor heat to the frame

This redundancy is used to determine the angle between the torso and elbow. The last joint of the arm is wrist yaw, and can rotate infinitely due to a slip ring. The waist of the robot can be rotated up to 720°, allowing the robot to cover the whole surrounding area without changing its standing position. Additionally, to prepare for accidents or modifications of the hardware, all robot joints are modularized into chunks, so that replacement of a problematic joint will suffice to fix the robot.

Table 1 shows the basic specifications of DRC-HUBO+. With two fully charged batteries, DRC-HUBO+ can operate normally for about 4 h. It has a force/torque (FT) sensor at the end of each of its limbs, and one inertia measurement unit (IMU) sensor and one fiber optic gyro (FOG) attached to the pelvis. Two optical flow sensors are attached at the shins.

2.1.2 Vision System of DRC-HUBO+

For the vision hardware, two cameras (Point Grey Flea3-GigE with 2.2 mm lens, 1288 × 964 pixels) and one Light Detection and Ranging (LIDAR) (Hokuyo UTM-30LX-EW, 180° scanning angle with 0.25° angular resolution) are used. The main camera, placed below the 2D LIDAR, is used to carry out all but the driving task, which is accomplished by the streaming camera placed on head guard (red bar). The streaming camera is also used to monitor the surrounding environment (Fig. 3).

Table 1 Basic specifications of DRC-HUBO+

Height (cm)		170
Weight (kg)		80
DOF	Head	1
	Arm	2 Arms × 8
	Waist	1
	Leg	2 Legs × 7
	Total	32
Payload (kgf)	Arm	10
	Fingertip grip	3
	Encompassing grip	10

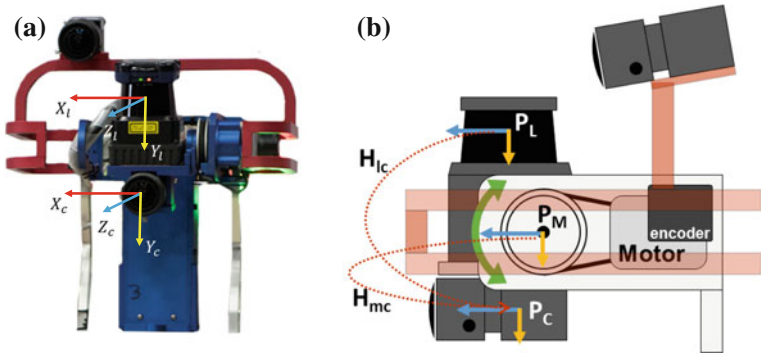


Fig. 3 Vision system configuration for DRC-HUBO+ and an example of its movement. DRC-HUBO+ has two cameras and one 2D LIDAR sensor installed on its head, allowing it to obtain color images and depth information

2.2 Software Architecture

To build the robot software system, a total of seven computers are used. As can be seen in Fig. 4, two computers are inside Robot, two are in Field, and the remaining three are in OCS. Robot-Motion is used to control the motion of the robot by communicating with hardware devices and executing commands from Field or OCS. Vision-Grabbing obtains environment visual data and sends them to Vision-Field. Vision-Field has several vision algorithm modules for calculating 3D point clouds and can obtain high quality depth information and pose of target objects using the point cloud within the given Region of Interest (ROI) in a 2D image. Motion-Field is located at the center of the communicational links; it prioritizes data, and controls the flow of data. It also controls the operational flow of each task. The three OCS computers are able to interact directly with the operators. OCS-Main gives instructions to Motion-Field to control the other computers. OCS-Virtual is used to check the status of DRC-HUBO+, including such aspects as the robot posture, sensor data, point cloud, and

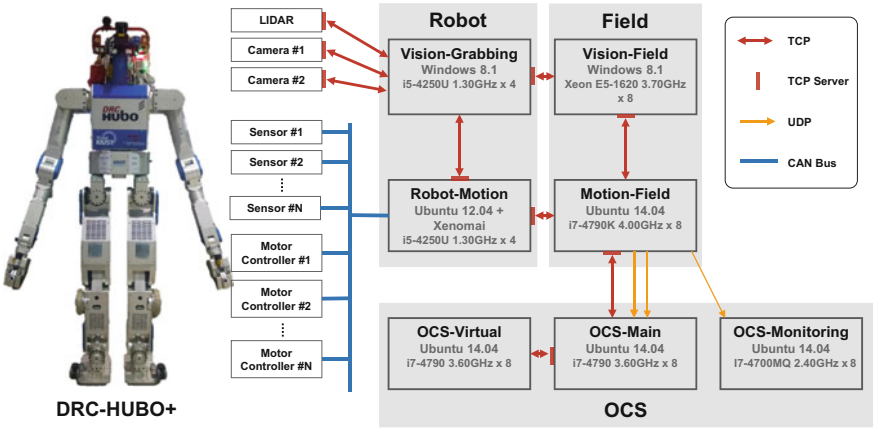


Fig. 4 System configuration of DRC-HUBO+ and its communicational links. There are Robot, Field, and OCS domains; they use a total of seven computers

images. OCS-Monitoring is used to observe the status of the processes working inside Robot and Field. OCS-Monitoring can also remotely turn on and off each process (Lim et al. 2015).

2.2.1 Real-Time Robot Controlling Framework: PODO

In this subsection a real-time robot controlling framework named PODO (which means "grape" in Korean) is introduced. PODO is expandable to both software and hardware. It provides users with multiple processes called ALs (which means "grape berries" in Korean); processes are independent of each other. All users can possess their own ALs, and these ALs' main threads can be turned on and off by PODO depending on whether the users have hardware access authority or not. Thus, even if a lot of ALs are operating at the same time, the computational resources that ALs practically use are not excessive. For this reason, PODO can accommodate a group of ALs; this system can be said to be software expandable.

There is a special process called Daemon that directly accesses the hardware. The hardware information is abstracted and categorized, and this information is located between Daemon and the ALs as Shared Memory (see Fig. 5); this Inter Process Communication (IPC) method is chosen because it is very fast and easy to use, and there is no increasing of the complexity regardless of the number of connected processes (Dantam and Stilman 2012). Thus, ALs can control the hardware indirectly through the Shared Memory and through Daemon. This means that users do not have to know the details of how to communicate with and manipulate the hardware devices. The only thing that changes when the hardware is modified or expanded is the Shared Memory: ALs and Daemon are not affected by hardware changes.

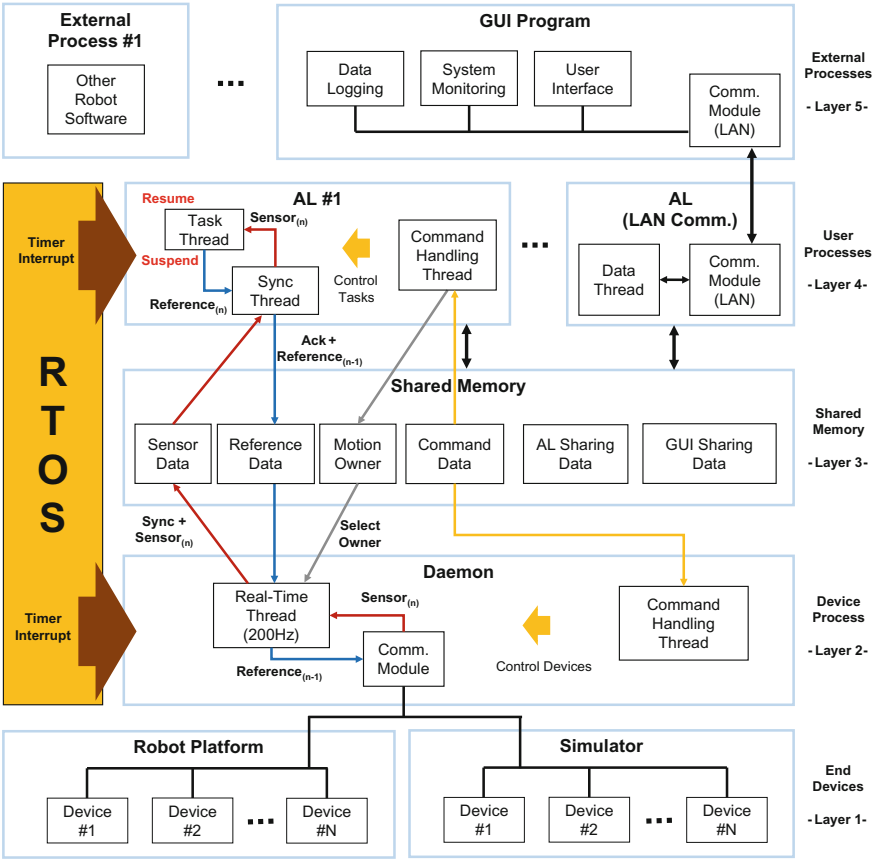


Fig. 5 Software architecture of PODO. Software consists of five hierarchical layers: end devices, device process (Daemon), shared memory, user processes (ALs), and external processes. The real-time thread generated by RTOS makes it possible to control the end devices in real-time

Daemon has a real-time thread to control the hardware accurately; Xenomai (Xenomai | Real-time framework for Linux 2015) is used to generate the real-time thread. The frequency of the thread was set at 200 Hz for walking and controls; the communicational and computational resources were sufficient at this frequency. To prevent early or late updating of joint references and sensor values, PODO suggests synchronization of threads between Daemon and ALs, as can be seen in Fig. 6.

Daemon updates the sensor data in the Shared Memory, and sends synchronization (Sync) messages to ALs. The ALs keep watching these Sync messages and read the sensor values when the Sync status changes. After that, the ALs update the joint references that were calculated in the previous step. At this time, acknowledgement (Ack) messages are also sent to Daemon via the Shared memory. The ALs can calculate their algorithms and generate joint references with the sensor values after

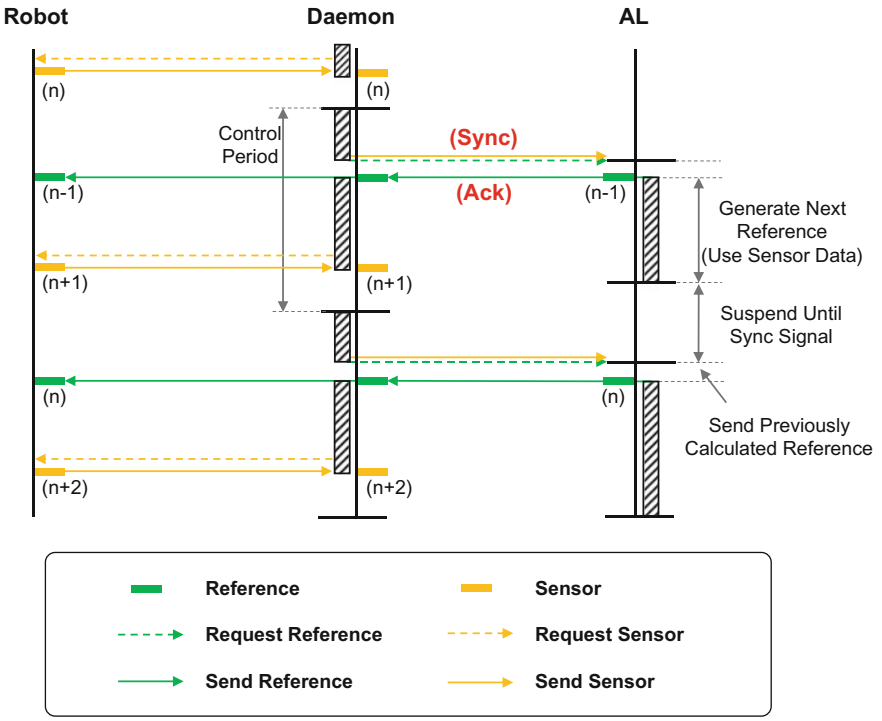


Fig. 6 Logical diagram of synchronization process between Daemon and ALs. The ALs get sensor data when a Sync message arrives; simultaneously, they send an Ack message to Daemon with new reference data

sending an Ack signal. Because time gaps between the obtaining of a current Sync message and of the next Sync message are almost identical and periodic, the time allowed to calculate a joint reference is almost the same as the control period of Daemon. While the ALs are working, each AL recognizes an Ack signal and proceeds to the next step. ALs send the received joint references to Daemon and then request the sensor's next data. Through this procedure, ALs synchronize with Daemon.

Figure 7a presents the control period of the real-time thread in Daemon; it shows that the jitter of the thread is about 10 ms. Figure 7b indicates the time consumed by the synchronizing process. It takes about 40 ms regardless of the number of ALs working; this value is less than 1% of the control period. These results indicate that the proposed architecture and synchronization algorithm ensure the real-time capacity of the controlling hardware devices.

Similar to PODO, a Vision Process Connector (VPC) was developed for acceleration of the development speed and maintenance of the program. It has a structure similar to that of PODO, but it uses ZeroMQ (Hintjens and Pieter 2013) as a communicational method between the core process and the other sub-processes, which have vision algorithms such as object extraction and recognition.

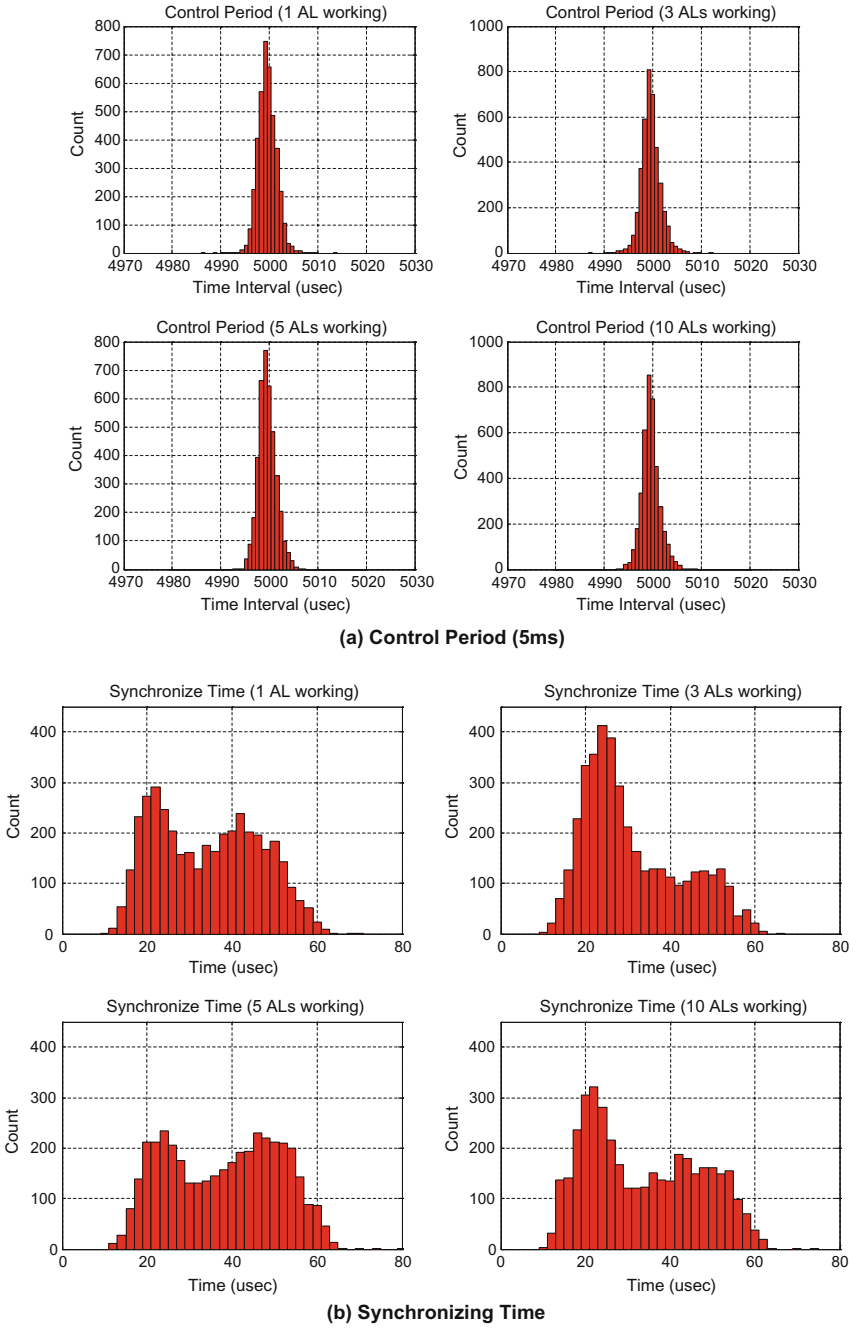


Fig. 7 **a** Histogram of the control period depending on the number of ALs. **b** Histogram of the synchronizing time depending on the number of ALs

2.2.2 Communication Strategy and Process Monitoring

The communication line between the robot and Field, Link 1, transmits at 300 Mbit/s; Transmission Control Protocol (TCP) is used because this bandwidth is sufficient for the suggesting system. In the cases of Links 2 and 3, to prevent the influence of degraded communication links, a stable and efficient communication strategy was proposed.

For Link 2, there are two individual processes to help developers: one is located in Motion-Field and the other is in OCS-Main. These two processes are in charge of the stable transfer of the burst data. The process in Motion-Field checks the latest data, compresses it, and keeps sending it to DCE. Then, the corresponding process in OCS-Main receives the data from DCE when Link 2 is opened. This process gathers incoming data until it makes a complete package. Then, it decompresses the data and sends all acquired data to the main program of OCS-Main. The only thing developers need to do is send the desired data to the process of Motion-Field; then, the data will be available at the process of OCS-Main after Link 2 has been opened.

Link 3 always permits communication in both directions if the amount of transferred data does not exceed 9600 bit/s. Because the command data and status signals of every process in DRC-HUBO+ always work without communication conditions, these are transmitted using Link 3. To guarantee data completion, the command data is transmitted by TCP; the status signals for monitoring and checking that are used are the User Datagram Protocol (UDP) from Motion-Field. The main program in Motion-Field also checks the entire amount of transferred data and manages the transfer order according to the data priority.

By using these communication strategies, it is possible to send and receive data through DCE at DRC. The status of all processes and hardware parts can be observed at the operating site. Additionally, AlwaysUp (AlwaysUp | Run Any Application as a Window Service 2015) and Supervisord (Supervisord | A Process Control System 2015), were used to manipulate the process remotely.

2.2.3 Supervised Autonomy System

Disaster tasks are actually practical issues, and the DRC tasks also need to be approached using a realistic method rather than an experimental one. For a 100% guaranteed perception solution, Team KAIST decided to use human recognition ability instead of an autonomous perception algorithm (Cheng and Zelinsky 2001).

Human operators can receive information such as robot status and environment data, command robot to operate, and intervene while the robot is operating. These three actions are called Reception, Instruction, and Interruption (see Fig. 8). Each task consists of task motion and moving motion, and each motion can be operated using certain environment and target parameters at the start time. It is possible to judge the success or failure of the motions, so the robot can retry or request further information. During these procedures, there is no need for any additional user intervention, and so this combination is called “Robot level autonomy.”

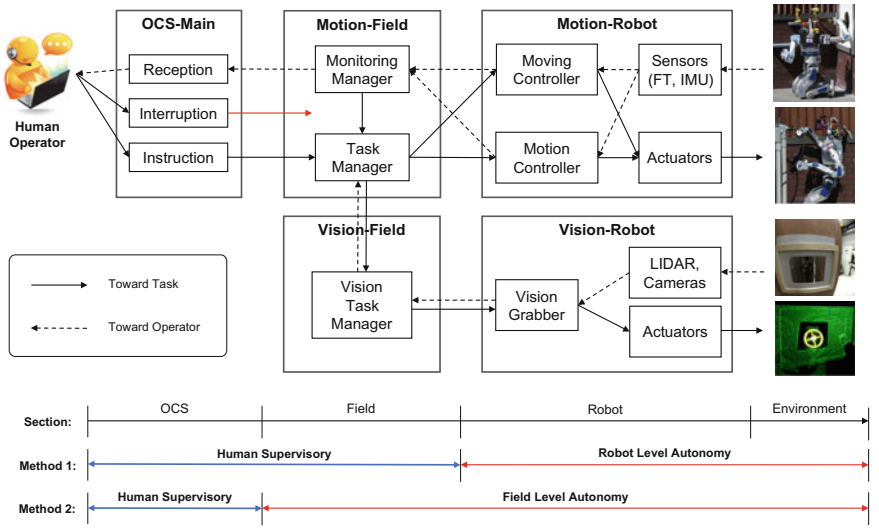


Fig. 8 Command flow of the DRC-HUBO+ system; autonomy level is also shown. There are two combinations that can be used: the first is human supervisory and Robot level autonomy, the second is human supervisory and Field level autonomy. There is no fully autonomous control

This robot level autonomy can be expanded to Field; this is called "Field level autonomy." Motion-Field receives an instruction from the operator and starts the state machine of the appropriate task with the given values. These given values are the task type and method, and the ROI of the target object. According to the state, the whole robot system conducts the task gradually inside Field and Robot.

3 Control and Vision

3.1 Whole Body Inverse Kinematics

This subsection details the inverse kinematics (IK) solution for DRC-HUBO+. A singularity can occur when the robot generates walking patterns and motions. To overcome any difficulties encountered near kinematic singularities, the exact inverse problem is reformulated as a damped least-squares problem that balances the error in the solution against the size of the solution. The use of the damped least squares was first proposed by Wampler and Charles (1986). Nakamura et al. introduced the damping factor, which chatters in the vicinity of the singular points; they also proposed the singularity-robust inverse matrix (Nakamura and Hanafusa 1986). Wampler and Charles proposed an IK solution based on the Levenberg-Marquardt (LM) method (Levenberg 1944; Marquardt and Donald 1963).

The IK solution of DRC-HUBO+ is based on the LM method, and it is assumed that there are two operational situations: 'stance' and 'walking.' Thus, the IK has two modes: one is for the lower body while the upper body is fixed; the other is for the upper body with fixed foot placement. The robot has a total of 33 tasks: the center of mass (CoM) position in the horizontal plane (2), the arm angle (1 × 2) (Shimizu et al. 2008) and hand tasks (6 × 2), leg tasks (6 × 2), pelvis height (1), pelvis orientation (3), and waist angle (1). Consequently, the number of robot tasks is 33 and the number of joints is also 33.

The IK for the lower body while the upper body is fixed is developed based on the conventional LM method. In most cases, CoM positions among the robot tasks are hard to bring to convergence when the updating rules are iterating because, in order to reach the desired position, all robot tasks except for the CoM must calculate the joint displacement for every iteration, and mass distribution occurs at every iteration. In short, there is a conflict between the tasks that must be solved to reach CoM position and the hand position. Generally, the CoM position tasks adjust with the pelvis placement. In this sense, the IK for the lower body has an advantage to attain the convergence of the CoM position. Due to the fixed upper body with regards to the pelvis position, the pelvis position, which is the most effective position to use to reach the desired CoM, is not influenced by the upper body joints. Actually, the number of iterations in this process is lower than those for the non-fixed upper body IK. Therefore, there is no problem in developing the IK for the lower body.

However, the IK for a non-fixed upper body requires a large number of iterations. This brings a large computational cost and that is a burden for the real-time system; indeed, this is one of the most important issues for humanoid robots. The many computations required for walking pattern generation, motion planning, sensor based controllers, etc., need to share the computational resources. In this way, there is insufficient computing time for the IK. Hence, the focus was on reducing the number of iterations, and on the proposal of a new IK.

First, a lumped mass on the hand was created to equal the mass of the arm and to remove all mass distribution of the arm. Then, the mass of the arm is only at the hand position. The mass of the arm belongs to the total CoM, which is one of the IK task values of the forward kinematics, which is conducted for every iteration loop to minimize the square of the error. The position of the lumped mass on the hand is predetermined at desired hand position, as can be seen in Fig. 9. In this way, the displacements of the arm joints do not affect the CoM position. Consequently, this method brings a fast convergence of the CoM position tasks. Figure 10 provides a comparison of the number of iterations required for the conventional LM (Wampler) method and for the proposed method. However, there will exist a slight amount of error between the real CoM position and the modeled CoM position because it was assumed to be a lumped mass. Hence, a cooperative balancing controller was used to keep the stability of the humanoid during the motion (Lee and Oh 2015).

The 7-DOF arm consists of end-effector positioning (3), orienting (3), and arm angle (1), which is defined in (Shimizu et al. 2008). Operators could undergo trial and error due to the joint limit of the arm when they generate arm motion via tele-operation; this trial and error delays the mission time. Hence, a simple algorithm

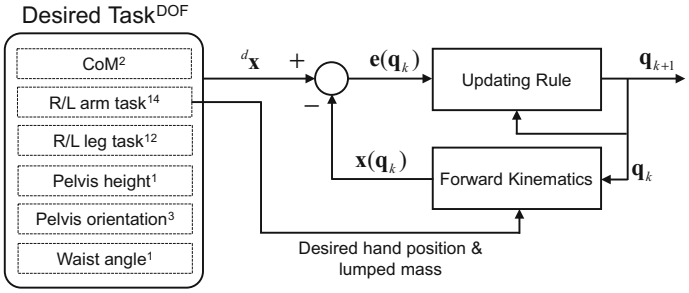


Fig. 9 Block diagram of the IK for the DRC-HUBO+. The joint value is updated at every iteration by the Hessian matrix and the error. The error is obtained according to the desired task and the task output, which are the results of the forward kinematics, in which the hand position is taken from the desired task

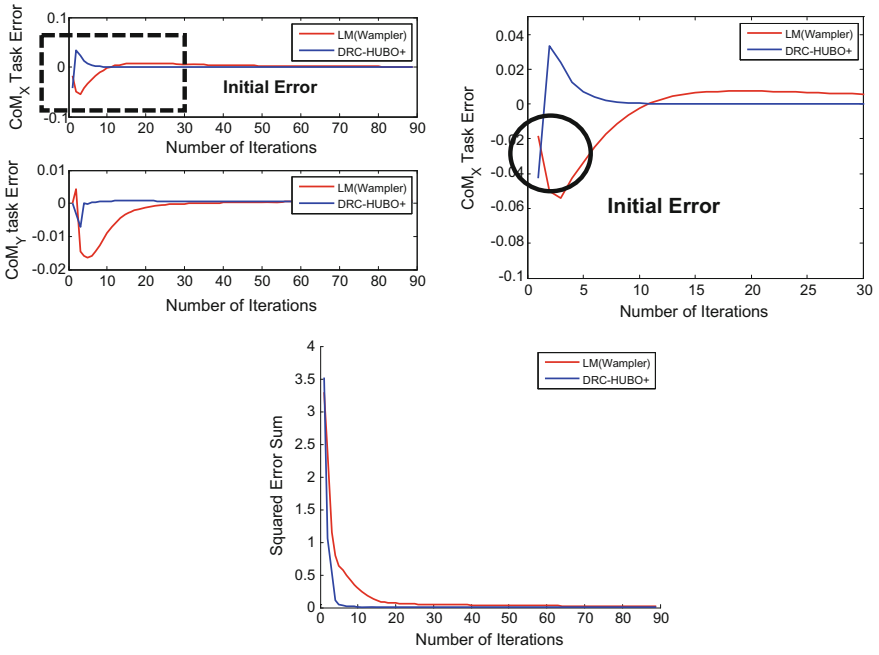


Fig. 10 Task error comparison results between LM (Wampler) and IK for DRC-HUBO+. The first result on the left shows the task error for the CoM position in the horizontal plane. Because the IK for DRC-HUBO + fixes a lumped mass on the desired hand position, the initial error of the CoM on the X-axis is larger than those found in previous works. The center results show the initial error clearly. Though the initial error is larger, the error converges faster than it does in LM (Wampler). The last results show the squared error sum

was developed to generate the trajectory of the arm angle automatically according to the end-effector trajectories. Actually, this issue is similar to the redundancy issue. There are many redundancy resolved solutions to minimize energy and displacement and to avoid joint limits, collisions, etc. A cost function ω is defined in the following Eq. 1; this function is related to the joint limits and is used to select the optimal arm angle via direct searching.

$$\omega = \sum_{i=1}^{7} \frac{\left(\theta^i - \frac{1}{2}\left(\theta^i_{\max} - \theta^i_{\min}\right)\right)^2}{\left(\theta^i_{\max} - \theta^i_{\min}\right)^2} \tag{1}$$

where $\theta^i_{\max}$ and $\theta^i_{\min}$ are the upper and the lower limit of the joint, respectively, for each joint ($i = 1 \sim 7$). This arm angle selection supports the generation of motion for the surprise task.

3.2 Compliance Control

Each joint of the DRC-HUBO+ is controlled by an electric motor with a highly geared harmonic drive. Thus, it has high friction and exerts a damping force on each joint. Additionally, the motor controller transmits only pulse width modulation (PWM) to the motors. Therefore, it is hard to generate joint torque equal to a given reference torque. To achieve compliance control, a hybrid position/force controller based on the computed torque method was devised.

3.2.1 Gain Override Technique

The motor controller has a Gain Override mode that changes the position-derivative (PD) servo controller gain. When the robot contacts the environment, small position errors cause high internal forces, and such situations can cause damage to the robot. In the DRC Trials in 2013, Team KAIST applied the Gain Overriding method to deal with this problem (Lim and Oh 2015). Using this technique, the robot can be protected from fractures in a multi-contact state, and its power consumption decreases. The amount of change of the gain is determined by experiment, so it cannot be specified all the time.

3.2.2 Non-complementary PWM Switching Mode

To obtain flexibility in the highly geared joint system of DRC-HUBO+, a control method called Non-Complementary Switching was adopted; this method removes the braking effect. In the motor control technique using an H-bridge, there are four Torque-Speed Quadrants, as can be seen in Fig. 11.

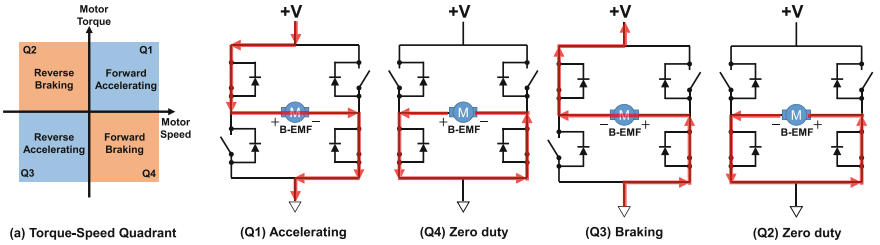


Fig. 11 **a** Torque-Speed Quadrant of motor drive. (Q1–Q4) 4-Quadrant unipolar drive. When positive duty is engaged on the motor, it accelerates in a positive direction (Q1). If zero duty is engaged while the motor is rotating in a positive direction, the motor speed is still positive but it decelerates (Q4). When negative duty is engaged on the motor, it accelerates in a negative direction (Q3). If zero duty is engaged while the motor is rotating in a negative direction, the motor speed is still negative but it decelerates (Q2)

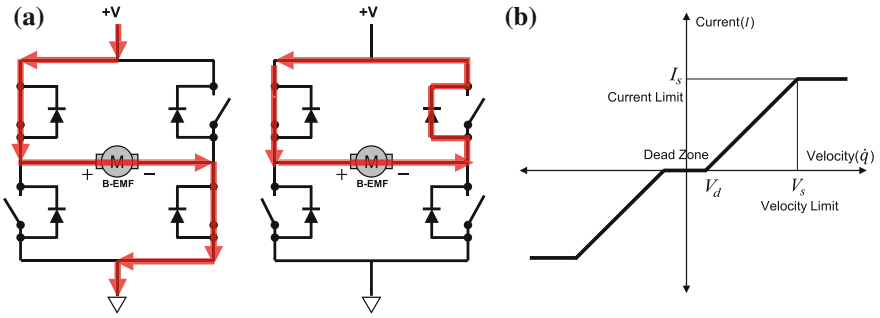


Fig. 12 **a** 2-Quadrant unipolar drive. In this motor drive circuit, motor does not experience back EMF during zero duty (zero voltage). When zero duty is engaged on the motor while motor is rotating, the motor does not decelerate in any direction. **b** Modeled friction compensation. Friction in the joint is modeled as linear damping friction, which friction is proportional to the joint velocity

Conventional motor driving mode is Complementary PWM Switching mode, which uses a 4-Quadrant unipolar drive (Q1, Q2, Q3, and Q4). This PWM switching technique has the advantage of fast braking, so it is suitable for position control of the robot manipulator. Figure 11 (Q1–Q4) shows a 4-Quadrant unipolar drive circuit of the Complementary PWM Switching mode.

Different from the Complementary PWM Switching mode, the Non-Complementary PWM Switching mode uses only a 2-Quadrant unipolar drive (Q1 and Q3). Figure 12a shows only the forward accelerating (Q1) and zero duty situations. Reversing the sign of the voltage will make the motor reverse accelerate (see Fig. 11a). With the Non-complementary PWM switching mode, the motor does not experience the braking effect. Joints are able to act like frictionless joints; friction is only present in the reduction gear.

3.2.3 Joint Friction Compensation

A high reduction ratio speed reducer like a harmonic drive has a high friction force when it experiences back drive. To compensate for the force of this friction on each joint, the motor controller provides a friction compensation open-loop controller. The friction force can be modeled as a linear damping force that is proportional to the joint velocity. By performing experiments, the model parameters, I_s, V_s, and V_d, can be defined for each joint (see Fig. 12b). After these parameters are defined, each motor controller generates additional current to compensate for the friction.

3.2.4 Gravity Compensation

To apply the computed torque method, the effect of gravity on the robot arms should be canceled. From the DRC-HUBO+ arm kinematics and design data, such as the mass center of each link, the necessary torque for each joint can be calculated with the assumption that the upper body is standing upright.

3.2.5 Hybrid Position/Force Control

In the DRC Finals, DRC-HUBO+ was able to control the position of the manipulators as well as to generate force on the end effectors. These abilities allowed the robot to stay stable in a multi-contact state. To achieve this type of precise control, hybrid position/force control based on the computed torque method was devised.

In most cases, the arm does not need to move rapidly. This means that neither arm joint experiences high angular acceleration or velocity. Thus, a static condition can be assumed and, by using the Virtual Work Principle (Tsai 1999), each joint torque that generates the end-effector force in each direction can be obtained.

$$F = \left[f_x, f_y, f_z\right]^T \tag{2}$$

$$\tau = [\tau_1, \tau_2, \cdots, \tau_n]^T \tag{3}$$

$$\tau = J^T F \tag{4}$$

When the robot grabs certain structures, the hand orientation is fixed, so only the position of the hands X needs to be considered:

$$X = \left[x_p, y_p, z_p\right]^T \tag{5}$$

The block diagram in Fig. 13 shows the entire control procedure. As stated before, only the quasi-static condition is considered; there is no dynamic model or acceleration term. By solving the forward kinematics, the end-effector's Cartesian information is calculated, and a simple Cartesian PD controller can be designed.

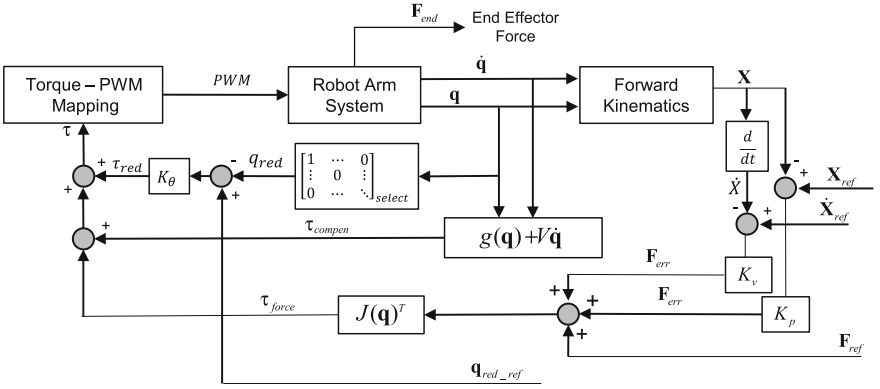


Fig. 13 Control block diagram of hybrid position/force control. From the position and velocity feedback of the end-effector and the desired force, the force that needs to be generated on the end-effector is calculated. DRC HUBO+ has a 7 DOF arm, and so a redundant joint should be considered. In this case, the shoulder roll joint was selected to be the constrained joint

$$F_{err} = K_p \left(X_{ref} - X\right) + K_v \left(\dot{X}_{ref} - \dot{X}\right) \tag{6}$$

Not only Cartesian space position control, but also Cartesian force control is needed. Thus, the desired Cartesian force F_{err} is added to generate F_{total}:

$$F_{total} = F_{err} + F_{ref} \tag{7}$$

The general dynamic equation of the manipulator can be expressed as in Eqs. 8 and 9.

$$\tau = M\left(q\right)\ddot{q} + h\left(q, \dot{q}\right) + V\dot{q} + g\left(q\right) + \tau_{ext} \tag{8}$$

$$\tau_{ext} = J^T\left(q\right) F_{ext} \tag{9}$$

Here, τ_{ext} is the external torque, and F_{ext} is the external force acting on the end-effector. From the computed torque and the total force reference, the total torque input can be expressed as in Eq. 10. The computed torque consists of friction compensation and gravity compensation.

$$\tau_{input} = \hat{V}\dot{q} + \hat{g}\left(q\right) + J^T\left(q\right) F_{total} \tag{10}$$

From Eqs. 8, 9 and 10, and by ignoring the inertial term and the Coriolis term, error dynamics can be obtained.

$$F_{ext} = F_{ref} + K_p \left(X - X_{ref}\right) + K_v \left(\dot{X}_{ref} - \dot{X}\right) \tag{11}$$

$$F_{ext} - F_{ref} = K_p E + K_v \dot{E} \tag{12}$$

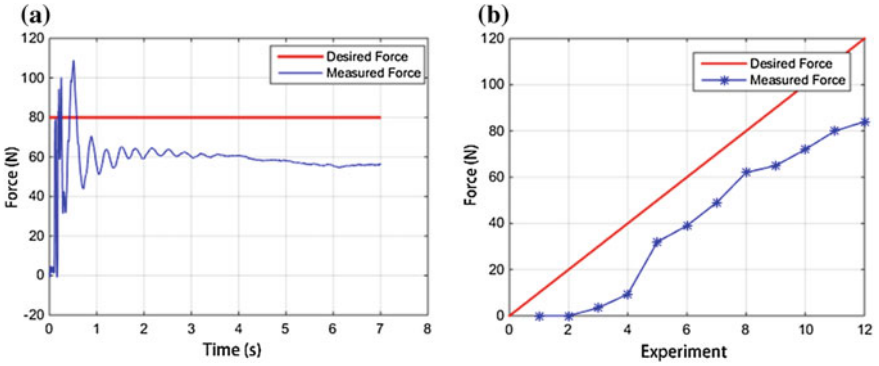


Fig. 14 **a** Time domain response of end-effector force. When desired x direction (front direction) force is given, measured force follows desired force. The oscillation of the measured force is due to the swing motion of the entire robot platform when the robot pushes the rigid wall. Steady state error was about 20 N when the desired force was 60 N. **b** Experimental data for various desired levels of force and for the steady state measured force

Equation 12 is the vector equation; thus, different values of K_p and K_v can be applied in each direction. This is the concept of hybrid control. In addition, different force references can also be applied in each direction. For example, the end-effector can be controlled by the position in the x, y directions and by the force in the z direction.

$$f_{x,ext} = k_{p,x} e_x + k_{v,x} \dot{e}_x \tag{13}$$

$$f_{y,ext} = k_{p,y} e_x + k_{v,y} \dot{e}_y \tag{14}$$

$$f_{z,ext} = f_{z,ref} \tag{15}$$

Equations 13 and 14 are the feedback loop in the x, y direction; Eq. 15 shows that the force produced on the end-effector follows the desired force.

This open loop force control of the robot arm was verified by simple experiment. DRC-HUBO+ was put in front of a rigid wall and was made to push the wall with a certain force. The pushing force can be measured via the FT sensor on the wrist. This experiment was not performed to verify the entire hybrid position/force control algorithm, but to verify the force tracking performance of the end-effector in quasi-static situation.

Figure 14 shows a steady state error in the output force. The system always possesses a certain amount of error because there is no feedback loop. Even when there is an error in the output force, the robot arm is controllable in the Cartesian space with the low gain position control, and it is also possible to attribute compliance characteristics to the arm. To solve DARPA tasks such as the Egress task, exact and precise force control was not required. The most important thing was to make the highly geared arm of the DRC-HUBO+ compliant and, if possible, to track the end-effector force.

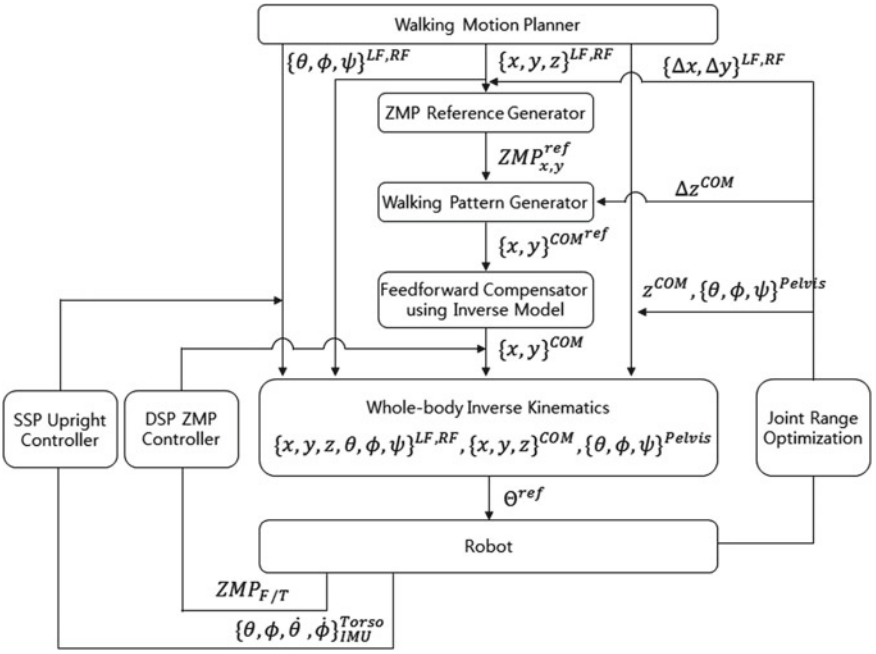


Fig. 15 Overview of information flow in the walking algorithm

3.3 Walking Algorithm

This subsection shows a walking pattern that generates an overall walking motion; it also shows a stabilizer that maintains stability against disturbances. The change of the vertical CoM in LIPM (Kajita et al. 2001) and the compliance of the robot in the walking pattern are considered; a controller that is proper for the position based robot is used as well. The overall walking algorithm for the DRC is shown in Fig. 15.

3.3.1 Walking Pattern

In Preview Control (Kajita et al. 2003), the height of CoM is constant, but the height of CoM must change for the robot to climb the stairs or walk through terrain. The Zero Moment Point (ZMP) error generated from the change of CoM height can be compensated for by applying Preview Control again as a dynamic filter (Stasse et al. 2008). When the robot is walking in place with vertical CoM motion, which is a sine wave, the measured ZMP with compensation and the measured ZMP with no compensation can be represented as shown in Fig. 16. It can be seen that the measured ZMP with compensation is closer to the reference ZMP.

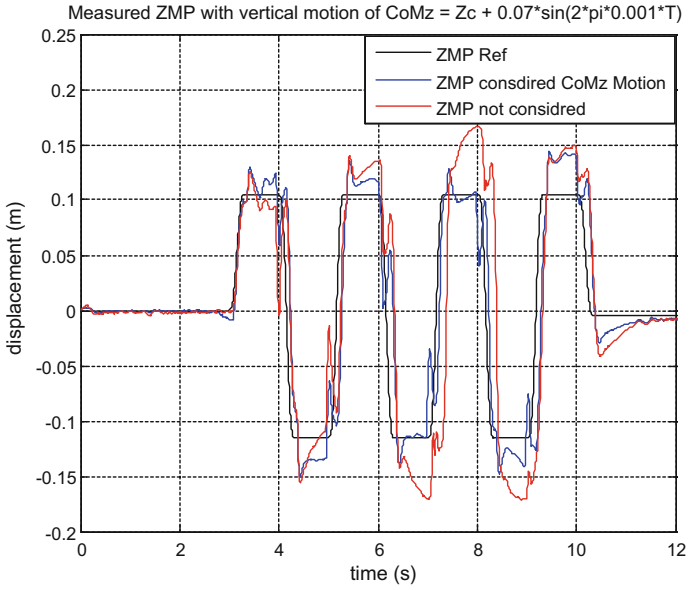


Fig. 16 Measured ZMP with vertical CoM motion. Measured ZMP with compensation is closer to the reference ZMP

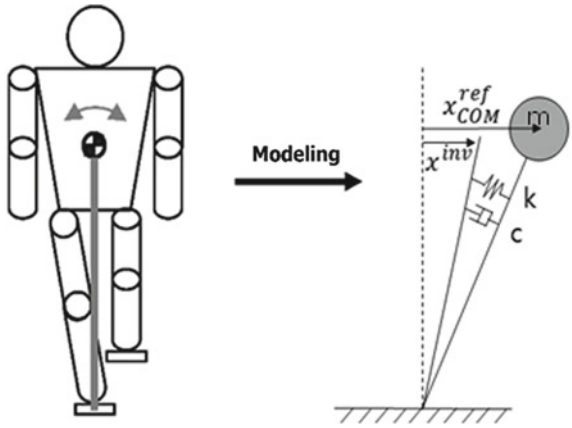
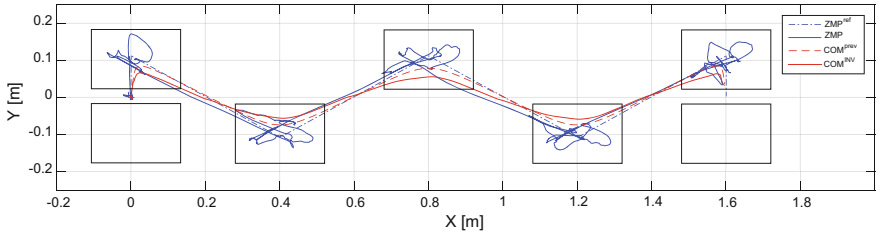
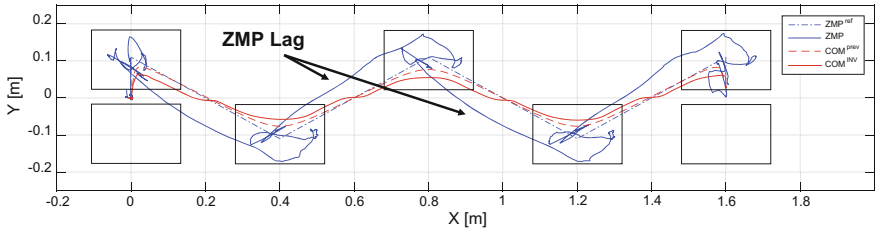


Fig. 17 Linear inverted pendulum with spring and damper

The robot is modeled as linear inverted pendulum with spring and damper and the model is represented in Fig. 17. This model is generally used for feedback controller (Kim and Oh 2013; Cho et al. 2011), but if the step length becomes longer and the mass of robot increases, compliance becomes a larger problem due to deflections of the links. Thus, an inverse model method was devised to modify the CoM trajectory that was generated in the Preview Control. This method applies a feedforward controller for the walking pattern; the transfer function can be calculated as



(a) With feedforward control of inverse model (step length: 0.4m, period: 0.9s)



(b) Without feedforward control of inverse model (step length: 0.4m, period: 0.9s)

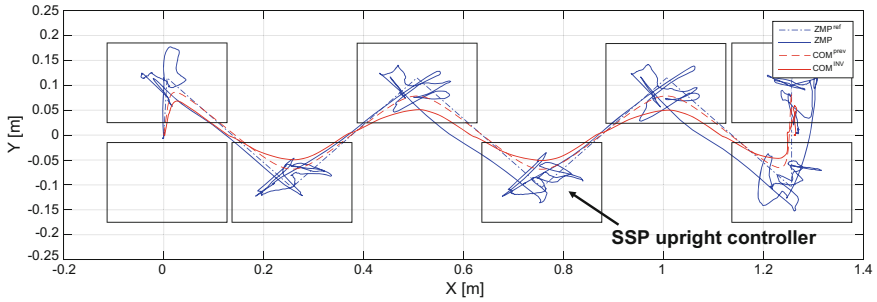
Fig. 18 ZMP and CoM trajectories **a** with and **b** without the inverse model method. ZMP tracks the reference ZMP better with the inverse model method

$$G(s) = \frac{x_{CoM}^{ref}}{x_{CoM}^{inv}} = \frac{\frac{k}{ml^2} + \frac{c}{ml^2}s}{s^2 + \frac{c}{ml^2}s + \frac{k}{ml^2} - \frac{g}{l}} \tag{16}$$

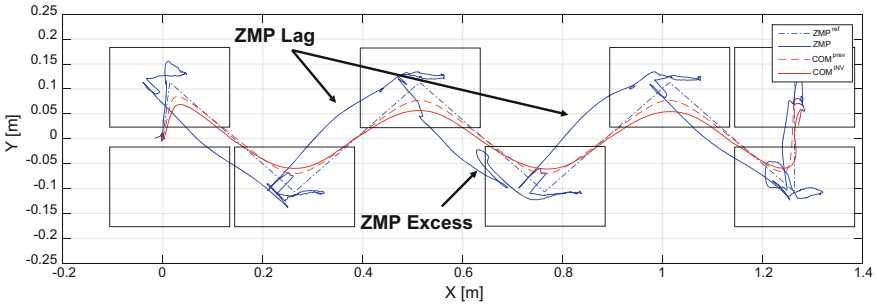
where k, c, m, l, and g are the spring coefficient, damping coefficient, total mass, and acceleration due to gravity, respectively. This inverse model method is very effective at long step lengths, as can be seen in Fig. 18. ZMP tracks better with this method.

3.3.2 Stabilizer

A stabilizer is used to modify the pattern in real-time to maintain walking stability against disturbances. The ZMP controller and the single support phase (SSP) upright controller were designed for the stabilizer. The ZMP controller protects the reference ZMP from the deviation of ZMP (Lee and Oh 2015); the SSP upright controller controls the upper body orientation using an FOG (Kim and Oh 2010). In SSP, controlling the angle and the angular velocity of the base frame is a more effective method of steadying the robot than controlling the ZMP because the supporting polygon is very narrow relative to the double support phase (DSP) and the measured ZMP rapidly saturates to the edge of the foot when a large disturbance is applied to robot. The performance of the SSP upright controller is represented in Fig. 19; ZMP tracks better with the SSP upright controller.



(a) With SSP upright controller (step length: 0.25m)



(b) Without SSP upright controller (step length: 0.25m)

Fig. 19 ZMP and CoM trajectories **a** with and **b** without the SSP upright controller. ZMP tracks the reference ZMP better with the SSP upright controller

3.4 Vision System

As described in Sect. 2.1.2 (Fig. 3a), the main camera and the LIDAR are mounted together on the same rig, which is rotated by a single motor, which enables the sensor system to acquire 3D point data. Through the rotation speed and scope of the motor control, it is possible to adjust the vertical density of the 3D points by trading off the capturing time.

The streaming camera is set up on the top-left side of the head guard. This camera, which is different from the rotating camera, is tilted at a fixed 10°. The main purpose of this camera is to deliver the scene of the field to the operators at 10 frames per second (fps). This real-time image sequences are transferred only when the network burst is activated. Using the images from the streaming camera, the predicted driving trajectory is provided to the operators for driving guidance (see Sect. 4.5).

Images from the main camera are handled differently according to the network mode. Under limited communication, Link 3, the captured image data was resized to one eighth of its original image size, and JPEG image compression was used with a percentage factor of 10% to make the captured image smaller. The highly compressed image is not intended for use by the computer vision algorithms, but

for the operators only. The transferring time for the compressed image in limited communication conditions is about 4 s. This image is transmitted first while the LIDAR is scanning, so that the operators can recognize the surrounding environments and reduce the time needed to set the operator guided ROI. When the burst mode is activated, the captured image data is resized to half and compressed to JPEG format with a percentage factor of 95%.

3.4.1 Calibration

Calibration of the sensor system is essential to make the point cloud useful for robot operations. A coordinate system was assigned to each sensor—three coordinate systems for the camera, laser sensor, and motor—and the relative pose among them was estimated (see Fig. 3). Without loss of generality, the motor's rotating axis was set as the x-axis of the motor coordinate system and the scanning plane of the 2D laser sensor was set as the x-z plane of the laser coordinate system; these decisions were made so that the laser sensor would scan from $-45°$ to $225°$ (total $270°$) of the x-z plane. The intrinsic parameters of the camera are calibrated by a conventional method (Zhang 2000). To achieve a wide field-of-view, a lens with a short focal length is used with a fisheye distortion model (Shim et al. 2015; Lee et al. 2017); or, other methods (Bouguet 2004; Kannala and Brandt 2006; Scaramuzza et al. 2006) can also be used to provide precise results. Figure 3b shows the coordinate setup of our sensor system. The vision system-centric coordinate is set to a coordinate of the motor, P_m. Calibration between the camera and motor is done by capturing a series of images with various poses of a checkerboard pattern in the motor coordinate system. For each pose of the pattern, twelve images are captured at various motor angles, from $-30°$ to $80°$, at intervals of $10°$. The motor-to-camera transformation H_{mc} and all of the pattern-to-motor transformations H_n are optimized using the squared sum of the projection errors of the checkerboard corners p_i as a cost function:

$$f_c\left(H_{mc}, H_1, \ldots, H_N, A_{-30}, \ldots, A_{80}\right) = \sum_n \sum_\theta \sum_i \left\| q_i - proj\left(H_{mc} R_\theta H_n p_i\right)\right\|^2 \tag{17}$$

where R_θ denotes the rotation of the motor (i.e., rotation along x-axis), and $proj\,(\cdot)$ indicates the process of the projection onto images, including both the intrinsic parameters and the radial distortion. The angles $A_{-30} \sim A_{80}$ are also included as variables because the angle at which the motor rotates may contain a small error ($A_0 = 0$ is not included but considered as a reference angle). Both the rotation and the translation of H_{mc} along the x-axis at zero were fixed because the pattern-to-motor and motor-to-camera transformations may compensate for each other. In the same pose at which the images for the camera-motor calibration are captured, the laser sensor scans the checkerboard pattern. The laser-to-camera transformation H_{lc} is estimated by minimizing the distance between the pattern and the points scanned by the laser sensor:

$$f_l\left(H_{lc}\right) = \sum \left(v^T H_{pc}^{-1} H_{lc} q\right)^2 \tag{18}$$

where v and q denote the normal vector of the pattern and the points scanned by the laser sensor, respectively. The pattern-to-camera transformation H_{pc} is computed using the results of the camera-motor calibration.

The results of the calibration are shown in Fig. 20.

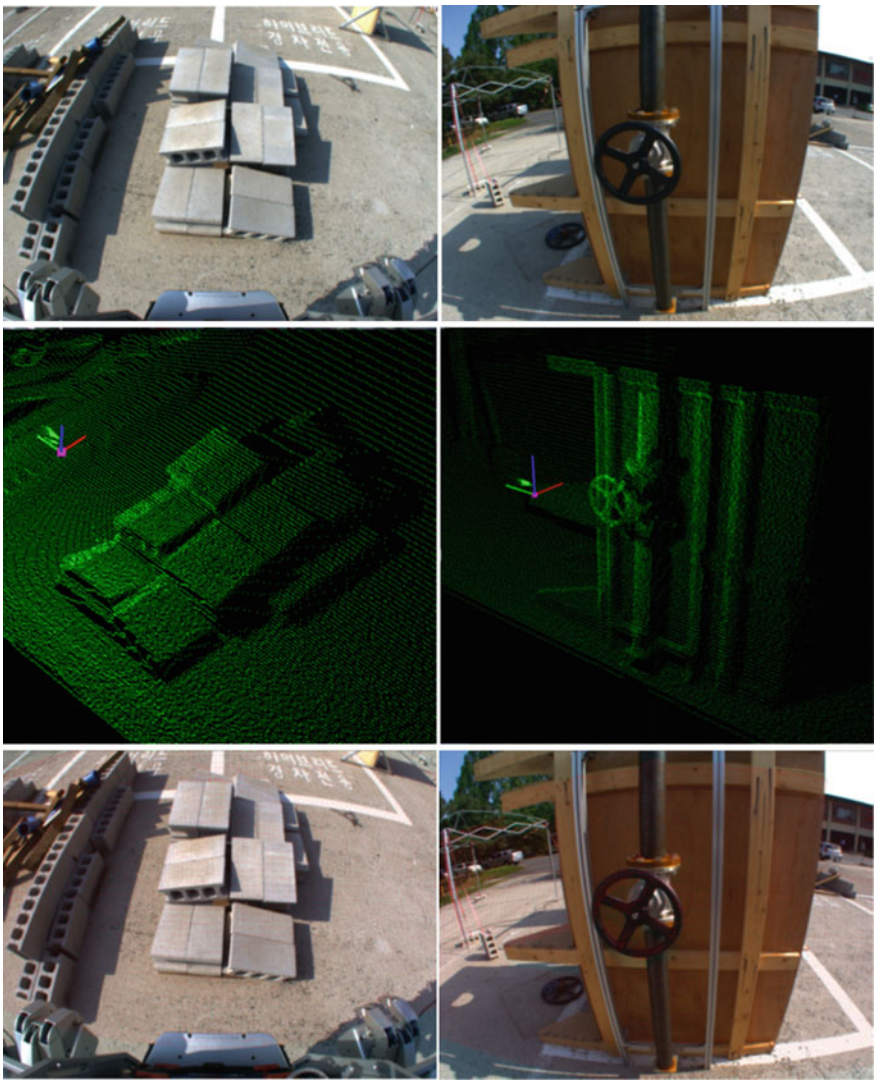


Fig. 20 Data from the camera (top) and laser (center), as well as the 3D cloud data projected onto an image (bottom)

3.4.2 Vision for Tasks

Most of the vision algorithms used in the challenge proceed in the following sequence: 1. Sensor scan, 2. ROI selection, 3. Data upsampling, 4. Object pose estimation, and 5. Robot action. First, both image and 3D point data are acquired by the main camera and the LIDAR. Then, ROI is selected to increase both the speed and accuracy of the algorithm. For this ROI selection, state-of-the-art detection algorithms such as Attention-Net (Yoo et al. 2015) and Region-based CNN (Girshick et al. 2014) were prepared to make a fully autonomous system; manual selection by operators was also prepared to ensure accuracy. Even with the ROI selection, however, object pose estimation with the raw 3D data will often fail due to sparsity of the raw 3D data. So, in order to resolve this problem, a new data upsampling algorithm was designed (Shim et al. 2015) to convert sparse data to dense data without the shadow problems involved in LIDAR scanning. In the next section, the creation of a predicted driving trajectory using the streaming camera for the driving task is described; details are given for the object pose estimation method for the valve, drill (wall), and toehold motions in the terrain (rubble) and stair tasks.

4 Tasks

Operating a robot in disaster circumstances is a practical problem rather than a theoretical issue. It is quite different from operating a robot in a laboratory. Because the DRC imitates a disaster situation, Team KAIST established a strategy in which DRC-HUBO+ would be able to cope with disaster environments. The following four paragraphs present Team KAIST's basic principles for solving the DRC tasks.

For a humanoid robot, the most critical type of damage is that which stems from falling. Even though there has been much research into the technology of humanoid walking, there is still a gap between theoretical research and practical implementation. Furthermore, a disaster environment does not provide even ground. So, DRC-HUBO+ has dual mobility modes: wheel and walking modes. The walking mode has an advantage of allowing the robot to traverse a landscape in which ground level varies, such as an area of rubble. Besides this, when the robot needs to reach its arm higher, the walking mode is required. The wheel mode is good for traversing normal ground quickly and stably. The supporting polygon for the wheel mode is larger than that used for the walking mode, so there is no possibility of the robot falling, and this is an advantage when conducting tasks. According to the ground and the environment, the robot can choose its mode of moving.

The whole robot system should be stable. The robot system is very complicated and large because its every element has an objective and use. One single failure, or the malfunction of a small portion of the robot system, can cause the whole system to fail. So, the whole robot system was stabilized in terms of not only the hardware and software themselves but also in terms of the integration of these two elements.

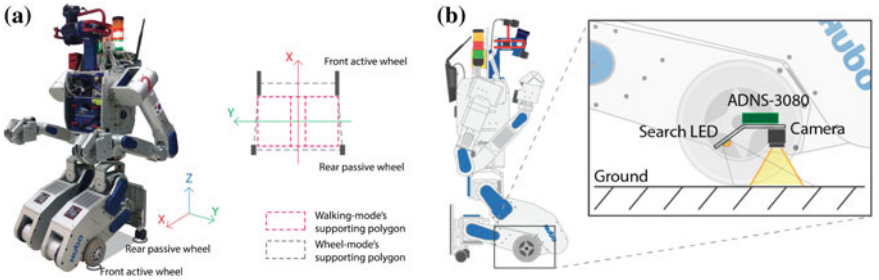


Fig. 21 **a** Wheel mode pose and comparison of supporting polygons used for walking mode and wheel mode. **b** Placement of optical flow sensor, ADNS-3080. Using this optical flow sensor and an IMU sensor on the robot, mobility performance and accuracy are increased

Additionally, built-in recovery routines were developed as backup for cases in which the system encountered unexpected situations.

The eight tasks in DRC are difficult and require a lot of effort to achieve because each task takes up a full section of the research area and demands researchers' profound study. So, no single researcher can develop solutions to all the given tasks within the allotted time. Consequently, the whole system architecture was designed so that it would be possible to solve tasks concurrently, as was shown in Sect. 2.2. Beyond this, the robot platform is very expensive, so not every developer can have access to his or her own robot. Further, it was necessary to be able to easily integrate outputs from each developer. These outputs, of course, must not interfere with others' work. Team KAIST concentrated on these software requirements and put in a lot of effort to realize them.

The autonomous nature of a robot system is one of biggest research areas in robotics. Because of the fast advance of perception algorithms, it is possible to build a fully autonomous robot system. However, it would require still more research to use such a system in a real field condition. The thing that was needed was a 100% guaranteed perception solution; ironically, humans were able to provide the solution. Supervised autonomy was settled on instead of a fully autonomous system.

4.1 Wheel Movement, Door, and Debris Tasks

DRC-HUBO+ has two active and two passive wheels, whose position were decided on so as not to disturb the existing bipedal locomotion of the robot. Figure 21a shows that the supporting polygon of the wheel mode is larger than that of the walking mode. Thus, the wheel mode has an advantage in terms of stability.

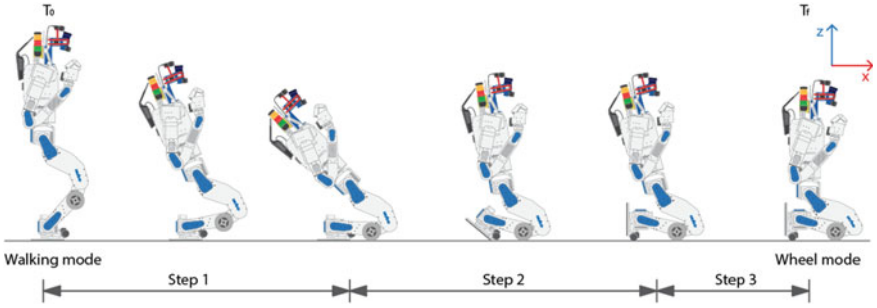


Fig. 22 Posture transformation process between 'walking-mode' and 'wheel-mode'. Process consists of three steps

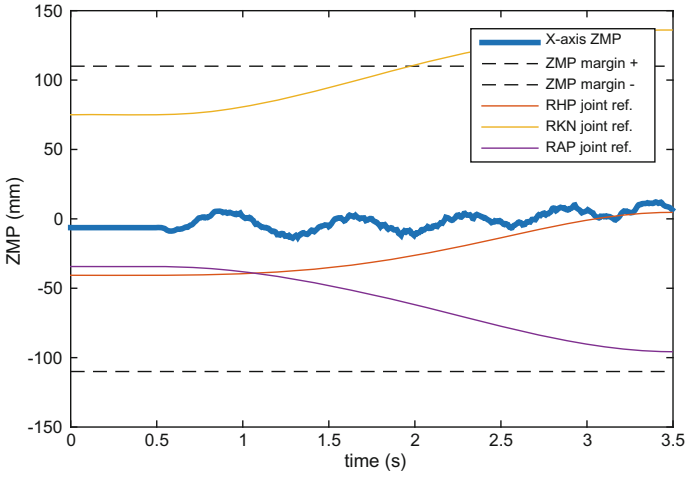


Fig. 23 ZMP variation during posture transformation process Step 1. ZMP is kept at around the zero value by ZMP constrained time optimal control

4.1.1 Posture Transformation Process

The overall transformation process between walking-mode and wheel-mode is illustrated in Fig. 22. This process consists of three steps. Step 1 motion is a transformation motion from the walking mode pose to the wheel mode pose. During this step, the robot sits down and brings the active wheels in contact with the ground. This process is performed by lowering the center of mass while maintaining the dynamic stability of the posture. A ZMP-constrained time optimization problem was formulated to obtain the joint reference angle. Joint references and measured ZMP during this step can be seen in Fig. 23.

Step 2 makes the foot plane vertical to the ground. By doing this, the caster wheel can rotate freely. The final step of this process is performed by reducing the position

control gain of the knee pitch and hip pitch joints. Details of this technique are provided by Lim and Oh (2015). In this step, the redundant joints of the lower body can act as a kind of passive damping suspension. More detail descriptions about this transformation process was provided by our preview work (Bae et al. 2016).

4.1.2 Performance Enhancement Using Redundant Joints and Sensors

When the robot moves in the wheel mode, the accuracy of the wheels drops due to slippage, so DRC-HUBO+ must compensate for slippage according to the measured odometry. For measurement, it uses FOG and two optical flow sensors, ADNS-3080, that is attached to the area below the knee joint, as can be seen in Fig. 21b. These sensors are vertically positioned toward the ground and measure the linear velocity along the calf. Compared to general encoder-based odometry, the robot's movement can be measured more precisely using these two types of sensor.

Each leg has a total of seven joints, including the active wheel joint; all of them are controlled by position control with high PD gain. In the wheel mode, all the joints except for the wheel joint can be used as passive suspension simply by adjusting the Gain Override. This suspension assists in maintaining rigid contact with the ground.

4.1.3 Door Task

In order to open the door, the following scenario was used. First, the robot scans the surrounding environment and detects the door and the doorknob; it obtains information about the width of the door and the knob position by using RANSAC cylinder model fitting. Then, it checks whether the knob is reachable or not. If the knob is not in the arm's workspace, the robot approaches the door and scans again. Conversely, if the knob is in the workspace, the robot grabs the knob. After grabbing, the robot rotates the knob about its axis. During this motion, using the FT sensor on the wrist, the robot checks the type of the door (push or pull door) and the grabbing status. If the grabbing process is successful and the type of the door is detected, the robot starts to rotate the door about the door hinge. Finally, the robot can go through the door using the pre-detected door width information. Consequently, in the wheel mode, DRC-HUBO+ is able to finish the door task in 108 s for a pull door and in 62 s for a push door.

4.1.4 Rubble Task

DRC-HUBO+ can produce a force of 380 N to bulldoze through obstacles. Odometry using FOG and optical flow sensors make it possible to estimate the robot states in real-time. This real-time estimation is used to detect slip conditions; this information is delivered to an operator, allowing the operator to interpret the situation and make a decision.

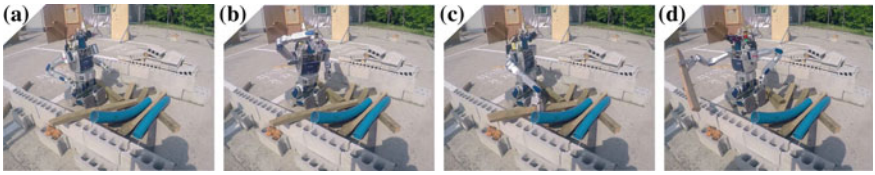


Fig. 24 Snapshot of debris removal motion. After plane detection (**a**), DRC-HUBO+ generates arm trajectories that are collision free (**b**). The grabbed debris (**c**) is removed from the side of the cinder blocks (**d**)

However, simple bulldozing is not a sufficient solution due to several problems. There are some special cases in which the DRC-HUBO+ cannot traverse an area of rubble without removal of the debris. Through the collision free check and the joint limit check, a safe motion is selected by adapting the method that Lee and Oh suggested (Lee and Oh 2015); Fig. 24 provides a snapshot of the debris removal motion.

4.2 Terrain and Stair Tasks

The goal of the rough terrain and stair tasks is to reliably detect local planes and utilize the planes as toeholds for humanoid robot treading. It is possible to detect multi-toeholds and estimate their poses to minimize the necessity of input from the operators. An initial set of segments S is generated by uniformly dividing the ROI of image domain. Each initial segment S_i has 3D points that are selected by the projected points to the image domain and their 3D normal values, which are obtained by RANSAC. The set of segments is expressed as follows: $S = \{S_1, \cdots, S_n\}$, $S_i = \{x_t, y_t, z_t, N_{3\times1}\}$, where S is an initial set of segments, S_k denotes the k-th segment ($k = 1\sim n$), x_t, y_t, and z_t denote 3D points and t is the number of points in the segment, and N denotes 3D normal vector of the 3D points. Then, initial groups were formed. Using the dot product, the initial segments are grouped by the similarity of the normal vectors of the segments. If the normal vector similarity between two segments is larger than a predefined threshold N_t, the two segments are bound to one group. The bounded group updates its own normal vector N. The initial groups are defined as follows: $G = \{g_1, \cdots, g_m\}$, $g_j = \{S_p, N\}$, where j is index of subgroup, S_p is a subset of initial segments, and N is the normal vector of the subset S_p. In this grouping process, two target segments for grouping are selected based on distance between the center points of the segments. Because the initial grouping process is trivial, the initial groups are commonly fragmented or over-segmentation. First, grouping process is run one more time for handling fragmented groups using G; then, over-segmented groups are separated by constructing an undirected graph by fully connecting the nodes using S_i. Each initial segment becomes a node and weight values of edges between two nodes are set to binary values (0 or 1). If an

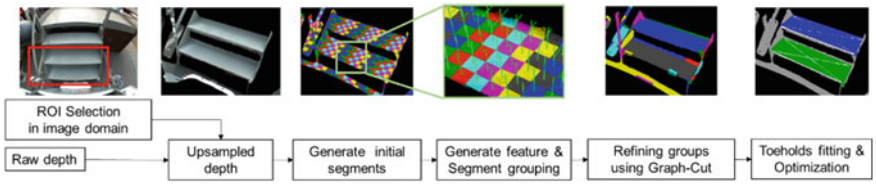


Fig. 25 Overview of the toeholds detection and their pose estimation

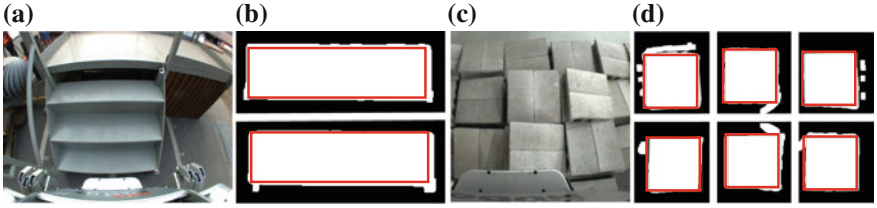


Fig. 26 **a**, **c** Color image. **b**, **d** Detected toehold planes in the projected virtual image domain

edge crosses another node assigned to another group, or if normal vectors of two nodes are smaller than the weight of the edge is set to 0. In other cases, the weight of the edge is set to 1. New group set $\bar{G}$ is regenerated using Graph-Cut (Boykov and Veksler 2006).

To determine toeholds planes in the group set $\bar{G}$, the 3D points in $\bar{G}_J$ are projected to virtual 2D image domain. A virtual 2D image can be generated by using the center of 3D points and N of the $\bar{G}_J$. Using the size and shape of the projected points in the virtual image domain, it is possible to determine which group is similar to the predefined plane model. Finally, for more accurate pose estimation, plane fitting based on RANSAC is used to assign additional 3D points to the selected $\bar{G}_J$ and the updated points are also projected to the virtual image domain (Fig. 25). Figure 26 shows the selected groups in the virtual image domain. Center point and rotation angle of toehold plane are optimized by minimizing the angle difference and the edge distance between the four sides of the predefined toehold model and the edges in the projected image domain. The calculated center point and rotation angle are converted to 3D space coordinates. The whole process can be seen in Fig. 25. If the results of the optimization are not sufficient to our threshold, O_t, our algorithm can accept input from the operators to assign the predefined plane models by force. The operators select three corner points of a target plane in the image domain. A center point and the plane parameters of the three points, are calculated, and inlier points of the calculated plane are collected using plane fitting based on RANSAC. This data set is regarded as selected $\bar{G}_J$; the algorithm extracts the predefined plane model in the same way as was mentioned above. In this case, the algorithm always provides plane information for the input. All threshold parameters are empirically determined.

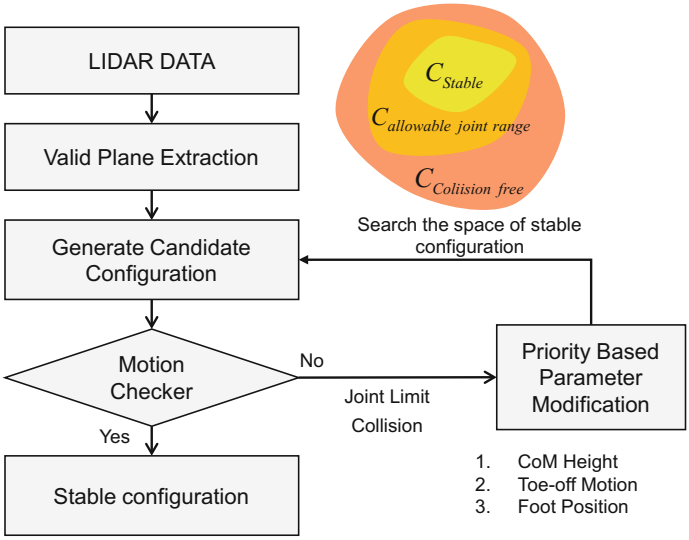


Fig. 27 Overall structure of the walking motion planner

In the terrain task, contrary to the case of walking on even ground, the robot has to consider height variations of the terrain, long strides, collisions, and joint limits. As can be seen in Fig. 27, a walking motion planner is developed for the terrain and stair tasks so that the robot can traverse the terrain by considering the constraints. First, the motion planner extracts valid planes from the LIDAR data and generates candidate footprints on the planes. Additionally, the motion planner generates two candidate waypoints of the CoM height of each support phase. Two waypoints ($CoM^1_{i,z}$, $CoM^2_{i,z}$) of the ith support phase are sampled according to the following rules.

$$CoM^1_{i,z} = \alpha \left(P_{i,Lz} + P_{i,Lz} \right) \tag{19}$$

$$CoM^2_{i,z} = \begin{cases} \beta P_{i,Rz} (if\ P_{i,Rz} < P_{i,Lz}) \\ \beta P_{i,Lz}\ (else) \end{cases} \tag{20}$$

where $P_{i,Lz}$ and $P_{i,Rz}$ are the left and right heights of the footprints of the ith support, respectively. α and β are heuristic values learned from many experiments; these values change depending on the height and the orientation of a block. When the configuration space is searched to satisfy a constraint like joint limit, a sampling based motion planning algorithm is used to find a stable configuration (Hauser et al. 2008). The motion planner provides a candidate configuration using the above rule, which is based on the experimental results, which are a list of approximately stable configurations. Through the above footprints and waypoints of the CoM height, the candidate

configurations are sampled. This sampling method reduces the computation time necessary for motion planning.

Second, the motion planner determines the overall motion by connecting the configurations between the initial configuration and the goal configuration with local paths in the motion checker. When connecting the adjacent configuration, the motion checker checks for collisions and joint ranges of the motion. If a collision occurs or if a joint exceeds its allowable joint range in a certain configuration, that configuration is modified by priority based on parameter modification. The parameters are the height of the CoM, the toe-off motion, and the footprint; the priority of the parameters is in the order that they are written in here. Additionally, the motion planner changes the pelvis orientation to increase the workspace of the legs. The pelvis is rotated in proportion to the velocity of the moving foot. These tasks of priority based parameter modification and pelvis rotation expand the stable configuration space.

4.3 Valve Task

For the valve pose estimation, the most important task is to determine inliers among the point clouds. First, using a modified flood fill algorithm, the segment closest to the sensor is extracted. The shortest range among all scanned data is found and set as the seed point of the expansion. Four-directional neighbors with range differences smaller than 3% are filled. If the segment is too small (50 range data in our implementation), it is discarded and the next closest segment is extracted.

Our algorithm of valve pose estimation is based on the RANSAC algorithm (Fischler and Bolles 1981). First, we determine inliers of the handle part of a valve, which is usually shaped like a thin donut. Three points are sampled to generate a circle in 3D space. Its normal n, center c and radius r are computed using the points p_1, p_2, and p_3:

$$n = \frac{(p_2 - p_1) \times (p_3 - p_1)}{\|(p_2 - p_1) \times (p_3 - p_1)\|} \tag{21}$$

$$c = [n,\, p_2 - p_1,\, p_3 - p_1]^{-T} \begin{bmatrix} n^T p_1 \\ (p_2 - p_1)^T (p_1 + p_2)/2 \\ (p_3 - p_1)^T (p_1 + p_3)/2 \end{bmatrix} \tag{22}$$

$$r = \|c - p_1\| \tag{23}$$

It should be noted that there is no assumption that the radius r is known. The cost function f_h for determining inliers is the Euclidean distance from each point p to the circle:

$$f_h = \left(n^T (p - c)\right)^2 + \left(\sqrt{\|p - c\|^2 - \left(n^T (p - c)\right)^2} - r\right)^2 \tag{24}$$

Only the points with cost values smaller than a user-defined threshold (20 mm in our implementation) are determined to be inliers of the handle part.

Second, we determine the inliers of the spoke part of a valve, which consists of N unknown bars with fixed angular intervals of $360/N°$ (e.g., 90° for $N = 4$, 72° for $N = 5$). To handle the points in 2D space, they are projected onto a plane perpendicular to n. Without loss of generality, the center of the handle can be projected onto (0, 0) of the new two-dimensional coordinate system. Spoke vectors v_k^N are predefined, starting from $[1, 0]^T$:

$$v_k^N = \begin{bmatrix} \cos\left(\frac{2\pi k}{N}\right) \\ \sin\left(\frac{2\pi k}{N}\right) \end{bmatrix} (0 \le k \le N) \tag{25}$$

A projected point $p' = \left[p_x, p_y\right]^T$ is determined as an inlier of v_k^N if the cost f_s—Euclidean distance between point and spoke—is smaller than a user-defined threshold (20 mm in our implementation):

$$f_s = \left(M_R p'\right)^T R_{q'} v_k^N \tag{26}$$

where M_R is adopted to generate a reversed vector with different signs:

$$M_R = \begin{bmatrix} 0 & -1 \\ 1 & 0 \end{bmatrix} \tag{27}$$

$$M_R p' = \begin{bmatrix} 0 & -1 \\ 1 & 0 \end{bmatrix} \begin{bmatrix} p_x \\ p_y \end{bmatrix} = \begin{bmatrix} -p_y \\ p_x \end{bmatrix} \tag{28}$$

It should be noted that the spoke vectors are multiplied by a rotation matrix $R_{q'}$ because the spoke part of the projected point cloud may be rotated. An arbitrary point $q' = \left[q_x, q_y\right]^T$ is selected to rotate spoke vectors.

$$R_{q'} \equiv \frac{1}{\|q'\|} \begin{bmatrix} q_x & -q_y \\ q_y & q_x \end{bmatrix} \tag{29}$$

Finally, using the squared sum of the distances from the inliers to their corresponding parts (handle or k-th spoke) as a cost function, all of the inliers are used to refine the valve-to-sensor transformation $[Rt]$:

$$\begin{aligned} f(R, t, c, r) = &\sum_{p \in H} \left(\left(\left\| M_{xy}(Rp + t) \right\| - r \right)^2 + \left(v_z^T \left(Rp + t \right) \right)^2 \right) \\ &+ \sum\nolimits_{k=0}^{N-1} \sum\nolimits_{p \in S_k} \left(\left(M_R M_{xy}(Rp + t) \right)^T v_k^N \right)^2 \end{aligned} \tag{30}$$

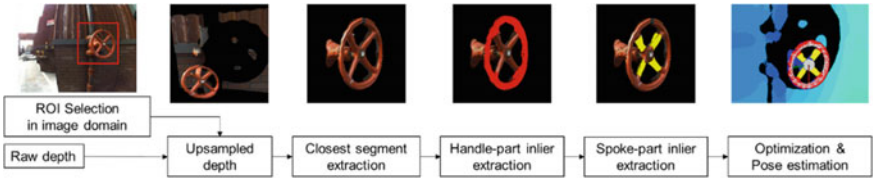


Fig. 28 Overview of the valve pose estimation

where H and S_k indicate sets of inliers that belong to the handle part and the k-th spoke part, respectively. M_{xy} and v_z are adopted to extract the x-y terms and the z term of the 3×1 vectors:

$$M_{xy} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix}, v_z = \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} \tag{31}$$

The whole process is shown in Fig. 28.

To deal with different heights of the valve and different floor conditions, the valve task is prepared for in standing mode, sitting mode, and height adjusted sitting mode (see Fig. 29). The approach position is fixed to the best pre-calculated position for each hand and mode. After the robot approaches the proper position, the valve task motion can be separated into two stages: reaching of the hand and rotating the valve. When the robot hand approaches the valve, the FT sensor on wrist is used to detect the collision between the valve and the robot hand and to check if the fingers are properly applied to the valve. To reduce unintended force between the valve and the robot during the rotation stage, a stretched finger motion and compliance of the under-actuated fingers were used.

Three different valve rotating strategies were designed for different sizes and numbers of spokes of valves. The first strategy is the center strategy, which uses the center of the valve to rotate the valve. The second strategy is the one finger strategy, which involves rotating the valve by putting one finger into the hole that is made by two spokes and a ring. The last strategy is the spoke strategy, in which the robot uses a single spoke to rotate the valve. Because it has a slip ring on its wrist, DRC-HUBO+ can rotate a valve without repositioning of its hand.

4.4 Wall Task

In the wall task, to cut a hole in the wall, the first step is to grab an appropriate tool such as a one or two-handed drill. In our case, the one-handed drill was selected because it provides a wider workspace for the DRC-HUBO+ than does the

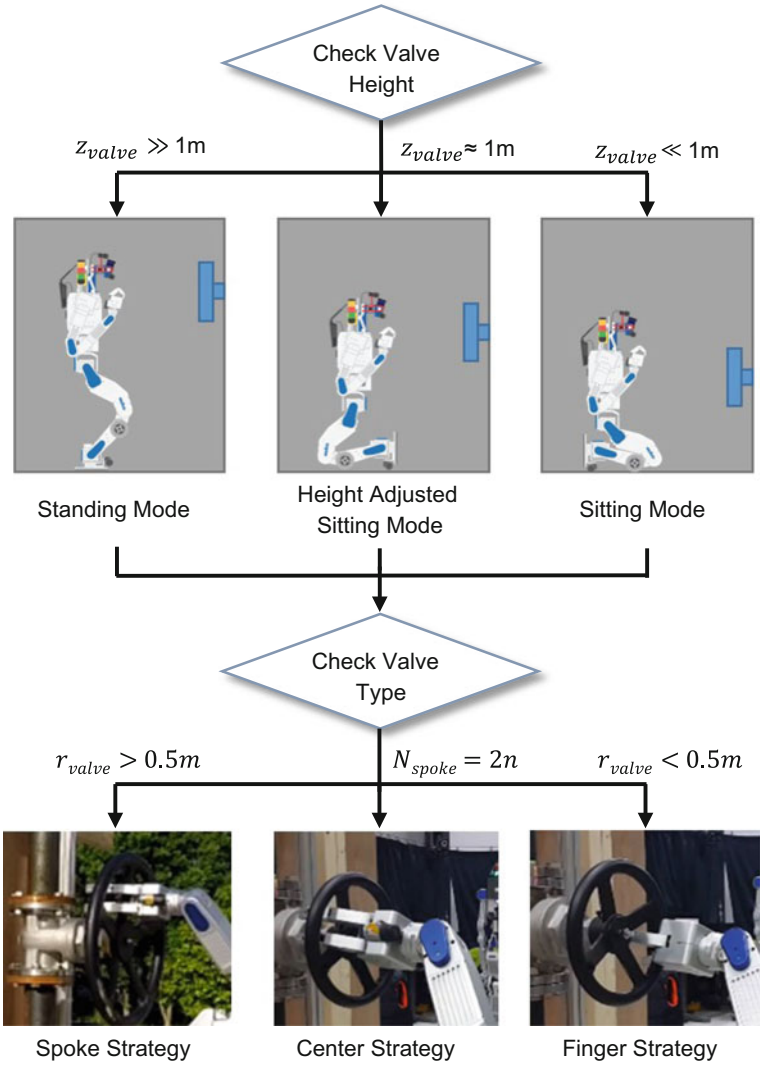


Fig. 29 Process flow of the valve task. When the valve is observed, the lower body pose is selected based on the height of the valve. Then, the hand mode is selected based on the features of the valve

two-handed drill. To grab and turn on the one-handed drill, it is necessary to know the pose (reference point and orientation) of the drill (see Fig. 30).

The shape of the one-handed drill was simplified into a 3D box and a cylinder, as can be seen in Fig. 30b. The reference point, which the robot has to grab, is determined by the center of mass of the cylinder, X_c. Two vectors v_u and v_h are also decided on to determine the grabbing pose. The vector v_u and v_h represent the axis of the cylinder and the major axis of the 3D box, respectively. The vector v_h is defined

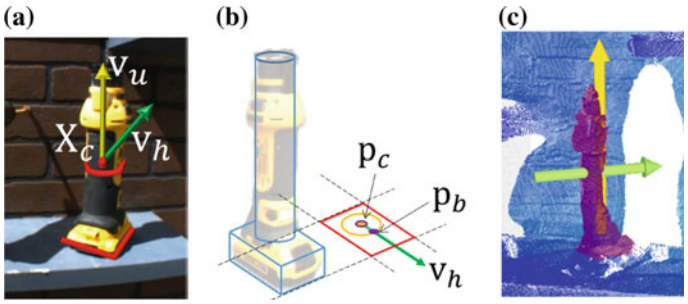


Fig. 30 Concept of drill pose estimation. **a** Illustration of drill pose in 2D image domain, consisting of center of mass X_c and two orientation vectors v_h and v_u. **b** Approximation of drill shape as a 3D cylinder and box. The two points p_c and p_b are the projected centers of the 3D cylinder and the box, respectively. **c** Estimated 3D drill pose. The yellow and green arrows indicate v_u and v_h, respectively. Cross point of the two arrows is the reference point X_c

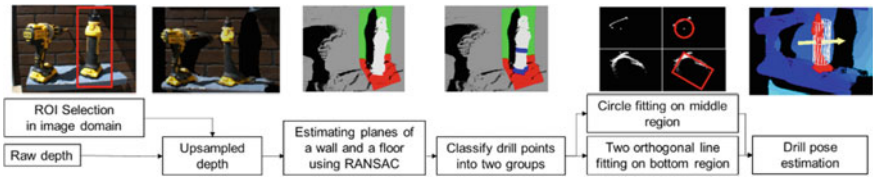


Fig. 31 Overview of the drill pose estimation

by two points p_c and p_b, i.e., $v_h = p_b - p_c$, where p_c and p_b are the projected centers of the respective 3D cylinder and the box on the floor plane, respectively. Thus, the 3D drill pose can be estimated by calculating the reference point X_c and the two orientation vectors v_u and v_h.

Figure 31 provides an overview of the proposed drill pose estimation approach. For a given upsampled depth within the ROI, as pre-processing, the planes of the wall and the floor are first estimated using RANSAC; then, the points on the two planes are considered to be outliers. The two sets of red and green points (see Fig. 31) represent the points on the floor and the wall planes, respectively. Especially, because the drill is located on the floor plane, this plane plays an important role in the process of drill pose detection, providing a projection domain to estimate X_c and v_h as well as a normal vector that corresponds to v_u.

After removing outlier points of the two planes, classification is performed for the remaining points, which correspond to the drill points, into two point sets according to two height thresholds from the floor plane. The middle points set X_m and the bottom points set X_b are determined as can be seen in the fourth column of Fig. 31; the parameters for the drill pose are approximated using X_m and the X_b.

To estimate X_c, the middle points set X_m is projected onto the floor plane. On the floor plane domain, a circle point, p_c, is estimated by RANSAC based circle fitting. Because the radius of the cylinder body part of the drill is already known, it is possible

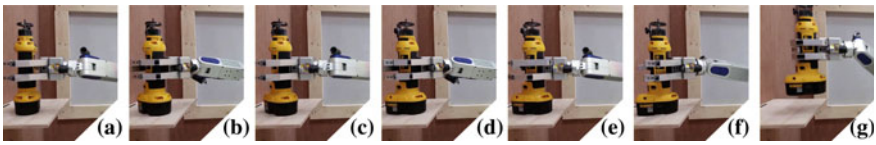


Fig. 32 Process of rotating the drill to a proper pose to lift it up. **a**, **c**, and **e** DRC-HUBO+ grasps drill. **b**, **d**, and **f** Robot rotates the drill a certain amount and releases it. **g** Finally, it lifts the drill

to easily estimate the center point p_c, and calculate X_c by compensating for the middle height threshold of v_u. To estimate v_h, it is necessary first to estimate the projected centers p_c and p_b. The circle center $\mathrm{p_c}$ was already calculated in the above step. To estimate the center of box p_b, the bottom points set X_b is projected onto the floor plane. In this case, because the 3D box cannot be fully observed using actual depth measures but can be only partially observed, two orthogonal lines are estimated using RANSAC instead of using rectangle fitting. From the estimated orthogonal lines, the rectangle and its center p_b can be estimated. Finally, v_h is estimated according to the above definition. Figure 30c shows the drill pose as estimated using the proposed approach.

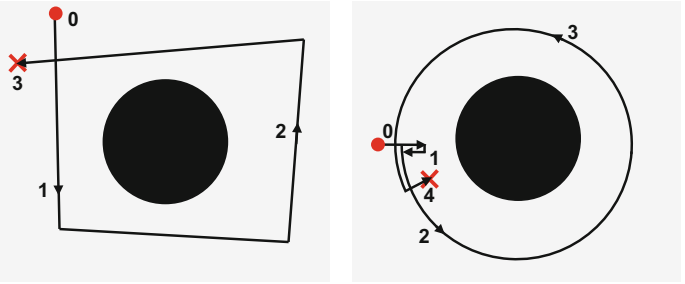
To turn on the one handed drill, the robot has to push the button with its other hand; this must be done hard and precisely enough. To do this, the orientation of the drill after the grip motion must be in certain range, so that it can be reached by the other hand.

Rotating the drill vertically after grabbing it requires two-handed motion; this task involves some uncertainty and risk, so the drill should be rotated before the robot grabs it and lifts it up. Using IK internal simulation, the minimum re-grabbing motion is generated, as can be seen in Fig. 32. To avoid a singularity and to widen the workspace, the waist joint and the redundancy of one arm are used.

After grabbing the drill, the robot uses a 2D image to check the vertical position and angle of the grabbed drill. The drill switch position/orientation is calculated using the pre-known model of the drill. Several trial movements are applied to turn on the drill, and a microphone is used to detect the drill's working status.

In the wall cutting motion, IK internal simulation is also used to determine the feasibility of the motion and the best position in which to stand and drill. At this time, the orientation of the drill can be fixed because the wall is flat; as such, a pre-calculated position reachability lookup table for the workspace is used to decrease the calculation time. To maintain contact with the wall while cutting, a force-tracking PD control is used. The angle of gravity and the force torque sensor change during any cutting motion, so gravity compensation for the hand and the drill is used to compensate for the force of gravity as it applies differently by position.

Circle-cutting mode and the polygon-cutting mode are both prepared to deal with these different situations. In the circle-cutting mode, the operator sets the center position and the radius in the 2D image and uses simple 2D-3D matching to find the center point for cutting. The RANSAC algorithm is used to find the normal of the wall. The cutting trajectory can be modified to make sure that the circle being



(a) Polygon-Cutting Mode (b) Circle-Cutting Mode

Fig. 33 Path examples of **a** polygon-cutting mode and **b** circle-cutting mode

Fig. 34 Concept of predicted driving trajectory projection for manual driving task. Leftmost image depicts the mapping between the ground plane and the camera coordinates to project the trajectory. The other images are examples of predictive driving trajectory projection in the DRC Finals

cut is closed. In each direction normal to the circle, a 2 cm cut to the center and in the direction opposite to the center are performed to make sure that even if there is slipping or some other error in the cutting motion (Fig. 33b 0 and 1), the trajectory will be a closed one. To remove the piece that has been drilled from the wall, the drill was moved a certain distance toward the center (Fig. 33b 4).

In polygon mode, to tell the robot where to cut, the operator clicks the proper point in the 2D image and uses simple 2D-3D matching to find the points to cut. The trajectory for each end is stretched 4 cm to make sure the cutting trajectory is closed.

4.5 Drive Task

There were two alternative solutions to the drive task. The first one is a manual method in which the operator looks at streaming images and provides continuous steering and pedaling commands to the robot. Because there is no blackout in communication at Link 2, it is possible to constantly obtain high quality streaming images. As can be seen in Fig. 34, the future path of the wheels is visualized in the 2D streaming images. The path is determined based on the steering angle; for precise visualization, the streaming image camera was calibrated before driving to display the predicted driving trajectory in the 2D image.

For the virtual trajectory projection, a transform T was estimated between the world and image coordinates, as can be seen in Fig. 34. In the case of a plane, such as a planar road, the transform can be simplified into a 2D transform, so-called homography matrix H (Hartley and Zisserman 2003).

Given a set of point correspondences $\{\mathrm{x_i}, \mathrm{X_i}\}_{i=1}^{\mathrm{N}}$ between the image and the 3D world coordinates, a homography matrix H can indicate the 2D transform that satisfies

$$x_i \cong H X_i \tag{32}$$

where $\cong$ denotes equality up to scale, $x_i = [x_i\, y_i\, 1]^T$ in homogeneous image coordinates, and $X_i = [X_i\, Y_i\, 1]^T$ is 3D correspondence points on the XY plane. By rewriting Eq. 32 as $x_i \times H X_i = 0$, it can be formulated using a linear equation, called a Direct Linear Transform (DLT), as

$$Ah = 0 \tag{33}$$

where

$$A = \begin{bmatrix} X_i & Y_i & 1 & 0 & 0 & 0 & -x_i X_i & -x_i Y_i & -x_i \\ 0 & 0 & 0 & X_i & Y_i & 1 & -y_i X_i & -y_i Y_i & -y_i \end{bmatrix} \tag{34}$$

The value of h is a vector representation of H. Due to noise, the equalities in Eq. 33 may not be satisfied in practice. As such, the global homography, H, can be estimated in a least squares manner, i.e., DLT, by

$$h = arg\,min_h \, \|Ah\|_2^2 \; subject\; to \; \|h\|_2 = 1 \tag{35}$$

The above optimization minimizes the algebraic error, and the global homography H can be obtained in a closed-form by finding the singular vector of A with the smallest singular value. It should be noted that because the degree of freedom of H is eight up to scale, at least four correspondences are required. Therefore, a rectangular plane object of known size is used and the four corners are manually selected as point correspondences. Using the estimated homography, H, the virtual trajectory of the wheels is projected into the image domain, as can be seen in Fig. 34.

The second solution for the drive task is the fully autonomous method. The RANSAC method (Fischler and Bolles 1981) was used to detect obstacles; the Wagon model was used to control the steering and the velocity of the vehicle using only the limited number of sensors that were installed on the DRC-HUBO+ (Jeong et al. 2015).

There are pros and cons in both methods. The manual method can allow the robot to drive fast, but this method depends on human operation. The autonomous method is accurate and robust, but the robot must drive slowly because it is hard to exactly

Table 2 Egress times of the seven teams who succeeded in the egress task

Team	KAIST	MIT	Tartan	IHMC	Robosimian	WPI-CMU	Trooper
Time (s)	125	210	130	212	350	359	485

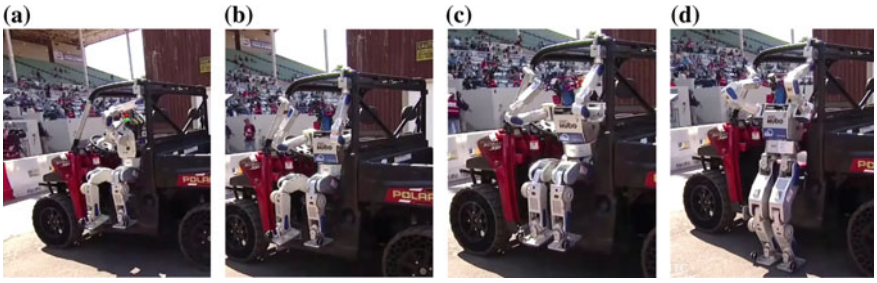


Fig. 35 When driving is finished, **a** DRC-HUBO+ changes its pose to prepare for egress. After this, **b** it grabs with both hands, and hybrid position/force control is started. **c**, **d** To reduce impact when DRC-HUBO+ lands on the ground, each arm supports about 10 kg

estimate the current velocity. Among the two options, Team KAIST used the manual method in DRC because obstacles were sparse and most of the path was straight.

4.6 Egress Task

It took DRC-HUBO+ about 2 min to egress from the vehicle in the DRC Finals. Team KAIST finished the egress task in the shortest time among the teams in the DRC Finals, as Table 2 shows.

After finishing the driving task (see Sect. 4.5), DRC-HUBO+ releases its right hand from the steering wheel and grabs the tilted roll cage. Then, hybrid position/force control is started (see Sect. 6.2). The X, Y positions (horizontal) and the desired Z (upward) direction force are controlled as predefined values that were determined experimentally. To reduce impact during landing, both hands pull on the roll cage with a force of about 100 N. If the robot detects landing, it stabilizes its ZMP by controlling its pelvis position. Before releasing its hands, DRC-HUBO+ uses feedback control of the FT sensor on its wrist to remove the force exerted on both hands. By applying this controller, it is guaranteed that the robot will not fall down after releasing both hands (Fig. 35).

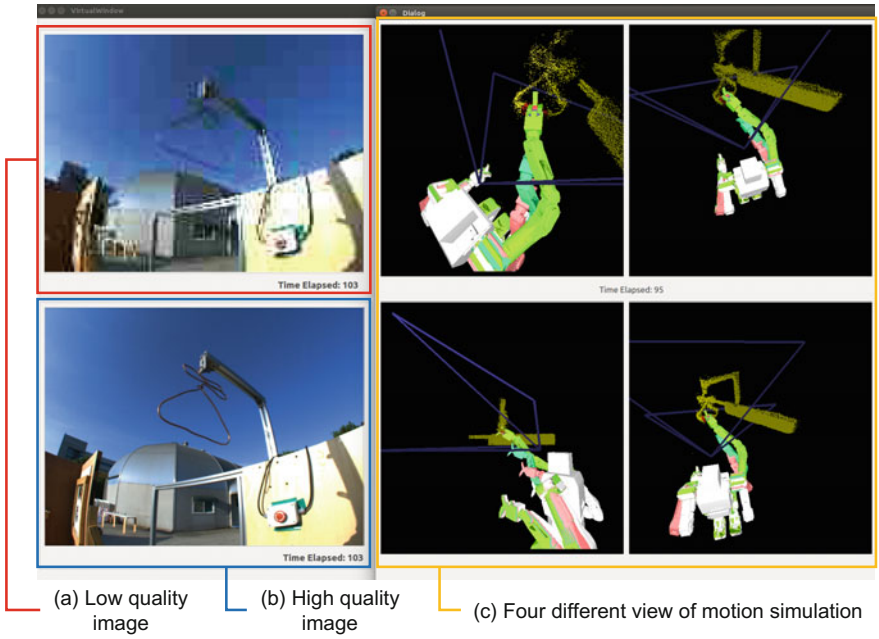


Fig. 36 In the surprise task, there were two kinds of visual feedback: 2D images and 3D point cloud data. With a joystick, user generates via-point command in 3D space. Operator controls a ghost in 3D space and selects current posture as a via-point. By connecting via-points, it is possible to generate sequential motion. With this tele-command interface, the robot can perform any kind of manipulation task

4.7 Surprise Task

In a situation of low band communication, the manipulator cannot be manually controlled in real-time because it is impossible to obtain continuous visual feedback. Hence, a via-point strategy was devised to create a set of via-points for the manipulator in the 3D space and send this information to the robot. As can be seen in Fig. 36c, via-points are created by joystick control based on the 3D cloud data. The joystick can be used to control the 7 DOF arm, the gripper movement, and the waist. At this time, the arm angle, which is a redundant task, is automatically determined based on the method described in Sect. 3.1.

5 Lessons

In this section, we will discuss what we learned from and feel about the DRC Finals. First of all, we thank DARPA for giving us the opportunity to participate in this

robotics challenge. It was possible to meet and learn from other teams and share our experiences and thoughts. There were many good hardware platforms, vision systems, operating methods, control algorithms, etc. All of the systems we encountered were the most progressive outputs in the present field of robotics, and it was a great experience to see them.

We thought that this competition showed the double-sidedness of the possibility of counteracting disaster scenarios using present robot technology. In the challenge, the given scenario was not an impossible mission for the present technology. However, the tasks designed in the competition are different from what robots will confront in real situations. In real disaster conditions, all of the circumstances will be worse. The ground condition, workable space, light condition, and communication will not be as good as those that the DRC Finals provided. In this competition, many robotic platforms fell down and never got back up by themselves; some of them broke. Participants were permitted to reset their robots with a certain time penalty, but in reality this is the same as a mission fail. Thus, we need to research stable mobility on uneven ground or rubble areas to reach the task spot.

In the aspect of vision system, it should be considered that the entire vision system can fail due to certain deficiencies of the sensors. For example, depth sensors, such as the LIDAR and the stereo camera, can completely fail due to clouds of vaporous air, severely low light conditions, or radioactivity. In these situations, every vision system that was introduced in the DRC Trials and Finals would fail to work properly. Thus, we should research robustness issues for depth measurement and estimation, and vision algorithms that can be used in severe conditions to counteract real disaster situation. Furthermore, communication, autonomy, and system diagnosis and recovery are important issues for disaster robots, and there is still a need for more research if we are to use such systems in real sites.

6 Conclusions

The performance of DRC-HUBO+ was validated at the DRC finals. The robot successfully carried out all of its tasks in real environments without any state initialization; we won first place with a full score. In this paper, we have presented a survey of the humanoid robot platform, DRC-HUBO+, including the overall hardware configuration, software architecture PODO, various control methods for operating the robot, and the vision system. We have also provided details on our overall strategy and on the task oriented vision algorithms that were used to solve the given tasks. In addition, we have discussed what we learned and our views on the limitations of current robot systems.

References

AlwaysUp| Run Any Application as a Window Service. Retrieved August 14, 2015, from http://www.coretechnologies.com/products/AlwaysUp/.

Bae, H., Lee, I., Jung, T., & Oh, J. H. (2016, October). Walking-wheeling dual mode strategy for humanoid robot, DRC-HUBO+. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), 2016* (pp. 1342–1348). IEEE.

Bouguet, J.-Y. (2004). Camera calibration toolbox for Matlab.

Boykov, Y., & Veksler. O. (2006) Graph curs in vision and graphics: Theories and applications. In *Handbook of mathematical models in computer vision*. (pp. 79–96) Springer.

Cheng, G., & Zelinsky, A. (2001). Supervised autonomy: A framework for human-robot systems development. *Autonomous Robots, 10*(3), 251–266.

Cho, B. K., Kim, J. H., & Oh, J. H. (2011). Online balance controllers for a hopping and running humanoid robot. *Advanced Robotics, 25*(9–10), 1209–1225.

Dantam, N., & Stilman, M. (2012 November). Robust and efficient communication for real-time multi-process robot software. In *2012 12th IEEE-RAS International Conference, on Humanoid Robots (Humanoids)* (pp. 316–322). IEEE.

DARPA Robotic Challenge Finals 2015. Retrieved August 10, 2015, from http://www.theroboticschallenge.org/.

DRC Finals Rule Book. Retrieved from August, 2015 http://www.theroboticschallenge.org/sites/default/files/docs/2015_04_09_DRC_Finals_Rule_Book_DISTAR_24388.pdf.

Fischler, M. A., & Bolles, R. C. (1981). Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography. *Communications of the ACM, 24*(6), 381–395.

Girshick, R., Donahue, J., Darrell, T., & Malik, J. (2014). Rich feature hierarchies for accurate object detection and semantic segmentation. In *IEEE Conference on Computer Vision and Pattern Recognition*.

Hartley, R., & Zisserman, A. (2003). *Multiple view geometry in computer vision*. Cambridge University Press.

Hauser, K., Bretl, T., Latombe, J. C., Harada, K., & Wilcox, B. (2008). Motion planning for legged robots on varied terrain. *The International Journal of Robotics Research, 27*(11–12), 1325–1349.

Hintjens, P. (2013). *ZeroMQ: Messaging for many applications*. O'Reilly Media, Inc.

Jeong, H., Oh, J., Kim, M., Joo, K., Kweon, I. S., & Oh, J. H. (2015, November). Control strategies for a humanoid robot to drive and then egress a utility vehicle for remote approach. In *2015 IEEE-RAS 15th International Conference on Humanoid Robots (Humanoids)* (pp. 811–816). IEEE.

Kajita, S., Kanehiro, F., Kaneko, K., Fujiwara, K., Harada, K., Yokoi, K. et al. (2003). Biped walking pattern generation by using preview control of zero-moment point. In *Proceedings of the IEEE International Conference on Robotics and Automation* (Vol. 2, pp. 1620–1626), Taipei, China.

Kajita, S., Kanehiro, F., Kaneko, K., Yokoi, K., & Hirukawa, H. (2001). The 3D linear inverted pendulum model: A simple modeling for a biped walking pattern generation. In *Proceedings of the 2001 IEEE/RSJ International Conference on Intelligent Robotic Systems*, (Vol. 1, pp. 239–246).

Kannala, J., & Brandt, S. S. (2006). A generic camera model and calibration method for conventional, wide-angle, and fish-eye lenses. *IEEE Transactions on Pattern Analysis and Machine Intelligence, 28*(8), 1335–1340.

Kim, I., & Oh, J. H. (2013). Inversekinematic control of humanoids under joint constraints. *International Journal of Advanced Robotic Systems*, *10*(74).

Kim, J. H., Park, S. W., Park, I. W., & Oh, J. H. (2002). Development of a humanoid biped walking robot platform KHR-1-initial design and its performance evaluation. In *Proceedings of 3rd IARP International Work on Humanoid and Human Friendly Robotics* (pp. 14–21).

Kim, M. S., & Oh, J. H. (2010). Posture control of a humanoid robot with a compliant ankle joint. *International Journal of Humanoid Robotics, 7*(01), 5–29.

Lee, I., & Oh, J. H. (2015). Humanoid posture selection for reaching motion and a cooperative balancing controller. *Journal of Intelligent & Robotic Systems*, 1–16.

Lee, I., Oh, J., Kim, I., & Oh, J. H. (2017). Camera-laser fusion sensor system and environmental recognition for humanoids in disaster scenarios. *Journal of Mechanical Science and Technology, 31*(6), 2997–3003.

Levenberg, K. (1944). A method for the solution of certain problems nonlinear in least square. *Quarterly of Applied Mathematics, 2,* 164–168.

Lim, J., & Oh, J. H. (2015). Backward ladder climbing locomotion of humanoid robot with gain overriding method on position control. *Journal of Field Robotics*.

Lim, J., Shim, I., Sim, O., Joe, H., Kim, I., Lee, J. et al. (2015, November). Robotic software system for the disaster circumstances: System of team KAIST in the DARPA Robotics Challenge Finals. In *2015 IEEE-RAS 15th International Conference on Humanoid Robots (Humanoids)* (pp. 1161–1166). IEEE.

Marquardt, D. W. (1963). An algorithm for least-squares estimation of nonlinear parameters. *Journal of the Society for Industrial and Applied Mathematics, 11*(2), 431–441.

Nakamura, Y., & Hanafusa, H. (1986). Inverse kinematic solutions with singularity robustness for robot manipulator control. *Journal of Dynamic Systems, Measurement, and Control, 108*(3), 163–171.

Oh, J., & Oh, J. H. (2015). A modified perturbation/correlation method for force-guided assembly. *Journal of Mechanical Science and Technology, 29*(12), 5437–5446.

Park, I. W., Kim, J. Y., Lee, J., & Oh, J. H. (2005, December). Mechanical design of humanoid robot platform KHR-3 (KAIST humanoid robot 3: HUBO). In *2005 5th IEEE-RAS International Conference on Humanoid Robots* (pp. 321–326). IEEE.

Scaramuzza, D., Martinelli, A., & Siegwart, R. (2006). A toolbox for easily calibrating omnidirectional cameras. In *IEEE/RSJ International Conference on Intelligent Robots and Systems* (pp. 5695–5701).

Shim. I., Shin. S., Bok. Y., Joo. K., Choi. D.-G., Lee. J.-Y. et al. (2015). Vision system and depth processing for DRC-HUBO+. arXiv:1509.06114.

Shimizu, M., Kakuya, H., Yoon, W. K., Kitagaki, K., & Kosuge, K. (2008). Analytical inverse kinematic computation for 7-DOF redundant manipulators with joint limits and its application to redundancy resolution. *IEEE Transactions on Robotics 24*(5), 1131–1142.

Stasse, O., Saïdi, F., Yokoi, K., Verrelst, B., Vanderborght, B., Davison, A. et al. (2008). Integrating walking and vision to increase humanoid autonomy. *International Journal of Humanoid Robotics*, *5*(02), 287–310.

Supervirordl A Process Control System. Retrieved August 12, 2015, from http://supervisord.org/.

Tsai, L. W. (1999). *Robot analysis: The mechanics of serial and parallel manipulators*. Wiley.

Wampler, C. W. (1986). Manipulator inverse kinematic solutions based on vector formulations and damped least-squares methods. *IEEE Transactions on Systems, Man and Cybernetics*, *16*(1), 93–101.

Xenomail Real-time framework for Linux. Retrieved June 11, 2015, from http://xenomai.org/.

Yoo, D., Park, S., Lee, J.-Y., Paek, A., & Kweon, I.S. (2015). AttentionNet: Aggregating weak directions for accurate object detection. CoRR, abs/1506.07704.

Zhang, Z. (2000). A flexible new technique for camera calibration. *IEEE Transactions on Pattern Analysis and Machine Intelligence, 22*(11), 1330–1334.