2015 IEEE-RAS 15th International Conference on Humanoid Robots (Humanoids)
November 3 - 5, 2015, Seoul, Korea

Multi-layered Real-time Controllers for Humanoid's Manipulation and Locomotion Tasks with Emergency Stop

Shunichi Nozawa[1], Eisoku Kuroiwa[1], Kunio Kojima[1], Ryohei Ueda[1], Masaki Murooka[1], Shintaro Noda[1], Iori Kumagai[1], Yu Ohara[1], Yohei Kakiuchi[1], Kei Okada[1], Masayuki Inaba[1]

***Abstract*— This paper describes a practical method to construct real-time controllers to achieve locomotion and manipulation tasks with a humanoid robot. We propose a method to insert emergency stop functionality to each layer to avoid robot's falling down and joint overloads even if recognition and planning error exist. We explain implementation of multi-layered real-time controllers on HRP2 robot and application to several manipulation and locomotion tasks. Finally, we evaluate emergency stop functionality in several manipulation tasks.**

I. Introduction

Recently humanoid robots are applied to tasks in which both locomotion and manipulation required. In the case that required tasks are various, humanoid real-time controller needs to receive various commands from human operators and motion planners.

In addition, humanoid robots are possible to fall down when some kinds of failures occurs such as inadequate motion planning and inacurrate environment recognition. In related researches, fault trerancy during walking and standing [1], [2], [3] and safe falling and recovery from falling down[4], [5] are discussed. Considering these works, we need to extend controllers to be robust against failures in manipulation.

In this paper, we introduce a method to construct real-time controllers applicable to both locomotion and manipulation and including emergency motion stopping functionality. First, we explain structure of our multi-layered real-time controllers consists of several modules for feed-forward motion generator, sensor feed-back controller, and fail-safe modules for locomotion and manipulation tasks such as Darpa Robotics Challenge (DRC [1]). Next, we propose a method to extend these controllers capable of stopping robot's motion and apply to failure of manipulation tasks. Finally, we apply proposed controllers to HRP-2's DRC task execution and experimentally validate proposed emergency stopping functionality is useful to failures because of inadequate motion planning and recognition errors.

II. Multi-layered Real-time Controller System with Emergency Stop

A. System Overview

We explain whole configuration of our proposed controllers including multi-layered emergency stopping for joint-position-controlled humanoid robots. Fig.2 shows system configuration. In our system, Operator commands manipulator and locomotion goal to Planner and triggers recognition during manipulation and locomotion tasks (details are explained in [6]). Planner plans robot's motion based on Operator commands and Object/Environment recognition and sends motion commands to Real-time Controller. Note that we can configure autonomy level by changing amount of Operator commands. In the following sections, we represent both Operator and Planner just by Planner.

Real-time Controller has Planned Manipulation Command Receiver, Walking-feedforward Balancing Controller, and Sensor-feedback Controllers. For manipulation, Planner sends manipulation commmads such as joint angles, reference force at the hands and feet to Planned Manipulation Command Receiver. Planned Manipulation Command Receiver interpolates sparse commands. For locomotion, Planner sends locomotion commands such as footsteps and goal standing position to Walking-feedforward Balancing Controller. Walking-feedforward Balancing Controller calculates

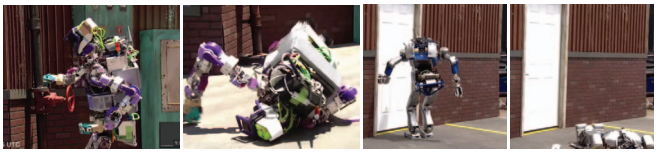


Fig. 1. Falling-down during Various Manipulation Failure

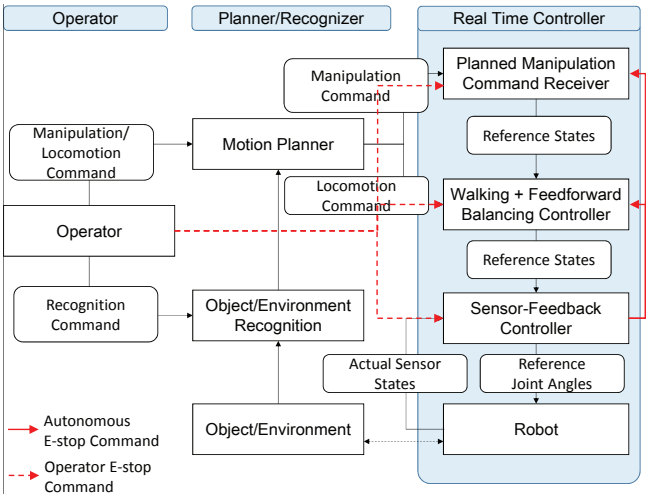


Fig. 2. Manipulation and Locomotion System Capable of Emergency Stop Rectangles are system modules and rounded rectangles are data flow. Real time controller supports multi-layered emergency stop and fail recovery. Red arrows are autonomous and operator emergency signals.

[1] S. Nozawa, K. Kojima, R. Ueda, M. Murooka, Y. Ohara, E. Kuroiwa, S. Noda, I. Kumagai, Y. Kakiuchi, K. Okada, and M. Inaba are with Graduate School of Information Science and Technology, University of Tokyo 7-3-1 Hongo, Bunkyo-ku, Tokyo 113-8656, Japan nozawa at jsk.t.u-tokyo.ac.jp

[1]http://www.theroboticschallenge.org/

978-1-4799-6885-5/15/$31.00 ©2015 IEEE

full-body motion by solving full-body inverse kinematics. Sensor-feedback Controllers receives full-body motion from these controllers and modifies joint angles to keep balance and track desired operational force.

B. Emergency Stop for Each Layer

Our emergency stopping is implemented both as Operator commands (red dotted lines) and Autonomous commands (red solid lines). In the following sections, we mainly discuss Autonomous Emergency-stop commands especially for manipulation. Our emergency stopping is multi-layered, e.g., the robot can choose stop which motion should be stopped.

When inadequate motion planning and object/environment recognition failure occur, unexpected contacts among the robot, the target object, and the environment occur. They invoke robot's falling down and joint overloads. In this case, absorbing disturbances while standing or changing manipulation motion is preferable to stepping. In many case of manipulation failure, robot's upper body has contact with objects and environments and the robot suffers from unexpected reaction force during stepping and stepping motion can be unstable. On the other hands, primal cause of manipulation failure is manipulation motion itself and changing or stopping manipulation motion is expected to be useful without any changes of support polygon through stepping. In this paper, we implement avoidance from robot's falling by emergency stopping of manipulation. Note that stepping during manipulation can be important when disturbance is quite large. Many researches discussed that humanoid robots are required to change balancing stratey during disturbance absorption during standing and stepping such as [7]. In our case, disturbance absorption during standing consists of sensor-feedback joint angle modifications and changing or stopping of planned manipulation motion. In future work, we will implement stepping functionality because of manipulation failure.

Proposed controller has also emergency walking stopping for locomotion failure.

In Fig.2, Sensor-feedback Controller is always activated even if emergency stopping from locomotion and manipulation failure. We add emegency stopping of Sensor-feedback Controller just for safety during experiments. Note that if this controller stops, the robot cannot adequately keep balance and control reaction forces.

The main contribution of multi-layered emergency stopping is that the robot continues motion control which is not stopped. For example, when the robot autonomously stops manipulation motion, sensor-feedback controller continues working and the robot can keep balance and force control. Therefore even if the fatal manipulation failures occurred, Planner can invoke object/environmetns recognition again and check diagnostics why such manipulation failure occurred.

In the following section, we explain the controller architecture in detail. In Sec. III, we explain detailed formulation of multi-layered controller. Especially we discuss emergency stopping of planned manipulation motion. In Sec. IV, we explain our HRP-2 hardware and a method to apply our proposed method to DRC tasks. In addition, we evaluate proposed emergency stopping through the experiments similar to actual DRC failure of our team. In these experiments, we add disturbance for manipulation intentionally and validate emergency stopping functionality.

C. Related Works

For humanoid's emergency stopping, Morisawa et al. [1] achieved emergency stopping of walking motion considering four phases during walking and proposed emergency stopping based on state space feedback [8] and discussed a method to stop upper body motion. Kaneko et al. proposed a method to constant decreasing velocity of Center Of Gravity (COG) [2] and discussed a method to decide the timing of emergency stop. Takubo et al. [9] achieved walking stop using map between foot landing remaining time and foot modification. Pratt et al. defined Capture point [3] and discussed stopping of motion focusing on linear pendulumn model and fly-wheel model. For humanoid's falling checking, Sugihara et al. [10] discussed balancing during standing and the condition of stepping. Yu et al. [5] introduce humanoid's Fall Triger Boundary and achieved falling direction control. Compared with these researches, the contribution of our method is providing multi-layered emergency stop functionality and especially stopping of manipulation motion to avoid falling down from manipulation failure without stepping motion. In our controllers, some controller modules keep working as much as possible. For example, when manipulation motion is stopped, balancing controller works. In [1], emergency stopping was applied just to walking pattern generation and balancing controller kept working. Compared with this, we introduce these functionality for manipulation.

III. Real-time Controller Architecture

In this section, we describe real-time controller architecture and detailed explanations of controller modules.

Our real-time controller is implemented as RT-Component (RTC) on OpenRTM[11] architecture. On real-time controller threads, each controller module is executed sequentially [2] like plugin architecture[12]. Modules are categorized into (a) Receiver of planner (b) State estimator (c) Feed-forward trajectory generator (d) Feed-back controller (e) Fail-safe component and logger. Fig.3 shows detailed controller serial execution. Each module communicates with each other and passes command values, sensor values, and state values.

As (b) State estimator, KalmanFilter estimates robot's body attitude based on measurements from gyro sensor and acceleration sensor. RemoveForceOffset removes mass and center-of-mass component from measured force and moments.

As (e), CollisionDetector[13] avoids self-collision of input motion. When self-collision does not occur, this component outputs joint angles same as input joint angles. Otherwise, this component outputs joint angles same as previous input

[2]Our real-time controller is open-source and hosted on https://github.com/fkanehiro/hrpsys-base from AIST.

joint angles without collision. HardEmergencyStopper stops robot's motion when emergency signal is triggered. SoftErrorLimitter limits output joint angles when joint angle error, velocity limit over, and position limit over occurred.

RobotHardware is hardware interface for real-time controller sequence. This component writes final joint angle commands from SoftErrorLimitter, reads sensor values such as encoder values, force/torque sensor measurements, gyro/acceleration measurements, and provide sensor values to each module. Note that the process which open robot's device is different from RobotHardware and RobotHardware just communicates with it through shared memory.

We explain motion generation and feed-back controllers in the next subsections.

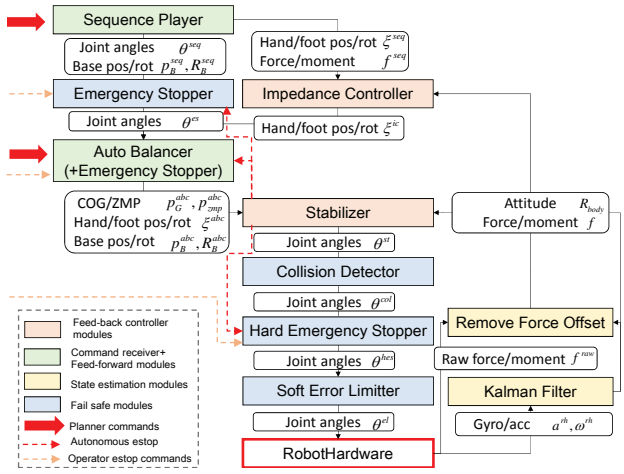


Fig. 3. Multi-layered Real-time Controllers
Controllers communicate with each other and modifies command data.

A. Sequence Player

SequencePlayer is the receiver of command values such as trajectories of reference joint angles, hands and feet wrenches, position and orientation of the base link sent from planners and operators. This component interpolates and stores trajectories. When difference of time stamp of commanded trajectories is lager rather than control loop, this component interpolates commanded trajectories so that difference of time stamp becomes control loop time. in every control loop. This component supports several interpolator methods such as HoffArbib [14] interpolation and linear interpolation. In HoffArbib interpolatoin, this component calculates jerk-level values from Eq. 1 and obtains position-level values by integrating them in every control loop. In Eq. 1, x_{cmd} is commanded position from planners and operators. a_{cur}, v_{cur}, and x^{seq}_{cur} are current acceleration, velocity, and position of interpolated variables. t_{ha} is remaining time to goal position. x^{seq}_{cur} is final outputs from this component (joint angles, wrenches, and position and orientation of the base link).

$$\dot{a} = -9a_{cur}/t_{ha} - 36v_{cur}/t^2_{ha} + 60(x_{cmd} - x^{seq}_{cur})/t^3_{ha} \quad (1)$$

In the following formulation, we use HoffArbib function:

$$x = \text{HoffArbib}(x_{from}, x_{to}, t_{from\to to}) \quad (2)$$

Here, let x be output. x is interpolated x_{from} and x_{to} in $t_{from\to to}$.

B. Impedance Controller

ImpedanceController performs wrench control at the end-effectors. This component modifies end-effector trajectories.

$$\boldsymbol{m}_{ic}\ddot{\delta\boldsymbol{\xi}}_{ee} + \boldsymbol{d}_{ic}\dot{\delta\boldsymbol{\xi}}_{ee} + \boldsymbol{k}_{ic}\delta\boldsymbol{\xi}_{ee} = \boldsymbol{f}_{ee} - \boldsymbol{f}^{seq}_{ee} \quad (3)$$

Let $\delta\boldsymbol{\xi}_{ee} \in R^6$ be difference of position and orientation of end-effectors and it holds $\delta\boldsymbol{\xi}_{ee} = \boldsymbol{\xi}^{ic}_{ee} - \boldsymbol{\xi}^{seq}_{ee} \in R^6$. Let $\boldsymbol{\xi}^{seq}_{ee}$ be reference effector position and orientation from SequencePlayer and and $\boldsymbol{\xi}^{ic}_{ee}$ be modified final output from ImpedanceController. Let $\boldsymbol{m}_{ic}, \boldsymbol{d}_{ic}, \boldsymbol{k}_{ic} \in R^{6\times 6}$ be inertia, damping, spring coefficient matrices. Note that this component considers $\boldsymbol{\xi}^{seq}_{ee}$ as the center of spring term, $\boldsymbol{\xi}^{ic}_{ee}$ completely tracks $\boldsymbol{\xi}^{seq}_{ee}$ without error in the case of no external forces applies.

C. Auto Balancer

AutoBalancer generates walking pattern and calculates joint angles satisfying COG and effector trajectories contraint. This component obtains reference joint angles and base link position and orientation from SequencePlayer, hands trajectories $\boldsymbol{\xi}^{ic}_{ee}$ from ImpedanceController. This component calculates feet trajectories based on footstep commands from planner.

1) Calculation of swing foot trajectory and ZMP trajectory considering toe-heel contact: AutoBalancer calculates reference ZMP trajectory based on goal standing position or footstep command from planner. This component interpolates footsteps into smooth feet trajectories. Here we use toe-heel contact during walking to obtain stable landing contact like [15] and relax foot kinematics reachability.

Here we introduce a method to calculate reference ZMP trajectory and swing feet trajectories based on user-defined transition time parameters. Transition time parameters includes (i) initial sole contact phase, (ii) zmp transition phase from sole to toe, (iii) zmp transition phase from toe to heel, (iv) zmp transition phase from heel to sole, (v) final sole contact phase. Note that these phase from (i) to (v) is for forward walking and we use the reversed phase for backward walking, e.g., from (v) to (i).

To calculate reference ZMP, we calculate reference COP of each foot and reference ZMP based on them. Reference COP of each foot $\boldsymbol{p}_{[r,l]cop}$ is calculated from linear interpolation from foot origin position to toe position in (i) phase, from toe position to heel position in (iii) phase, and from heel position to sole position in (v) phase. Reference ZMP is distributed from each foot COP.

$$\boldsymbol{p}^{abc}_{zmp} = (1-r)\boldsymbol{p}_{rcop} + r\boldsymbol{p}_{lcop} \quad (4)$$

r is set to 1 in left foot support phase, to 0 in right foot support phase, and linearly interpolated between 0 and 1 in double support phase.

To use toe-heel contact, we offset foot orientation using maximum toe angle offset θ_{toe} and maximum heel angle offset θ_{heel}. We calculate offset by HoffArbib interpolation

for each phase: from 0 θ_{toe} to in (ii) phase, from θ_{toe} to $-\theta_{heel}$ in (iii) phase, from $-\theta_{heel}$ to 0 in (iv) phase. Because (i) and (v) are sole contact, we set foot orientation offset as 0. In the case of the robots without toe joints, toe angle in (ii) and (iii) is treated as foot orientation offset. In the case of the robots with toe joints, toe angle in (ii) and (iii) is treated as toe joint angles and foot orientation offset is set to 0.

2) COG trajectory generation: Here we assume the relationship between COG and ZMP is linear. We calculate COG trajectory based on future reference ZMP trajectory using Preview Control theory [16]. We calculate COG jerk $\dddot{x}_G^{pc}$ minimizing cost function J and obtain COG position x_G^{pc} by integrating it.

$$J = \sum_{j=k}^{\infty} \left\{ Q(x_{zmp}^{j} - x_{zmp}^{abc,j})^2 + R\Delta\dddot{x}_G^{pc,j\,2} \right\} \tag{5}$$

Let $x_{zmp}^{abc,j}$ be x component of reference ZMP at j. We utilize reference ZMP trajectory until 1.6 s future instead of inifinite time future.

3) Base height change: AutoBalancer obtains base link position and orientation from SequencePlayer's HoffArbib-interpolated outputs. Therefore we can change base height smoothly during standing and walking. Note that base height and COG vertical motion influences linearity of the relationship between COG and ZMP. In this paper, we compensate these influence in sensor feedback controller described in the next subsection.

4) COG trajectory offsetting to balance against hand external forces: In the case that the external force at the hands applies to the robot, the robot can maintain full-body balance considering external force influence in COG trajectory calculation [17]. Here we approximately achieve statically feasible COG trajectory by offsetting x_G^{pc} based on external forces.

$$x_G^{abc} = \frac{M x_G^{pc} g + \sum_{i \in hands} \{ -x_i f_{i,z} + (z_i - z_{zmp}^{abc}) f_{i,x} \}}{Mg - \sum_{i \in hands} f_{i,z}} \tag{6}$$

Let M be total mass, $[x_i, y_i, z_i]^T$ be hand position, $[f_{i,x}, f_{i,y}, f_{i,z}]^T$ be hand force. $\boldsymbol{p}_G^{abc} = [x_G^{abc}, y_G^{abc} z_G^{abc}]^T$ is final output COG from AutoBalancer. Although Eq. 5 and Eq. 6 is formulated for x component, we can apply these equations to y component.

5) Full-body Inverse Kinematics: AutoBalancer finally calculates full-body joint angles by solving inverse kinematics constrained by X-Y COG position $\boldsymbol{p}_G^{abc}$, hands and feet position and orientation, base link height and orientation.

6) Example for trajectory generation: Fig.4 shows example trajectories of 2-step walking with toe-heel contact. (a) shows COP position at each foot and reference ZMP is calculated from COP offset in (b). (c) is foot orientation offset and AutoBalancer calculated smooth foot trajectory to use toe-heel contact.

D. Stabilizer

Stabilizer stabilizes robot's standing balance and walking motion. In this formulation, we approximate the robot as

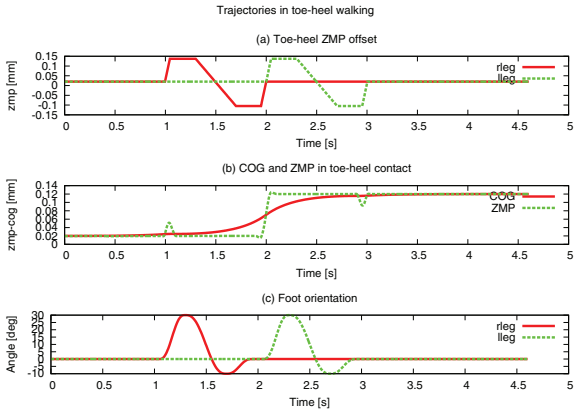


Fig. 4. Walking trajectories using toe-heel contact
(a) COP offsets, (b) ZMP-COG trajectories, and (c) Foot angle offset
Step time is 1.0 s. Toe-heel transition times are (i) 0.05 s, (ii) 0.25 s, (iii) 0.4 s, (iv) 0.25 s, and (v) 0.05 s. COP offset for toe edge is 0.137 m and that for heel edge is -0.10 m.

Linear Inverted Pemdulum (LIP) Model[18] and Stabilizer modifies robot's joint angles to track reference ZMP and COG from AutoBalancer.

1) LIP state feed-back control: In LIP control, we construct feed-back controller to track reference COG, COG velocity, and ZMP. As reference values, we use p_G^{pc} for COG and p_{zmp}^{abc} for ZMP. We estimate actual values based on encoder values from RobotHardware, external force measurements from RemoveForceOffset, and estimated attitude from KalmanFilter.

State equation for LIP and feed-back law in x component are the following:

$$\frac{d}{dt}\boldsymbol{X} = \boldsymbol{A}\boldsymbol{X} + \boldsymbol{B}u \tag{7}$$

$$u = x_{zmp}^{abc} + \boldsymbol{K}(\boldsymbol{X}^{abc} - \boldsymbol{X}) \tag{8}$$

Here let $\boldsymbol{X} = [x_G, \dot{x}_G, p_x]^T$ be state vector including COG, COG velocity, and ZMP. Please see [18] for detailed derivation of equation and calculation for $\boldsymbol{A}$ and $\boldsymbol{B}$.

2) Force-moment distribution based on Quadratic Programming: Distribute total force and total moment equivalent to new ZMP u into foot force and moment of both feet. We approximate foot sole plane as projected one onto horizontal plane.

Let $\boldsymbol{p}_{i,j}$ be j-th vertex of i-th foot convex hull. Let $\boldsymbol{f}_{i,j}$ be normal force at the vertex. Total force and moment holds the following equation:

$$\boldsymbol{F} = \boldsymbol{G}\boldsymbol{f} \tag{9}$$

$$\boldsymbol{f} \geq \boldsymbol{0} \tag{10}$$

Here $\boldsymbol{f} = [f_{0,0,z}, \ldots, f_{N_f-1,N_v-1,z}]^T$ is all vertex forces. Let N_f be the number of foot and N_v be the number of vertex. Total vertical force and horizontal moment is $\boldsymbol{F} = [Mg, 0, 0]^T$. $\boldsymbol{G}$ holds the following equations:

$$\boldsymbol{G} = \begin{bmatrix} 1 & \cdots & 1 \\ (p_{0,0,y} - p_{z,y}) & \cdots & (p_{N_f-1,N_v-1,y} - p_{z,y}) \\ -(p_{0,0,x} - p_{z,x}) & \cdots & -(p_{N_f-1,N_v-1,x} - p_{z,x}) \end{bmatrix}$$

Note that when $\boldsymbol{f} \geq \boldsymbol{0}$ is satisfied, COP of each foot included in convex hull of each foot and the foot keeps contact.

To solve $\boldsymbol{f}$, we introduce Quadratic Programming.

$$\min_{\boldsymbol{f}} \sum_{i}^{N_f} ||\boldsymbol{F}_{cop,i} - \boldsymbol{G}_{cop,i}\boldsymbol{f}||^2_{\boldsymbol{W}_{cop,i}} + ||\boldsymbol{f}||^2_{\boldsymbol{W}_f} \quad (11)$$
$$\boldsymbol{F} = \boldsymbol{G}\boldsymbol{f}$$
$$\boldsymbol{f} \geq \boldsymbol{0}$$

The first term of cost function is to minimize distance from reference COP of each foot $\boldsymbol{p}_{[r,l]cop}$. The first term and second term of cost function use weighted reast sqared solution. $\boldsymbol{W}_{cop,i}$ is weighting matrix for i-th COP. Weighing matrix $\boldsymbol{W}_f$ is determined from ZMP. In the case of a bipedal robot, we set $\boldsymbol{W}_f = diag(w_r, ..., w_r, w_l, ..., w_l)$, $w_r = 1/\alpha$, and $w_l = 1/(\alpha - 1)$. $\alpha \in [0, 1]$ is distribution ratio between right foot and left foot sole edges [18]. For Quadradic Programming solver, we use qpOASES[19] library.

Finally, we calculate both foot force and moment from $\boldsymbol{f}$.

$$f_{i,z} = \sum_{j=0}^{N_v-1} f_{i,j,z} \quad (12)$$

$$n_{i,x} = \sum_{j=0}^{N_v-1} -(p_{i,j,y} - p_{cop,y}) f_{i,j,z} \quad (13)$$

$$n_{i,y} = \sum_{j=0}^{N_v-1} (p_{i,j,x} - p_{cop,x}) f_{i,j,z} \quad (14)$$

3) Foot force-moment control: Obtained foot force and moment is given to force controller. Here we use similar equation to Eq. 3 and set $\boldsymbol{m}_{ic}$ as $\boldsymbol{0}$. For vertical force and horizontal moments, we set Eq. 12 as reference force and moment. For moments around the vertical axis, we apply position control ($\boldsymbol{k}_{ic} \rightarrow \infty$). For horizontal forces, we set reference force as zero.

E. EmergencyStopper with Motion Reverting

EmergencyStopper in Fig.3 stops planned motion sequence from SequencePlayer to avoid from falling down because of inadequate planned motion.

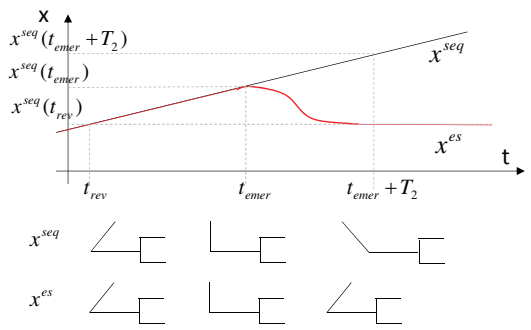


Fig. 5. Emergency Stop by Motion Reverting
Upper graph shows trajectory of input x^{seq} (black) and output x^{es} (red). Lower figure shows robot's pose at t_{rev}, t_{emer}, and $t_{emer} + T_2$.

The key feature of EmergencyStopper is reverting of motion to reduce the cause of falling down. EmergencyStopper receives planned motion and outputs motion to the next modules.

$$x^{es}(t) = \begin{cases} x^{seq}(t) & \text{(If no emergency)} \\ x^{seq}(t_{rev}) & \text{(If emergency and } dt_{rem} \leq 0) \\ \text{HoffArbib}(x^{es}(t-1), x^{seq}(t_{rev}), dt_{rem}) & \\ & \text{(If emergency and } dt_{rem} > 0) \end{cases} \quad (15)$$

Let $x^{es}(t)$ be output from EmergencyStopper at the time t and $x^{es}(t-1)$ be previous output. t_{emer} is the timing when emergency stopping flag is invoked. t_{rev} is the timing EmergencyStopper should be return and $t_{rev} = t_{emer} - T_1$. dt_{rem} is remaining time and $dt_{rem} = (t_{emer} + T_2) - t$. T_1 is user-defined time delay of motion reverting. T_2 is also user-defined time delay to stop motion. By using this equation, EmergencyStopper does not modify input motion when no emergency flag is invoked (the first case of Eq. 15). When emergency flag is invoked, it reverts output motion to previous safe motion at the time t_{rev} by using Eq. 2 interpolation (the third case of Eq. 15). During emergency flag is not resolved and reverting of motion is finished, it keeps outputting previous safe motion at the time t_{rev} (the second case of Eq. 15). In this case, we assume that unexpected reaction force increases near the timing at t_{emer} and reaction force at t_{rev} is smaller than that at t_{emer}. Therefore by reverting motion from $x^{seq}(t_{emer})$ to $x^{seq}(t_{rev})$ we can reduce unexpected reaction force and possibility of robot's tilting and overloads. In this case, x^{es} is used for SequencePlayer outputs such as joint angles, base link position and orientation, and hand/foot wrenches.

For emergency flag, we employ the error of Capture point [3]. In Fig.3, Stabilizer calculates the Capture point error between desired Capture point from x_G^{abc} and $\dot{x}_G^{abc}$ and actual Capture point from x_G and $\dot{x}_G$. Stabilizer also outputs emergency flag to EmergencyStopper by thresholding the error. By using motion reverting, because unexpected reaction forces reduces, motion error such as Capture point error is expected to decrease.

IV. Experiments

In this section, we explain application to proposed controllers and emergency stop functionality to the real robot. First, we explain application to HRP2 robot. Next, we explain application of controllers to several manipulation and locomotion tasks. Finally, we evaluate proposed emergency stopper during real robot experiments.

A. Summary of Hardware Configuration

We applied our controller architecture to HRP2-JSKNT which is the extension of HRP2-JSK[20] (Fig.6).

Extension from original HRP2[21] is (a) additional wrist pitch joints and toe joints are added, (b) 6DOF three-fingered hands similar to [22]. Originally HRP-2 has 6D force/torque sensor at each wrists and ankles and gyro sensor and acceleration sensor at body link. HRP-2 utilizes Art-Linux as real-time OS and controller loop is 4 ms. For visual sensors, we equip with multisense SL[3] on robot's head and

[3]http://carnegierobotics.com/multisense-sl/

high-resolution camera with fish-eye lens at robot's chest. We employ high-power Lithium Ferrite battery Table.II instead of original HRP-2 battery. We added additional battery to robot's hip to be activated more than 90 minutes.

TABLE I

HARDWARE DATA OF HRP ROBOTS

Robot	HRP2-JSKNT	Multi-fingered Hand
Height [mm]	1632	120
Width [mm]	630	115
Depth [mm]	463	150
Weight [kg]	61	1.3
DOF	46	6

TABLE II

BATTERY SPECIFICATIONS

Volt. (Total/Cell)	Cell#	Current (Max/Cont.)	Capacity
52.8 V / 3.3 V	16	120 A / 70A	2.5 Ah

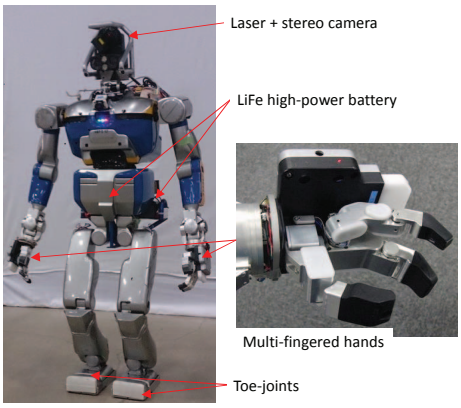


Fig. 6. HRP2-JSKNT with Hardware Extensions
HRP2-JSKNT with multi-fingered hands, high-power batteries, head with stereo camera and laser.

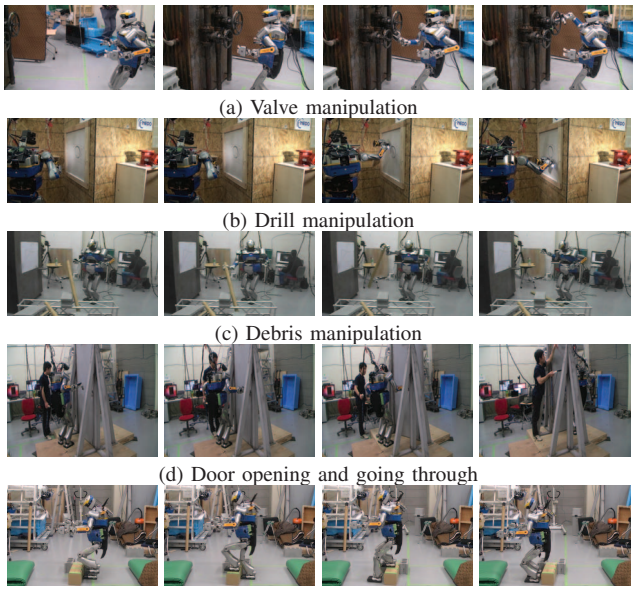
B. Application to Darpa Robotics Challenge Scenario

We apply our controllers to DRC related tasks (Fig.7).

1) Vehicle driving task: In the case of vehicle driving task, Operator outputs initial joint angles to SequencePlayer to seat the vehicle. Then according to Operator's teleoperation, Planner solves arm inverse kinematics and calculates leg joint angles to manipulate car handle and foot pedal based on vehicle kinematics model. Planner outputs joint angles to SequencePlayer.

2) Valve, drill, and door manipulation: We introduce configuration in manipulation tasks for valve, drill, and doors. These task involves both locomotion to desired position and manipulation. First, Planner plans standing position in which the robot can solve whole manipulation motions by considering full-body inverse kinematics. Planner sends goal standing position command to AutoBalancer and the robot executes walking motion. Then, Planner plans motion sequence to manipulate objects. In the case of drill manipulation, Planner sends hand reference forces to balance against holding drill weight. Therefore, the robot can walk with holding drill. Through these manipulation tasks, ImpedanceController and Stabilizer are always activated.

3) Terrain walking and stair climbing: In the case of rough terrain walking and stair climbing, a task is devided into two phase: looking-ground phase and walking phase. In looking-ground phase, the robot looks at ground and stair surface and obtains point cloud information. In this case, Planner or Operator commands joint angles for looking. In walking phase, Planner or Operator commands key-pose joint angles for each steps and footsteps based on point cloud recognition. These commands are sent to SequencePlayer and AutoBalancer. Even if recognition error occurs, Stabilizer keeps balance to track desired COG and ZMP from AutoBalancer.



(a) Valve manipulation

(b) Drill manipulation

(c) Debris manipulation

(d) Door opening and going through

(e) 20cm wall traversing

Fig. 7. Manipulation and Locomotion Tasks

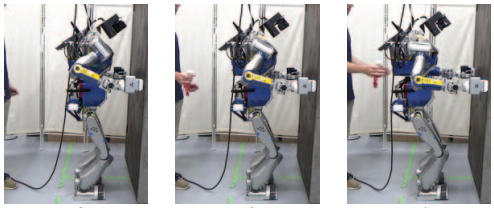
C. Evalation for Emergency Stopping

1) Autonomous Emergency Stopping of Planned Motion: We verified emergency stopping of planned motion through the real robot experiments. In these experiments, we deactivated ImpedanceController and activated AutoBalancer and Stabilizer. We put the robot in front of the wall.

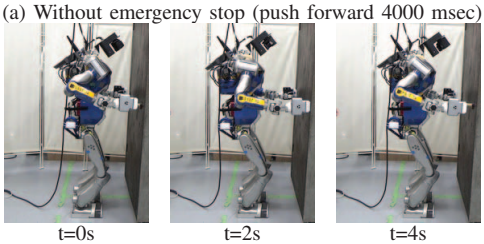
In Fig.8 (a) and (b), Planner send joint angle command to push the hands 0.25 m forward in 4 s (Fig.9(α)). Because of this command, the hands collided with the wall. In (a), we did not use emergency stop functionality and the robot started to fall after t=4 s. Fig.10 (a) shows corresponding Capture point error (red) and the rear edge of foot support polygon. In this graph, the error reached to the edge. This experiment shows balancing stabilizing controller could not compensate for unexpected contact and reaction forces from the environment.

In Fig.8 (b), the robot kept standing thanks for emergency stopping. The robot detected emergency flag and reverted motion to the left figure of Fig.9(α) instead of continuing planned motion in the right figure of Fig.9(α). Fig.10 (b) shows corresponding error graph. In t=3.7s, the error exceeded threshould (blue) and emergency flag was invoked (pink). Note that the error kept growing to about 80 mm after emergency flag was invoked. Thanks for retrieving motion, the error decreased.

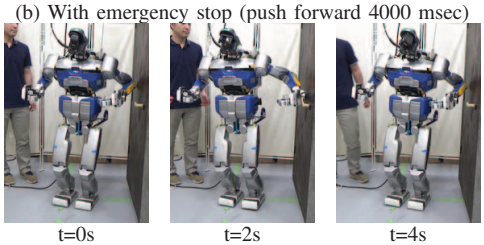
Fig.8 (c) shows the results in which Planner send joint angle command to push the left hand 0.25 m leftward in 2 s (Fig.9(β)). The robot also detected emergency flag and reverted its motion. In this case, the robot successfully stopped planned motion. Fig.10 (c) is corresponding graph. In t=3.2s, the error exceeded threshold and emergency flag was invoked. This experiments shows the robot could stop in the case that it collided with the environment at the contact points with no force sensors.



(a) Without emergency stop (push forward 4000 msec)



(b) With emergency stop (push forward 4000 msec)



(c) With emergency stop (push side 2000 msec)

Fig. 8. Verification of Emergency Stopping of Planned Motion

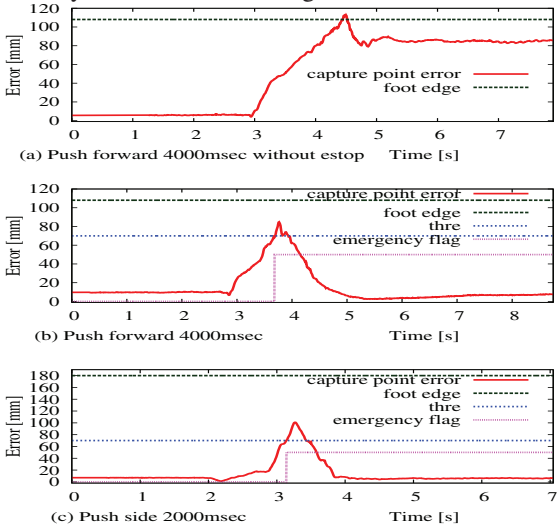

(α) For Fig.8 (a) and (b) (β) For Fig.8 (c)

Fig. 9. Planned Motion for Pushing Verification

2) Autonomous Emergency Stopping during Manipulation Tasks: We verified emergency stopping through DRC manipulation tasks. Here we evaluated in valve manipulation and door manipulation. ImpedanceController, Stabilizer, AutoBalancer ware activated through experiments. First, we confirmed that the robot could successfully achieved valve rotating and door manipulation if we disturbed manipulation.

Next, we intentionally added disturbance to planner to evaluate robustness of proposed emergency stopping. For the valve task, we added disturbance by modifying the results from point-cloud-based valve recognition by Graphical User Interface (GUI) . In Fig.11 (a-1), we moved target valve position to forward. in Fig.11 (b-1), we moved target valve position to leftward. In Fig.11 (a-1) and (b-1), "Recognized" is the original recognition result and "Disturbed" is disturbed result. Position difference ware 131.13 mm forward and 32.92 mm upward for (a-1) and 194.54 mm leftward for (b-1). Fig.13 (α) and (β) ware corresponding planned motion. In (a-2), the robot's body tilted backward because of large reaction force from the valve. In (b-2), the robot's body tilted leftward. In both case, the robot avoided from falling down thanks for emergency stopping of planned motion.

For the door task, we utilized pulling rotational door and we added disturbance by applying motion plan for door pushing (Fig.13(γ)). In Fig.12, the robot reached to the knob and pushed the door. The robot tilted backward because of reaction forces from the door. In this case, the robot also successfully avoided from falling down.



(a) Push forward 4000msec without estop
(b) Push forward 4000msec
(c) Push side 2000msec

Fig. 10. Capture Point Error and Emergency Checking

V. Conclusion

In this paper, we proposed a method to construct multi-layered humanoid's real-time controllers including emergency stop functionality. We explained structure of real-time controller including several modules for manipulation and locomotion tasks and application of HRP-2 hardware. We also discussed manipulation failure and proposed emergency stopping controller using motion reverting. We experimentally verified proposed controllers ware able to prevent the robot from falling down even if recognition error and inadequate planning exist and even if the robot made contact with environments at the points without force sensors.

In future work, we need to generalize emergency stopping strategy in terms of disturbance absorption during standing and stepping. Our proposed emergency stopping is required to be analyzed quantitatively to decide parameter such as reverting time and robot's motion stability.

A part of this work was supported by JSPS KAKENHI Grant Numbers 26220003, 26700022.

References

[1] M. Morisawa, S. Kajita, K. Harada, K. Fujiwara, F. Kanehiro, K. Kaneko, and H. Hirukawa, "Emergency stop algorithm for walking humanoid robots," in *Proceedings of the 2005 IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2005, pp. 31–37.

[2] K. Kaneko, F. Kanehiro, S. Kajita, and M. Morisawa, "Motion suspension system for humanoids in case of emergency; real-time motion generation and judgment to suspend humanoid," in *Proceedings of the 2005 IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2005, pp. 5496–5503.

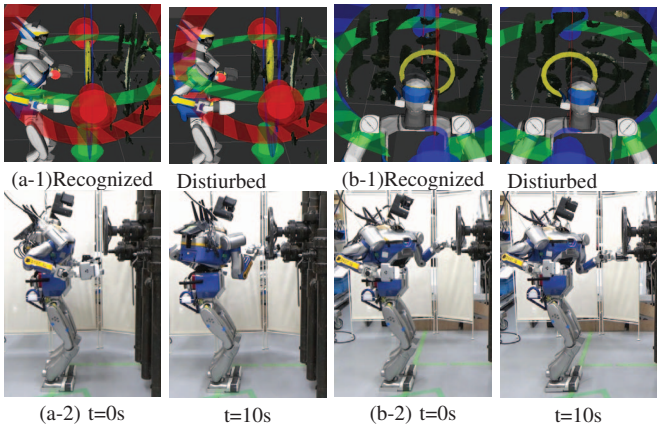


Fig. 11. Verification of Emergency Stopping during Valve Rotating
Yellow torus is a visual recognition result. Other red and green objects are GUI objects.

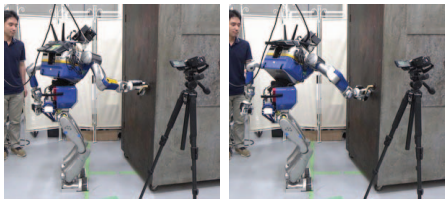


Fig. 12. Verification of Emergency Stopping during Door Manipulation

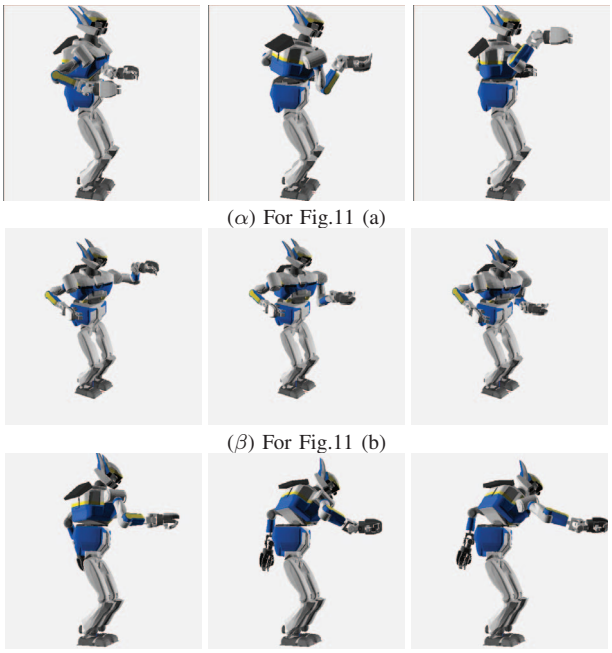


Fig. 13. Planned Motion for Valve Rotating and Door Pushing

[3] J. Pratt, S. Drakunov, and A. Goswami, "Capture point: A step toward humanoid push recovery," 2006, pp. 200–207.

[4] H. Hirukawa, S. Kajita, F. Kanehiro, K. Kaneko, and T. Isozumi, "The human-size humanoid robot that can walk, lie down and get up," *The Internationaltional Journal of Robotics Research*, vol. 24, no. 9, pp. 755–769, 2004.

[5] S.-K. Yun, A. Goswami, and Y. Sakagami, "Safe Fall: Humanoid robot fall direction change through intelligent stepping and inertia shaping," in *Proceedings of The 2009 IEEE International Conference on Robotics and Automation*, 2009, pp. 781–787.

[6] R. Ueda, M. Murooka, Y. Ohara, I. Kumagai, R. Terasawa, Y. Furuta, K. Kojima, T. Karasawa, F. Sugai, S. Iwaishi, S. Nozawa, Y. Kakiuchi, K. Okada, and M. Inaba, "User interface for tele-operating and massive data visualization of humanoid robot and system integration over narrow and unreliable network communication for drc competition," in *Proceedings of the 2015 IEEE-RAS International Conference on Humanoid Robots*, 2015.

[7] M. Morisawa, F. Kanehiro, K. Kaneko, N. Mansard, J. S. and Eiichi Yoshida, K. Yokoi, and J.-P. Laumond, "Combining suppression of the disturbance and reactive stepping for recovering balance," 2010, pp. 3150–3156.

[8] M. Morisawa, K. Kaneko, F. Kanehiro, S. Kajita, K. Fujiwara, K. Harada, and H. Hirukawa, "Motion planning of emergency stop for humanoid robot by state space approach," in *Proceedings of the 2006 IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2006, pp. 2986–2992.

[9] T. Takubo, T. Tanaka, K. Inoue, and T. Arai, "Emergent walking stop using 3-d zmp modification criteria map for humanoid robot," in *Proceedings of The 2007 IEEE International Conference on Robotics and Automation*, 2007, pp. 2676–2681.

[10] T. Sugihara, "Standing stabilizability and stepping maneuver in planar bipedalism based on the best com-zmp regulator," in *Proceedings of The 2009 IEEE International Conference on Robotics and Automation*, 2009, pp. 1966–1971.

[11] N. ANDO, T. SUEHIRO, K. KITAGAKI, T. KOTOKU, and W.-K. Yoon, "RT-Middleware: Distributed Component Middleware for RT (Robot Technology)," in *Proceedings of the 2005 IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2005, pp. 3555–3560.

[12] E. S. of Humanoid Robot HRP-1S, "Kazuhito yokoi and fumio kanehiro and kenji kaneko and shuuji kajita and kiyoshi fujiwara and hirohisa hirukawa," *The Internationaltional Journal of Robotics Research*, vol. 23, no. 4-5, pp. 351–362, 2004.

[13] K. Okada and M. Inaba, "A hybrid approach to practical self collision detection system of humanoid robot," in *Proceedings of the 2006 IEEE/RSJ International Conference on Intelligent Robots and Systems*, October, 2009, pp. 3952–3957.

[14] B. Hoff and M. Arbib, "Models of trajectory formation and temporal interaction of reach and grasp," *Journal of MotorBehavior*, vol. 25, no. 3, pp. 175–192, 1993.

[15] K. Nishiwaki and S. Kagami, "Trajectory design and control of edge-landing walking of a humanoid for higher adaptability to rough terrain," in *Proceedings of the 2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2012, pp. 3432–3439.

[16] S. Kajita, F. Kanehiro, K. Kaneko, K. Fujiwara, K. Harada, K. Yokoi, and H. Hirukawa, "Biped walking pattern generation by using preview control of zero-moment point," in *Proceedings of The 2003 IEEE International Conference on Robotics and Automation*, Sep 2003, pp. 1620–1626.

[17] K. Harada, S. Kajita, K. Kaneko, , and H. Hirukawa, "Pushing manipulation by humanoid considering two-kinds of zmps," in *Proceedings of The 2003 IEEE International Conference on Robotics and Automation*, 2003, pp. 1627–1632.

[18] S. Kajita, M. Morisawa, K. Miura, S. Nakaoka, K. Harada, K. Kaneko, F. Kanehiro, and K. Yokoi, "Biped walking stabilization based on linear inverted pendulum tracking," in *Proceedings of the 2010 IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2010, pp. 4489–4496.

[19] H. Ferreau, C. Kirches, A. Potschka, H. Bock, and M. Diehl, "qpOASES: A parametric active-set algorithm for quadratic programming," *Mathematical Programming Computation*, vol. 6, no. 4, pp. 327–363, 2014.

[20] K. Okada, T. Ogura, A. Haneda, J. Fujimoto, F. Gravot, and M. Inaba:, "Humanoid Motion Generation System on HRP2-JSK for Daily Life Environment," in *International Conference on Mechatronics and Automation*, July, 2005, pp. 1772 – 1777.

[21] K.Kaneko, F.Kanehiro, S.Kajita, H.Hirukawa, T.Kawasaki, M.Hirata, K.Akachi, and T.Isozumi, "Humanoid Robot HRP-2," in *Proceedings of The 2004 IEEE International Conference on Robotics and Automation*, 2004, pp. 1083–1090.

[22] K. KANEKO, K. HARADA, F. KANEHIRO, G. MIYAMORI, , and K. AKACHI, "Humanoid Robot HRP-3," in *Proceedings of the 2008 IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2008, pp. 2471–2478.