2012 IEEE/RSJ International Conference on
Intelligent Robots and Systems
October 7-12, 2012. Vilamoura, Algarve, Portugal

Efficient Reaching Motion Planning and Execution for Exploration by Humanoid Robots

Fumio Kanehiro and Eiichi Yoshida and Kazuhito Yokoi

Abstract— **This paper presents a reaching motion planning and execution framework tailored for exploration missions by human-operated humanoid robots in hazardous environments such as nuclear plants. This framework offers low-level but practical autonomy that allows the robot to plan and execute simple tasks, such as reaching a target object, within a reasonable amount of time. The human operator benefits from the efficiency of the framework to maneuver the robot without waiting for the planning results for minutes. The efficiency improvement is achieved in the following two phases. In the first phase, a reaching motion is planned quickly through approximation of mass distribution and kinematic structure to apply analytical solutions of inverse kinematics. Supposing that the robot is working in environments not completely known, the proposed planner can use measured voxel maps. In the second phase, the planned path is executed while compensating the approximation error in real time without violating other constraints. We confirm through simulations that a reaching motion for the HRP-2 humanoid with 30 DOFs in a constrained environment with pipes is planned in around one second. The simulation results also validate the efficiency of execution with real-time error compensation.**

I. INTRODUCTION

The anthropomorphic structure of humanoid robots makes humanoid robots suitable to work in the environment designed for humans. One of many possible application which takes advantage of this feature would be exploring hazardous environments, such as the Fukushima nuclear plant, through human teleoperation. Although mobile robots equipped with manipulators have been already deployed, there are tasks they can not accomplish. These tasks include accessing valves placed beyond their reachable region, exploring places that can be reached only by ladders. These limitations come from the fact that the environment is designed for humans.

Since a humanoid robot has many degrees of freedom (DOFs), it is difficult to fully teleoperate using ordinary input devices, especially in a highly constrained environment as shown in Fig. 1. Low-level autonomy is therefore required for easy teleoperation: when an operator gives a walking direction, the robot must be able to generate appropriate walking motions not only on flat floors but also on slopes and stairs, or when an operator indicates a target object in the robot's view, it must be able to plan and execute simple tasks such as opening/closing valves, turning on/off switches, and so on. In this paper, we focus on reaching motion planning and execution as a representative basic functions for such humanoids.

The authors are with Intelligent Systems Research Institute, National Institute of Advanced Industrial Science and Technology(AIST), Tsukuba Central 2, 1-1-1 Umezono, Tsukuba, Ibaraki, 305-8568 Japan {f-kanehiro, e.yoshida, kazuhito.yokoi}@aist.go.jp

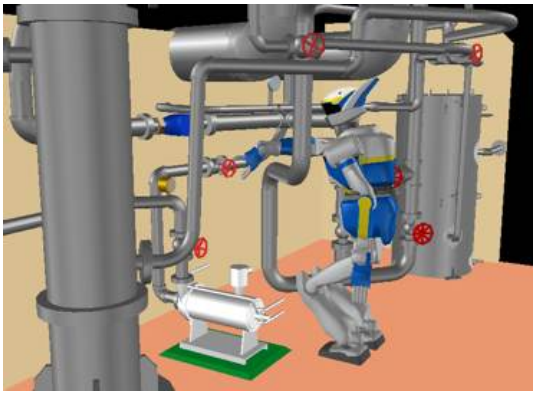

Fig. 1. A humanoid robot HRP-2 reaching its right arm for a red valve in an industrial plant

Concerning the planning in practical use, random sampling based methods such as RRT[1] and PRM[2] are very good at quickly finding a solution in high dimensional configuration space including those of humanoid robots. But when applied to humanoid robots, the following two issues become critical.

The first issue is that balance must be taken into account in addition to joint limits and collision avoidance since a humanoid robot is an underactuated robot. Kuffner et al. proposed a dynamic motion planning method[3] which plans a static motion first and then speeds it up using a dynamics filter as long as it is collision-free. Yoshida et al. proposed a walking motion planning method[4] which plans a collision-free upper body motion first and then gives it to a walking pattern generator and reshapes the path if collisions are caused by swinging motion of the waist. Harada et al. also proposed a walking motion planning method[5] which avoids collisions using the upper body. Dalibard et al. proposed a motion planning method[6] which grows a search tree in a constrained manifold using an whole body inverse kinematics.

The second issue is that the goal is not given in the joint space, but in the workspace. The goal might not be a single position and orientation but a manifold in 6D space. Several methods have been already proposed for this issue. Weghe et al. proposed JT-RRT[7] which interleaves growing a tree randomly and growing a tree towards a goal using Jacobian transpose. An advantage of the method is an analytical solution of inverse kinematics is not required. Berenson et al. proposed IKBiRRT[8] which interleaves sampling goals

and growing a tree. Goal configurations are obtained by computing a configuration which corresponds to a goal sampled from a manifold called *Workspace Goal Region* using inverse kinematics.

Although many general planning algorithms have been already proposed in the literature, there are still two practical and critical problems to be addressed. The first one is time spent for the planning. Since the robot is teleoperated by a human, it is a very important factor. The planning must be done within a few seconds in order not to keep the human operator waiting too long. The second is that a polyhedral model of the environment is not given a priori. The environment is measured by sensors on the robot and its model is constructed while the robot is exploring.

The proposed framework takes those problems into account in two phases, planning and execution. In the planning phase, a reaching motion is planned quickly using a "light" configuration projector with approximated mass distribution and kinematic structure. In the execution phase, the approximation error is compensated without breaking other constraints by solving whole body inverse kinematics during execution.

The rest of the paper is organized as follows. Section II explains how to create collision models of the environment from measured data which are used for collision detection and distance computation. A reaching motion planning method which plans quickly utilizing analytical solutions of inverse kinematics is presented in Section III. Section IV presents a reaching motion execution method. Section V shows some simulation results. Finally Section VI concludes the paper.

II. Collision models for the environment and the robot

We assume that the environment around the robot is measured by sensors such as a stereo vision system or a laser range finder while the robot is exploring and those measurements are accumulated as a voxel map.

The simplest way to create a collision detection model from the voxel map is assigning a small cube to each voxels. But this method not only consumes large amount of memory, but also cubes are not strictly convex, which is not a suitable shape for the collision avoidance algorithm used in the execution phase. We therefore represent the environment by a sphere tree[11] instead. A sphere is assigned to each voxels and a sphere tree is constructed from those spheres. Diameters of those spheres are equivalent to the resolution of the voxel map in this work. The safety margin of a planned motion can be increased by using larger diameters. The sphere tree is used to detect collisions in the planning phase and to compute distances to reshape the path to avoid collisions caused by balance compensation.

The sphere tree is constructed by a top down approach. First, an axis aligned bounding box(AABB) of spheres is computed. Spheres are split into two groups at center of gravity of spheres along the longest axis of the AABB. This procedure is repeated until every leaf nodes has only one sphere. A radius of a tree node is determined by comparing two radii: for a sphere which bounds AABB and which bounds two child bounding spheres. The smaller is chosen after comparison.

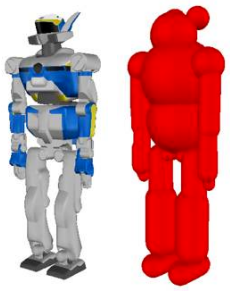

Fig. 2. Left:original polyhedral model of HRP-2, Right:approximation model for collision detection and distance computation

The robot shape is also approximated to detect collisions quickly and make it easy to compute distances. Fig. 2 shows a polyhedral model of a humanoid robot HRP-2(left) and its approximation model(right). The robot shape is approximated by spheres and capped cylinders since it is easy to compute distances. To make bounding volumes tighter, STP-BV[12] will be used in the future.

III. Reaching motion planning

A. Inverse Kinematics

In general, time consuming processes of the motion planning are (1) a collision detection between the robot and the environment and (2) a projection of a sampled configuration onto constrained manifolds. Since these processes are called so many times to find an initial path and optimize it, they should be done efficiently. Unfortunately, the configuration projection tends to be computationally heavy because a humanoid robot must respect many constraints while reaching such as feet position/orientation and center of gravity (COG) position. Due to high redundancy, the usual approach is to solve whole-body inverse kinematics numerically through iterative convergence computation. It is however obvious that analytical solutions of inverse kinematics should be used for quick planning.

B. Sampling Goals

One of features of a reaching motion planning problem is that a goal is given in the workspace, not in the joint space. Moreover, the goal is not necessarily given as a point in 6D(position/orientation) space but as a manifold which is called *Workspace Goal Region(WGR)*[8]. We define **WGR** as a set of ranges of x, y, z components of position and ϕ, θ, ψ components of orientation. For example, when a target is a cylindrical object, an end-effector can approach from any direction around the axis and the grasping height needs not to be fixed to a single point. In this case, upper and lower limits for x, y, ϕ and θ are the same with given specific values and those for z and ψ have different values.

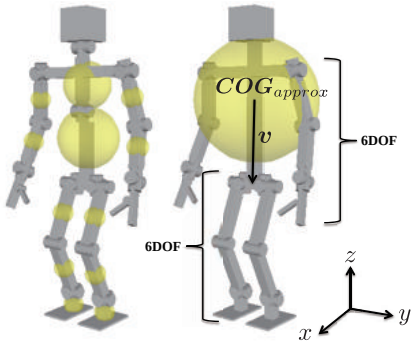

Fig. 3. The original kinematic chain(left) and the simplified kinematic chain used to find goal postures(right). Some of joints are fixed and the original kinematic chain is split into four 6-DOF chains connected through the trunk. Yellow spheres show COG positions and masses of each link. Distributing masses(left) are assumed to be concentrating on the trunk(right). $\boldsymbol{v}$ is a vector from $\boldsymbol{COG}_{approx}$ to the origin of the trunk link.

The goal posture can be obtained by sampling **WGR** first and then computing the corresponding humanoid posture as follows:

1) Sample an end-effector position/orientation in the given **WGR** and joint angles in the joint space
2) Move an end-effector to the sampled goal, move feet to positions/orientations to keep given stance and move COG above the support polygon using whole body inverse kinematics

The second step of the procedure is computationally expensive as mentioned earlier. We fix some of the joints and split the robot's kinematic chain into sub-chains in order to apply analytical solutions of inverse kinematics, as shown in Fig. 3. Neck, waist and finger joints are fixed and 6-DOF kinematic chains are connected to the trunk. In this section, we assume that arms and legs are composed of six DOFs. This is the case of our humanoid robot, HRP-2[9]. If a robot has more degrees of freedom, this method can be used by fixing some of the joints or adding those joints as new dimensions of the configuration space.

Yellow spheres in Fig. 3 show COG positions and masses of each link. We assume that the whole mass concentrates on a point fixed to the trunk link. Hereafter, we call the COG position computed using distributed masses *the exact COG position* and a concentrated mass *the approximated COG position*. The latter is denoted by $\boldsymbol{COG}_{approx}$. Based on this assumption, we can determine the trunk horizontal position easily so that $\boldsymbol{COG}_{approx}$ does not move. And then joint angles of arms and legs are computed by solving analytical solutions of inverse kinematics using the trunk position/orientation and end-effector positions/orientations.

Using this simplified kinematic chain and the approximated COG position, the configuration space to find goal postures is defined as follows.

$$\boldsymbol{q}_{goal} = (\boldsymbol{p}_e^T \ \boldsymbol{rpy}_e^T \ z_t \ \boldsymbol{rpy}_t^T)^T \tag{1}$$

This is a concatenation of an end-effector position $\boldsymbol{p}_e = (x_e, y_e, z_e)^T$, an end-effector orientation $\boldsymbol{rpy}_e = (\phi_e, \theta_e, \psi_e)^T$, the height of the trunk z_t and an orientation of the trunk $\boldsymbol{rpy}_t = (\phi_t, \theta_t, \psi_t)^T$. A sampled configuration is projected by Algorithm 1. All positions, orientations and vector in Algorithm 1 are expressed in the world coordinates. First, the trunk position $\boldsymbol{p}_t$ is determined so that the horizontal position of $\boldsymbol{COG}_{approx}$ does not move (from line 2 and 4). $\boldsymbol{v}$ is a vector from $\boldsymbol{COG}_{approx}$ to the origin of the trunk link. And then arms and legs are checked if they can reach specified positions/orientations. Solve{RightArm,LeftArm,RightLeg,LeftLeg}IK() are functions to solve inverse kinematics of a kinematic chain analytically. $\boldsymbol{p}_{rf}, \boldsymbol{R}_{rf}, \boldsymbol{p}_{lf}$ and $\boldsymbol{R}_{lf}$ are feet positions and orientations. When Algorithm 1 returns True and it is collision-free, a goal posture is obtained.

Algorithm 1 ProjectConfig($[\boldsymbol{p}_e^T, \ \boldsymbol{rpy}_e^T, \ z_t, \ \boldsymbol{rpy}_t^T]$)

1: $\boldsymbol{R}_e \leftarrow$ RollPitchYaw($\boldsymbol{rpy}_e$)
2: $\boldsymbol{R}_t \leftarrow$ RollPitchYaw($\boldsymbol{rpy}_t$)
3: $\boldsymbol{p}_t \leftarrow \boldsymbol{R}_t\boldsymbol{v} + \boldsymbol{COG}_{approx}$
4: $\boldsymbol{p}_t[2] \leftarrow z_t$
5: **if** SolveRightArmIK($\boldsymbol{p}_t$, $\boldsymbol{R}_t$, $\boldsymbol{p}_e$,$\boldsymbol{R}_e$) $\neq$ True **then**
6: **if** SolveLeftArmIK($\boldsymbol{p}_t$, $\boldsymbol{R}_t$,$\boldsymbol{p}_e$,$\boldsymbol{R}_e$) $\neq$ True **then**
7: **return** False
8: **end if**
9: **end if**
10: **if** SolveRightLegIK($\boldsymbol{p}_t$, $\boldsymbol{R}_t$, $\boldsymbol{p}_{rf}$,$\boldsymbol{R}_{rf}$) $\neq$ True **then**
11: **return** False
12: **end if**
13: **if** SolveLefLegIK($\boldsymbol{p}_t$, $\boldsymbol{R}_t$, $\boldsymbol{p}_{lf}$,$\boldsymbol{R}_{lf}$) $\neq$ True **then**
14: **return** False
15: **end if**
16: **return** True

Following this procedure, 100 goal postures to reach the red valve HRP-2 is reaching in Fig. 1 are generated to confirm that difference between the approximated COG position and the exact one is small enough. Fig. 4 shows exact COG positions relative to approximated COG positions in the horizontal plane for those goal postures. The exact COG positions move a few centimeters forward(direction+ x) since the robot must stretch its arm forward to reach the target valve. Differences are within a few centimeters and they are small enough compared to the size of the support polygon(23×34[cm]).

C. Planning a Motion

A reaching motion is planned using RRT-connect[10]. The initial configuration and goals obtained in III-B are used as seeds for search trees. For the reaching motion planning as well, only analytical solution of inverse kinematics is used to find solutions quickly. The configuration space for reaching motions is defined as follows:

$$\boldsymbol{q}_{plan} = (\boldsymbol{q}_{arm}^T \ z_t \ \boldsymbol{rpy}_t^T)^T \tag{2}$$

where $\boldsymbol{q}_{arm}$ is a vector of joint angles of an arm used to reach. While RRT-connect grows a tree, the trunk horizontal

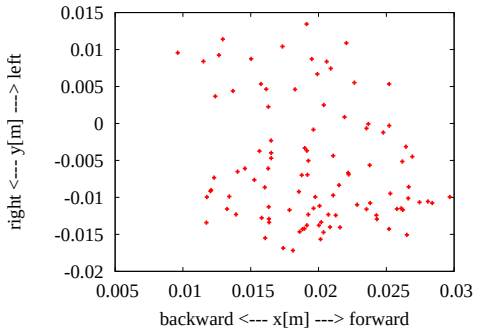

Fig. 4. Differences between the approximated COG positions and the exact ones for sampled goal postures

position is determined in the same way with Algorithm 1 to keep balance. And leg joint angles are computed by solving analytical solutions of inverse kinematics.

IV. EXECUTION WITH REAL-TIME COMPENSATION

The planned path is a sequence of postures which are collision-free and roughly statically stable. Although the approximation error of COG position is small enough as shown in Fig. 4, it is preferred to maintain the horizontal position of the exact COG to maximize stability margin. The planned path is thus executed while compensating the approximation error in real time without breaking other constraints such as collisions and end-effector positions.

As the path is a sequence of discrete configurations, it should be transformed into an executable trajectory first. Durations to move between neighboring postures in the path are computed as the maximum difference of joint angles divided by a given joint velocity. An initial trajectory is then obtained by interpolating postures using clamped cubic spline.

In order to compensate the approximation error of COG without breaking other constraints, all joints(except fingers) are taken into account. Although required modification is expected to be small, it might be impossible to respect all constraints at the same time. In order to continue execution of the trajectory even in such cases, constraints are prioritized and a feasible motion is obtained by solving whole-body inverse kinematics. Since some of constraints such as joint limits and collision avoidance are inequality constraints, a prioritized inverse kinematics solver which is able to consider both of equality and inequality constraints[13] is used. As shown later, the solver is efficient enough to generate the executed posture within the low-level control loop (5[ms] in case of HRP-2). We introduce the following four priority levels.

1) Joint limits and collision avoidance constraints have the highest priority since if they are not respected, the robot will damage itself or the environment immediately. Environment spheres within a distance bound are picked up first and *velocity dampers*[14] are inserted between each pair of an environment sphere and a robot geometry. These are inequality constraints.
2) Maintaining feet position/orientation and the exact COG position have the second priority since these are important to keep balance. These are equality constraints.
3) Maintaining a hand position/orientation constraint has the third priority. Although moving a hand to the specified position is the main objective, it has lower priority compared to above two levels since ensuring robot's safety is most important to continue exploration.
4) Residual redundancy is used to realize a posture given by the initial trajectory.

TABLE I

PERFORMANCE OF GOAL SAMPLING

average total goal sampling time	288[ms]
collision detection	0.45[%] of total time called 8125 times(0.016[ms/call])
the number of samples	1308226
used arm(right/left)	16/84
IK failure	arm: 1295881(right)/1278559(left) leg: 18617(right)/2925(left)

V. SIMULATION RESULTS

In order to confirm efficiency of the proposed framework, reaching motions in an industrial plants shown in Fig. 1 are planned and executed on a simulator. In this example **WGR** is given as follows.

$$[0.71\ 0.71; 0.03\ 0.03; 0.9\ 0.9; -\frac{\pi}{2}\ \frac{\pi}{2}; -\frac{\pi}{2}\ \frac{\pi}{2}; -\pi\ \pi] \quad (3)$$

The end-effector position is fixed but its Z axis can be any direction in hemisphere. A sphere tree for the workspace is created from 22,753 voxels. It takes 20[ms] to construct the sphere tree on a modern computer (CPU:Intel Core i7 3.2[GHz]).

Table I shows performance indexes obtained when 100 goal postures are sampled using Algorithm 1. It takes 288[ms] to find a goal on average. Most of the time is spent for solving inverse kinematics and rejecting invalid configurations. More than one million configurations are sampled to get 100 goal postures. It is expected that it takes much more time if numerical solution of inverse kinematics is used.

Table II shows performance indexes obtained when 100 paths are planned for each goals obtained above. Most of the processing time is used by collision detection. Duration of the planning is 429[ms] on average and 707[ms] in the worst case. The total time an operator must wait before a robot start to move is expected to be around 1[s].

Fig. 5 shows snapshots of the executed reaching motion. The right arm passes through narrow passage between a pipe and its lower body and reaches the target position. The planned path consists of nine configurations. Snapshots in Fig. 6 show a reaching motion by the left arm. As can be seen, the robot lowers its body to avoid collisions between

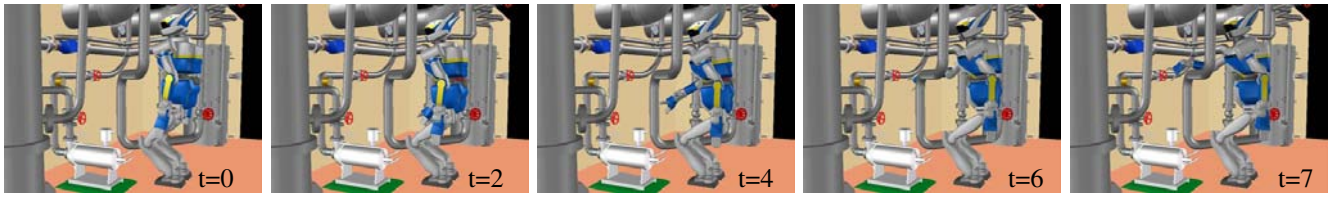

Fig. 5. Execution of a reaching motion by the right arm

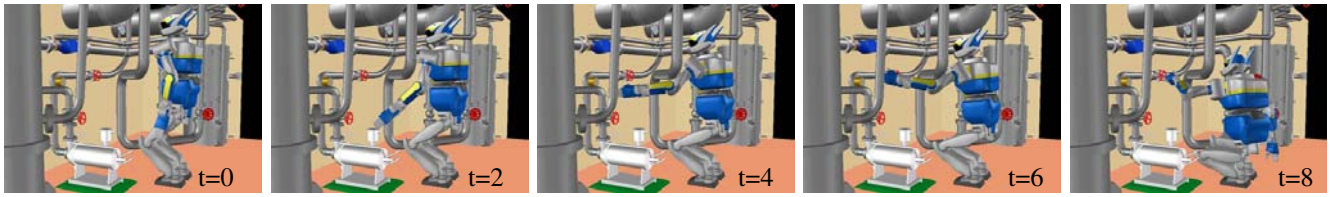

Fig. 6. Execution of a reaching motion by the left arm. The robot lowers its body to avoid its left shoulder colliding with a pipe.

TABLE II
PERFORMANCE OF PATH PLANNING

average total path planning time	total:429[ms] right:549[ms/goal], left:406[ms]
max time	right:692[ms], left:707[ms]
collision detection	77[%] of total time called 1019794times(0.032[ms/call])

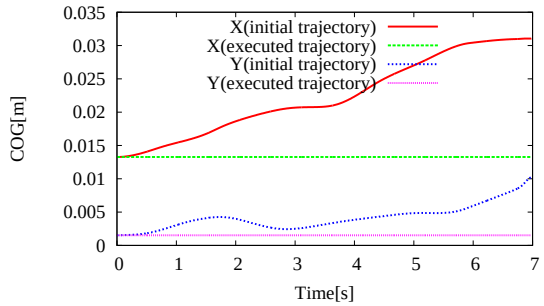

Fig. 7. Comparison of exact COG trajectories. When the COG constraint is enabled, the horizontal position of the exact COG doesn't move.

its left shoulder and a pipe. The planned path consists of 14 configurations.

Fig. 7 shows exact COG trajectories for the reaching motion by the right arm. If the initial trajectory is executed without compensation, the exact COG moves a few centimeters. We can confirm that the horizontal position of the exact COG is maintained after the compensation. In Fig. 8, minimum distances between the robot and the environment are plotted. Negative distances mean that the robot is colliding with the environment. Even though the planner finds a set of collision-free postures, there are collisions in the initial trajectory around 2[s] and 6[s]. There are two causes of these collisions. First, it is due to discrete collision detection. Since collisions are tested for discrete configurations on the planned path, some of collisions occurring between tested configurations might be overlooked. The other cause is the difference of interpolation method. During the planning

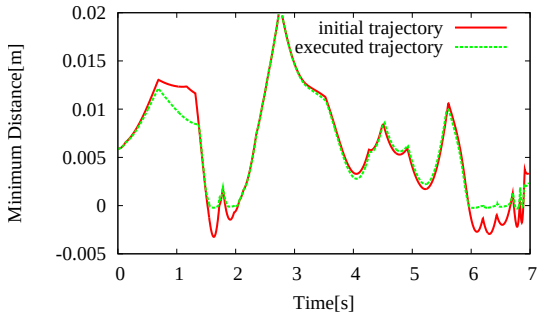

Fig. 8. Minimum distance between the robot and the environment. There are collisions in the initial trajectory around 2[s] and 6[s]. Those collisions are resolved by enabling a collision avoidance constraint.

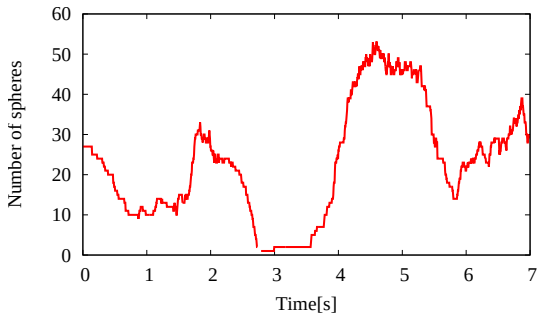

Fig. 9. the number of spheres in a distance bound

phase, configurations are connected by straight lines in the configuration space. But the initial trajectory is generated by interpolating those configurations using clamped cubic spline in order to make the trajectory smooth. These collisions are resolved by enabling a collision avoidance constraint.

Red lines in the red circles of Fig. 11 connect closest points between spheres in a distance bound and robot shapes. Constraints are preventing right shoulder, right elbow and right wrist from colliding with pipes.

Fig. 9 shows the number of spheres in a distance bound and Fig. 10 shows processing time used for finding spheres in a distance bound, solving prioritized inverse kinematics

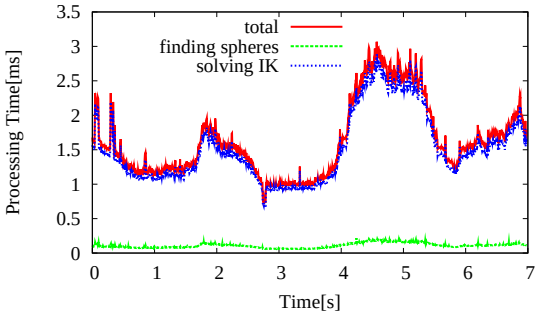

Fig. 10. Processing time used for finding spheres in a distance bound, solving prioritized inverse kinematics and sum of them. The total time is enough short to execute online.

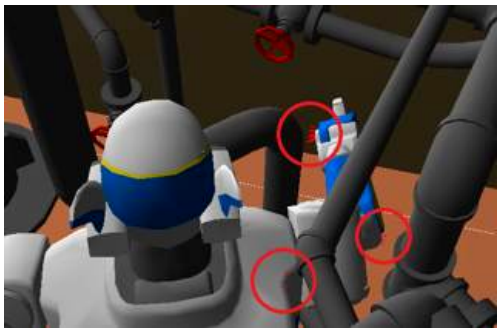

Fig. 11. Collision avoidance constraints generated at 5.62[s]. Right shoulder, elbow and wrist are avoiding collisions.

and sum of them. The processing time is proportional to the number of spheres in a distance bound. Most of the time is used to solve prioritized inverse kinematics. The processing time is enough short to execute online since the control cycle of HRP-2 is 5[ms].

VI. CONCLUSIONS

We presented an efficient framework for reaching motion planning and execution for humanoid robots on exploration missions. Assuming the robot is teleoperated by a human operator, we focused on making planning time short considering a human patience. The framework consists of a planning phase and an execution phase. The former plans a reaching motion quickly utilizing analytical solution of inverse kinematics and the latter executes the planned path by compensating approximation error in real-time while maintaining other constraints. The effectiveness of the framework was confirmed by generating reaching motions in a simulated industrial plant.

Future work includes the following issues. In this work, we did not deal with manipulation of the target object. In the worst case, the robot in the presented example might not be able to open/close the valve if the reached position and orientation are not appropriate. Reaching and object manipulation must be considered in an unified framework in the future work. Another issue concerns how to select a standing position and its stance that are assumed to be given in this work. It is often difficult for a operator to choose appropriate standing position and stance. In the future development we will also investigate automatic decision of standing posture that maximizes the workable range for the robot.

REFERENCES

[1] S. M. LaValle, "Rapidly-exploring random trees: A new tool for path planning," Computer Science Dept., Iowa State University, Tech. Rep. 98-11, 1998.

[2] L.E.Kavraki, P.Svestka, J.-C.Latombe, and M.H.Overmars, "Probabilistic Roadmaps for Path Planning in High-Dimensional Configuration Spaces," *IEEE TRANSACTIONS ON ROBOTICS AND AUTOMATION*, vol. 12, no. 4, pp. 566–580, 1996.

[3] J. J. Kuffner, S. Kagami, and H. I. Koichi Nishiwaki, Masayuki Inaba, "Dynamically-Stable Motion Planning for Humanoid Robots," *Autonomous Robots*, vol. 12, no. 1, pp. 105–118, 2002.

[4] E. Yoshida, C. Esteves, I. Belousov, J.-P. Laumond, T. Sakaguchi, and K. Yokoi, "Planning 3D Collision-Free Dynamic Robotic Motion through Iterative Reshaping," *IEEE Trans. on Robotics*, vol. 24, no. 5, pp. 1186–1198, 2008.

[5] K. Harada, S. Hattori, H. Hirukawa, M. Morisawa, S. Kajita, and E. Yoshida, "Motion Planning for Walking Pattern Generation of Humanoid Robots," in *Proc. of the IEEE/RSJ International Conference on Intelligent Robotics and Systems (IROS'07)*, 2007, pp. 4227–4233.

[6] S. Dalibard, A. Nakhaei, F. Lamiraux, and J.-P. Laumond, "Whole-Body Task Planning for a Humanoid Robot: a Way to Integrate Collision Avoidance," in *Proc. of the IEEE-RAS International Conference on Humanoid Robots*, 2009, pp. 355–360.

[7] M. V. Weghe, D. Ferguson, and S. S. Srinivasa, "Randomized Path Planning for Redundant Manipulators without Inverse Kinematics," in *Proc. of the IEEE-RAS International Conference on Humanoid Robots*, 2007, pp. 477–482.

[8] D. Berenson, S. S. Srinivasa, D. Ferguson, A. Collet, and J. J. Kuffner, "Manipulation Planning with Workspace Goal Regions," in *Proc. of the 2009 IEEE International Conference on Robotics & Automation*, 2009, pp. 1397–1403.

[9] K. Kaneko, F. Kanehiro, S. Kajita, M. Hirata, K. Akachi, and T. Isozumi, "Humanoid Robot HRP-2," in *Proc. of the 2004 IEEE International Conference on Robotics & Automation*, 2004, pp. 1083–1090.

[10] J. James J. Kuffner and S. M. LaValle, "RRT-Connect: An Efficient Approach to Single-Query Path Planning," in *Proc. of the 2000 IEEE International Conference on Robotics & Automation*, 2000, pp. 995–1001.

[11] S. Quinlan, "Efficient Distance Computation between Non-Convex Objects," in *Proc. of the 1994 IEEE International Conference on Robotics & Automation*, 1994, pp. 3324–3329.

[12] A. Escande, S. Miossec, and A. Kheddar, "Continuous gradient proximity distance for humanoids free-collision optimized-postures," in *Proc. of the IEEE-RAS International Conference on Humanoid Robots*, 2007, pp. 188–195.

[13] O. Kanoun, F. Lamiraux, and P.-B. Wieber, "Kinematic Control of Redundant Manipulators: Generalizing the Task Priority Framework," *IEEE Transactions on Robotics*, vol. 27, no. 4, pp. 785–792, 2011.

[14] B. Faverjon and P. Tournassoud, "A Local Based Approach for Path Planning of Manipulators With a High Number of Degrees of Freedom," in *Proc. of IEEE International Conference on Robotics and Automation*, 1987, pp. 1152–1159.