Workshop on Robot Motion Planning:
Online, Reactive, and in Real-time
2012 IEEE/RSJ International Conference on
Intelligent Robots and Systems, IROS 2012
Vilamoura, Algarve, Portugal, October 7-12, 2012

Reactive Humanoid Motion Planning for Reaching Tasks

Eiichi Yoshida and Fumio Kanehiro.

***Abstract*— This paper addresses a reactive motion planning framework that allows a humanoid robot, supposedly teleoperated, to perform reaching tasks in complex environments with uncertainty like a damaged plant using measured information like voxel map or point clouds. Since sufficient reactiveness is required for smooth human operation, we have developed an efficient computation method to update the environment information and to plan or replan a feasible whole-body reaching path within a second when necessary. This highly responsive planning scheme benefits from rapid computation of whole-body stable posture using approximated center of gravity and analytical inverse kinematics, combined with effective representation of 3D environment using sphere tree that can be rapidly updated when environmental changes occur. We validate the proposed method in a cluttered plant environment with moving object.**

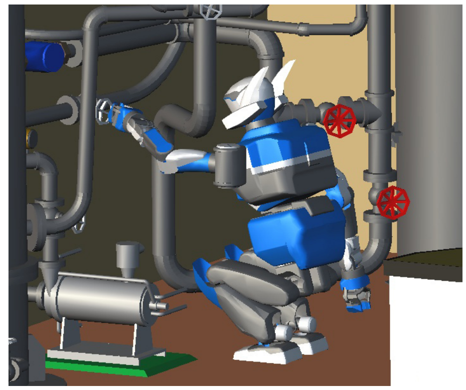


Fig. 1. Example of a reaching task in complex plant environment

I. Introduction

Probabilistic sampling-based motion planning methods have recently made great progress in its efficiency and gained strong attention in many application areas. a collision-free path that connects the initial and goal configurations is computed using a roadmap composed of nodes and edges that represent admissible configurations and local paths respectively. Two roadmap building mechanisms are identified as mainstream of sampling-based method: diffusion (e.g. Rapidly-exploring random tree, RRT) and sampling (e.g. Probabilistic RoadMap, PRM) [1], [2].

One of the most challenging applications for motion planning is the humanoid robot, which is currently expected to work to replace humans in difficult situations than ever. In hazardous environments like a damaged nuclear plant including unknown obstacles, a teleoperated humanoid seems to be a reasonable solution than fully autonomous one. In this case, the operator may want to give commands with certain abstraction level like "reach that point" or "rotate that valve", to perform such a motion shown in Fig. 1, from a mobile or tablet interface. The robot should have minimum autonomy that can interpret and execute those commands to execute its motion. We can here benefit from the capacity of the sampling planning approach that can handle many degrees of freedom (DOFs) efficiently.

Although many general planning algorithms have been already proposed in the literature for humanoid motion planning, 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.

For this purpose, we have recently developed an efficient motion planner [3] that can generate humanoid whole-body motion quickly in complex environments such as plants with many pipes, using approximated inverse kinematics computation guaranteeing stability and bounding volumes with sphere trees that can model measure point clouds or voxel map. As this is one-shot planning that assumes complex but fixed environments, in this research we present reactive motion planning method in changing environments.

In our previous study [4] we have proposed a reactive motion method that combines the replanning and deformation methods. Once a collision-free path is planned and starts being executed, the robot keeps executing the path as long as the path remains feasible with necessary local path deformation according to the motion of obstacles. If the executed path becomes infeasible even after deformation, the replanning is activated to find an alternative path through queries on the updated roadmap. We have validated the effectiveness of this method by applying it to redundant manipulators. However, further improvement was necessary so that the method can work with humanoid robots as the computation was still heavy.

The efficient motion planning method we have proposed [3] meets the requirements of fast computation to establish reactive planning framework for a humanoid robot to move in complex environment using measured information.

Eiichi Yoshida and Fumio Kanehiro are with CNRS-AIST JRL (Joint Robotics Laboratory), UMI3218/CRT, and F. Kanehiro is also with Humanoid Research Group, both belonging to Intelligent Systems Research Institute, National Institute of Advanced Industrial Science and Technology (AIST), 1-1-1 Umezono, Tsukuba, Ibaraki 305-8568 Japan (e.yoshida@aist.go.jp)

II. REPLANNING FRAMEWORK

A. Parallel Planning and Execution

In this section we first present briefly our reactive planning framework composed of parallel execution and planning [5] that satisfies the following specifications.

- In the event of environmental changes, if collisions are anticipated on the planned path being executed, the planner starts replanning immediately. As soon as another collision-free path is obtained again, it is executed.
- During the path replanning, the robot continues its motion unless it approaches obstacles within the specified safety distance. The path is executed in such a way that it decelerates when approaching the obstacle and makes a complete stop at a safety distance.
- There may be a case where the anticipated collision on the path is removed during the replanning. In this case, the replanning is canceled and the robot continues executing the original path without stopping.

Figure 2 illustrates how the reactive planner works with different states. It has the feature of having two "threads", Execution and Planning, running simultaneously. In Fig. 2, the texts in box correspond to the "states" of the planner. State transition occurs when a "signal" is received by the thread solid arrows in Fig. 2) or when the internal status of the thread changes (dotted arrows in Fig. 2). The signals and internal status changes causing state transition are indicated by italic and underlined texts in Fig. 2.

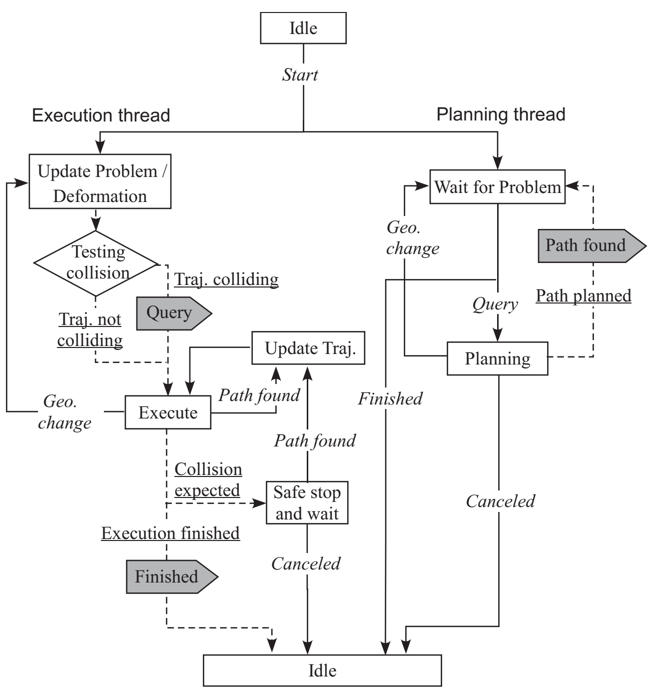


Fig. 2. State transition diagram of replanning. The replanning is performed by parallel threads of Execution and Planning that exchange signals. The states are shown in the boxes and the signal emissions are indicated by shadowed boxes. The italic and underlined texts depict the signals and internal status changes that bring about state transitions respectively.

We assume that the environmental changes can be detected by sensing mechanisms in an appropriate manner, to send "Geo. change" signals to the threads as soon as those changes are observed.

If the Execution thread at "Execute" state receives during path execution, it verifies if replanning is necessary. If it is the case, then Execution thread sends "Query" signal to the Planning thread to start the replanning. If the replanning is successful, the Execution thread updates the path and executes the replanned path without stopping. If the robot goes within safety distance to the obstacle before the planner finds a collision-free path, the Execution thread makes the robot stop by decelerating and wait for the planner to return another collision-free path. The planning fails if a feasible path is not found within the specified time. Of course there are limitations in the speed of the moving obstacles that can be avoided with respect to the robot capacity [5].

The proposed framework is implemented on RT (Robot Technology) Middleware which has been proposed as software platform [6] as a software unit called an RT Component. RT Middleware encourages the modularization and the reuse of software in the robotic field. Other software modules communicating with the motion planner, like the robot controllers and sensor systems to detect the environmental changes, can also be implemented as RT Components.

B. Roadmap Reuse

The replanning is performed based on an incrementally updated roadmap to benefit from the knowledge acquired during the previous exploration of the environment. Two kinds of roadmap, working and learning roadmaps are utilized for this replanning. As shown in Fig. 3 the learning roadmap stores the information about the environment over the whole planning time, whereas the working roadmap is continuously updated so that it includes only the valid part involved in the current replanning problem [5]. As a result, the working roadmap remains compact but reflects the most recent changes in the environment.

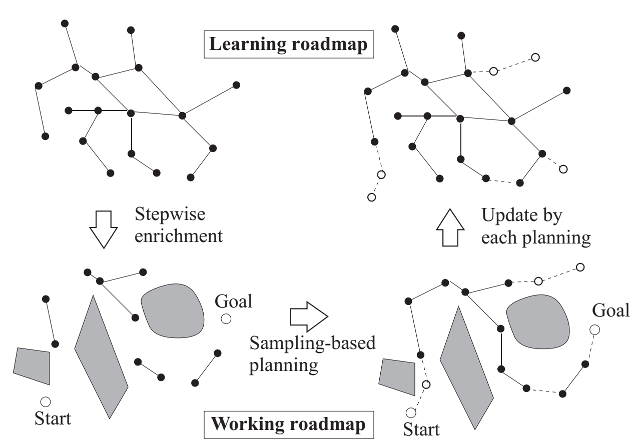


Fig. 3. Learning and working roadmaps.

III. EFFICIENT WHOLE-BODY MOTION PLANNING

A brief overview of efficient whole-body humanoid motion planning introduced in [3] is provided in this section. 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. We adopt a collision model using sphere trees and a projection that satisfies stability and kinematic constraints by maintaining approximated center of gravity (COG) position and computing arm and leg configurations with analytical inverse kinematics.

A. Collision models using sphere tree

We here 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. Since assigning a small cube to each voxel is memory-consuming, we represent the environment by a sphere tree [7]. A sphere is assigned to each voxel with the diameter equivalent to voxel resolution. A sphere tree is composed of many spheres and is used to detect collisions during the planning 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 using aligned bounding box (AABB) by dividing the groups into single sphere. The robot shape is also approximated to detect collisions quickly and make it easy to compute distances. The robot shape is approximated by spheres and capped cylinders since it is easy to compute distances.

B. Projection satisfying stability and kinematic constraints

The configuration projection unfortunately tends to be computationally heavy because a humanoid robot must respect many constraints while reaching such as feet position/orientation and 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.

The reaching task can be naturally defined by a goal position $\boldsymbol{p}_e = (x_e, y_e, z_e)^T$ and orientation $\boldsymbol{rpy}_e = (\phi_e, \theta_e, \psi_e)^T$ of the end effector. To compute the corresponding robot posture by projection, we define a configuration as follows.

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

This is concatenation of the end-effector position and orientation $\boldsymbol{p}_e$, $\boldsymbol{rpy}_e$, the height of the trunk z_t and an orientation of the trunk base $\boldsymbol{rpy}_t = (\phi_t, \theta_t, \psi_t)^T$. A sampled $\boldsymbol{q}_{goal}$ is projected so that it does not violate the stability and kinematic constraints, assuming that:

1) the whole mass concentrates on a point fixed to the trunk link at $\boldsymbol{COG}_{approx}$ as shown in Fig. 4.

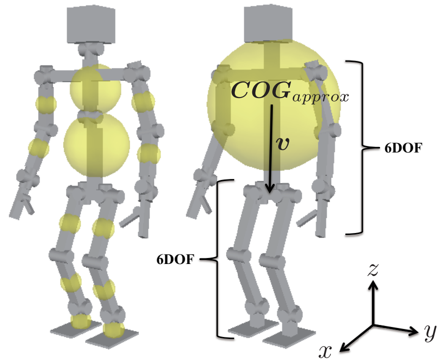


Fig. 4. The original kinematic chain(left) and the simplified kinematic chain used to find goal postures(right). Some of joints are fixed and the original kinematic chain is split into four 6DOF chains connected through the trunk. Distributing masses are assumed to be concentrating on the trunk.

2) the arms and legs are composed of six DOFs. This is the case of our humanoid robot, HRP-2 [8].

First, based on this assumption 1, we can determine the trunk horizontal position easily so that $\boldsymbol{COG}_{approx}$ does not move. This can be done by computing the trunk base position $\boldsymbol{p}_t$ from $\boldsymbol{COG}_{approx}$ based on a fixed relative vector $\boldsymbol{v}$ from $\boldsymbol{COG}_{approx}$ to the origin of the trunk link, its sampled orientation $\boldsymbol{rpy}_t$ and height z_t.

Then from this trunk base position, angles of the arms are computed by solving analytical solutions of inverse kinematics using the trunk position and orientation $\boldsymbol{p}_t$, $\boldsymbol{rpy}_t$ and end-effector position and orientation $\boldsymbol{p}_e$, $\boldsymbol{rpy}_e$. The joint angles of legs are calculated to keep the feet positions in the same way. We have verified that the error of approximation of COG is within 2[cm] in most of the cases [3] and those errors are compensated during the execution time.

C. Simulation of reaching motion

A reaching motion is planned using RRT-Connect [9]. The initial configuration and goals obtained by the projection are used as goals 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:

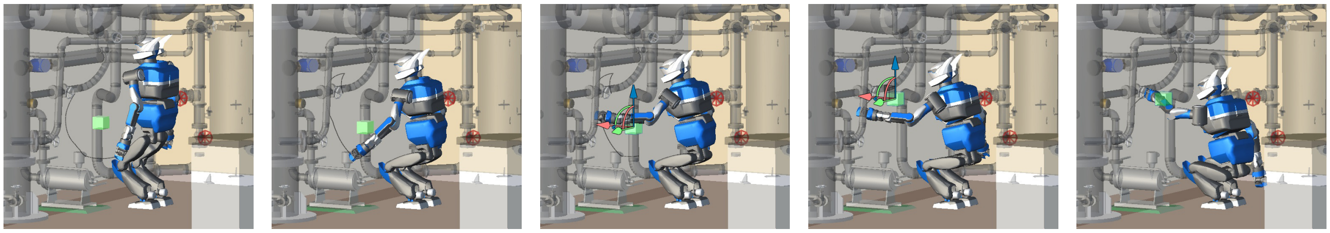
$$\boldsymbol{q}_{plan} = [\boldsymbol{q}_{arm}^T \ z_t \ \boldsymbol{rpy}_t^T] \quad (2)$$

where $\boldsymbol{q}_{arm}$ is an array of joint angles of an arm used to reach. While RRT-Connect grows a tree, the horizontal position of the trunk base is determined in the same way the projection described above to keep the robot balance. Leg joint angles are computed by solving analytical solution of inverse kinematics as well.

IV. REACTIVE REACHING MOTIONS

The efficient planning method introduced the previous section has been integrated into the reactive planning framework presented in Section II, using a motion planning software KineoWorks^{TM} [10].

We employed the plant environment shown in Fig. 1 as an example of complex environments. In addition to the



(a) (b) (c) (d) (e)

Fig. 5. Reaching motion replanned to avoid a moving obstacle in a complex environment

one-shot motion planning [3], reactive path replanning is performed in case of there are (possibly unknown) moving obstacles that are also measured as voxels or point clouds represented as sphere tree here.

As we assume all this point information comes from sensors, we actually do not have to distinguish static and moving obstacles, but we just need to update the newly measured region. The whole environment of 4m x 5m in Fig. 1 is represented by around 25,000 points with the resolution of 2 cm which are modeled as spheres of radius 1 cm. The average time required for sphere-tree model reconstruction was 28.3 ms on average with Intel processor Core i7 CPU with 2.70GHz. Although an optimal data management is preferable in case of partial changes, collision model updating is not a bottleneck in this scale of environment.

Figure 5 shows snapshots of replanning process to a valve in a plant environment, where the obstacles are displayed as transparent for better visibility. The green cube simulates an unknown or moving obstacle that is detected only when the robot gets closer to the goal. A collision-free path of reaching with the left hand is first planned as shown in Fig. 5a. When the obstacle moves downwards, new path is immediately replanned by avoiding outside (Fig. 5b, c). The obstacle finally comes upwards, which leads the replanned path to avoid underneath (Fig. 5d, e).

Figure 6 is the final configuration of the planned motion. We can observe that the left arm reaches the goal avoiding the static environment (the pipe) and moving obstacle.

The average planning time was 96 ms, including 10ms for average 2671 collision computation. With this example we can conclude that the proposed planning framework can provide a collision-free motion in a changing environment within a second for reaching tasks, which leads to comfortable teleoperation by human operator.

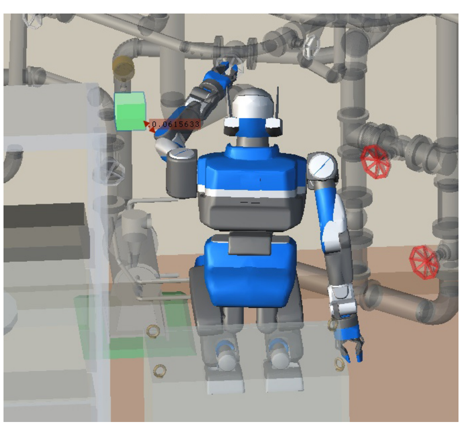


Fig. 6. Final configuration of the reaching motion

V. Conclusions

In this paper we presented an efficient reactive planning framework for a humanoid robot performing reaching tasks. We integrated an efficient environment modeling and whole-body stable configuration computation with approximated COG and analytical inverse kinematics into a reactive planning framework in changing environments. We could show that a replanning can be finished within a second even in a complex environment with moving obstacle. This provides a sufficient autonomy required for a humanoid robot that can accept high-level task commands from human operators.

In this paper we focused on the feasibility of reactive planning, and obviously path execution becomes the next important issue. By integrating the real-time execution also presented in [3], we will validate the proposed reactive planning with first realistic simulator then a physical robot.

Future work also includes extension of the proposed method to a variety of motions other than reaching, including walking or more complex tasks like object manipulation or repairing.

References

[1] H. Choset *et al.*, *Principles of Robot Motion: Theory, Algorithms, and Implementation*. MIT Press, 2006.

[2] S. LaValle, *Planning Algorithm*. Cambridge University Press, 2006.

[3] F. Kanehiro, E. Yoshida, and K. Yokoi, "Efficient reaching motion planning and execution for exploration by humanoid robots," in *Proc. 2012 IEEE/RSJ Int. Conf. on Intelligent Robots and Systems*, 2012, in press.

[4] E. Yoshida and F. Kanehiro, "Reactive robot motion using path replanning and deformation," in *Proc. 2011 IEEE Int. Conf. on Robotics and Automation*, 2011, 5457–5462.

[5] E. Yoshida, K. Yokoi, and P. Gergondet, "Online replanning for reactive robot motion: Practical aspects," in *Proc. 2010 IEEE/RSJ Int. Conf. on Intelligent Robots and Systems*, 2010, 5927–5933.

[6] N. Ando *et al.*, "RT-middleware: Distributed component middleware for RT (robot technology)," in *Proc. 2005 IEEE/RSJ Int. Conf. On Intelligent Robots And Systems (IROS2005)*, 2005, 3555–3560.

[7] S. Quinlan, "Efficient distance computation between non-convex objects," in *Proc. 1994 IEEE Int. Conf. on Robotics and Automation*, 1994, 3324–3329.

[8] K. Kaneko *et al.*, "The humanoid robot HRP-2," in *Proc. 2004 IEEE Int. Conf. on Robotics and Automation*, 2004, 1083–1090.

[9] J. Kuffner and S. LaValle, "RRT-connect: An efficient approach to single-query path planning," in *Proc. 2004 IEEE Int. Conf. on Robotics and Automation*, 2004, 995–1001.

[10] J.-P. Laumond, "Kineo CAM: a success story of motion planning algorithms," *IEEE Robotics & Automation Magazine*, vol. 13, no. 2, 90–93, 2006.