Real-Time Self Collision Avoidance for Humanoids by means of Nullspace Criteria and Task Intervals

Hisashi Sugiura, Michael Gienger, Herbert Janssen, Christian Goerick
Honda Research Institute Europe GmbH
Carl-Legien Strasse 30
D-63073 Offenbach/Main, Germany
Email: hisashi.sugiura@honda-ri.de

***Abstract*— We describe a new method for real-time collision avoidance for humanoid robots. Instead of explicitly modifying the commands, our method influences the control system by means of a nullspace criteria and a task interval. The nullspace criteria is driven by a virtual force acting on a joint center vector that defines the minimum of a potential function in joint space. The task interval defines the target constraints in task coordinates and allows the avoidance system to specify deviations from the given target position. The advantages of this indirect method are that smooth trajectories can be achieved and the underlying motion control may use any trajectory generation method that is able to satisfy the constraints given by the collision avoidance. It is most useful for highly redundant robots like typical humanoids. The method is able to assure smooth collision free movement on the humanoid robot ASIMO in real time interaction even in cases where the dynamical constraints of legged walking apply.**

I. Introduction

For humanoid robots recently requirements such as autonomy, interactivity and robustness become more and more important. In the real world and especially in interaction with humans, movements targets cannot be predicted in advance and planning methods generally become less attractive. Nevertheless, it is very important to detect and avoid self-collision and obstacles both predictively and interactively. In order not to break the robot, some traditional systems freeze the robot in a so called "*emergency stop*" which means a discontinuous stop of all movements. But if dynamical constraints apply like e.g. in case of legged robots that have to keep dynamical balance, both the robot and its environment including interacting humans may be in danger.

The purpose of the work presented in this paper is real-time self-collision avoidance between the body and both arms of our humanoid robot ASIMO, i.e. allow any command to be given interactively without breaking the robot by collisions of its link segments and smooth robot's motions unless the target or at least a collision free posture close to the target is reached.

There have been many papers presented about collision avoidance. Collision avoidance methods can be roughly distinguished into two categories. The first type of methods uses planning Popular methods are for instance Potential field methods [7] and Rapidly-exploring Random Trees (RRTs) [8]. However these methods take a lot of computation time because they use configuration space [9]. The second type of methods are reactive ones which usually operate in task space [1], [6], [10]. This type of method is attractive because it requires less computation time and is less dependent on the overall complexity of the scene. One of the most important points for this method is how to deal with target reaching motions and avoidance motions. Hanafusa and Nakamura et al. [5], [11] proposed the task priority method. Seto et al. [12] applied an instantaneous inverse kinematics solution [13] for collision avoidance. The collision avoidance method we propose here is in the latter category since we want to deal with highly dynamic environments and for safety reasons use only onboard computation resources. We use task interval in order to switch the priority.

II. Distance computation

For the collision avoidance method we need to compute the closest points and thus the distance between the *segments* of the robot which are the physical links separated by joints. All segments are modeled by *SSLs* (sphere swept line) or spheres since computation based on the real shape of the segments is computationaly too expensive within our real-time and hardware constraints [2], [14]. The segment model of the robot is shown in Fig. 1. The closest points and distances are computed between all possible segment pairs.

Since the computational complexity of pairwise distance computation is $O(N^2)$ the robot's collision model has two layers. One is a coarse model consisting of one primitive per segment and the other is a fine model which can use up to 10 primitives in case of the upper body. All segment pairs are first computed based on the coarse model. The fine model is only used when a pair of coarse model primitives is close enough to each other *(warning zone)*.

III. Collision avoidance

We define the requirements for collision avoidance as follows.

- The segments should not come closer to each other than a certain minimum distance *(red zone)*, otherwise the robot will freeze and all (arm and upper body) motion will be stopped. This is an emergency situation and should never happen during normal operation.

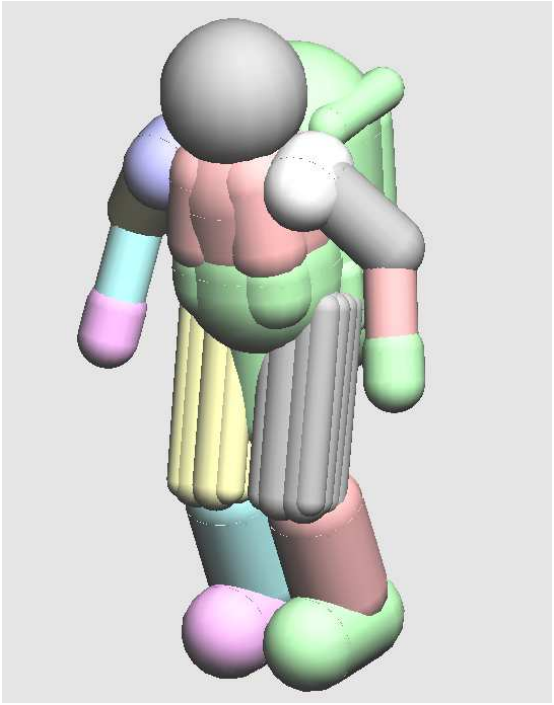


Fig. 1. Asimo's collision model is composed of sphere-swept-lines and spheres. Differing colors were used for visibility.

- If segments come closer to each other than another specific distance *(yellow zone)* they should be "pushed away" from each other.
- The motion should not be terminated unless the target or at least a position close to the target is reached.
- A target should always be reached exactly if at all possible, i.e. if the target is outside the yellow zone.

A. Whole Body Motion Control

To control a robot with redundant degrees of freedom, we use a whole body motion control system with nullspace optimization criteria. This means that the task space target is always tracked while the nullspace criteria is used to derive a unique solution for the redundant kinematics.

In this control method, each joint velocity is computed as

$$\dot{\mathbf{q}} = \mathbf{J}^{\#}\dot{\mathbf{x}}_{task} + \mathbf{N}\boldsymbol{\xi} \tag{1}$$

$$\mathbf{N} = \mathbf{I} - \mathbf{J}^{\#}\mathbf{J} \quad . \tag{2}$$

The matrix $\mathbf{N}$ maps an arbitarary joint velocity vector $\boldsymbol{\xi}$ into the nullspace, $\dot{\mathbf{q}}$ is the joint velocity vector, $\mathbf{J}^{\#}$ the pseudo inverse Jacobian, $\dot{\mathbf{x}}_{task}$ the task space velocity and $\mathbf{I}$ the identity matrix.

The nullspace optimization criteria can be expressed as a vector cost function $H(\mathbf{q})$. Its gradient is mapped into the null space.

$$\dot{\mathbf{q}} = \mathbf{J}^{\#}\dot{\mathbf{x}}_{task} - \mathbf{N}(\frac{\partial H(\mathbf{q})}{\partial \mathbf{q}})^T \tag{3}$$

In our case we chose the cost function to penalize deviations from an arbitrary joint center vector $\tilde{\mathbf{q}}$. A joint limit avoidance cost function is used.

$$H(\mathbf{q}) = \frac{1}{2}(\mathbf{q} - \tilde{\mathbf{q}})^T\mathbf{W}(\mathbf{q} - \tilde{\mathbf{q}}) \tag{4}$$

The matrix $\mathbf{W}$ is a weighting matrix.

In the easiest case by choosing $\tilde{\mathbf{q}}$ accordingly, this cost function allows to keep simply away from the joint limits [3]. Additionally by using $\tilde{\mathbf{q}}$ as an input parameter to our control system we can easily shift the null space potential to favor any externally given posture while simultaneously obeying the task space target velocity.

B. Virtual Force

To compute the joint center vector we use a virtual force which is projected on the joint center $\tilde{\mathbf{q}}$. It generates a force $\mathbf{f}$ proportional to the distance a joint moves from its center.

In our collision avoidance method we call $\mathbf{p}$ the smallest of all segment pair closest point vectors. If the length of this vector $\mathbf{p}$ is smaller than the yellow zone distance d_{yz} a non-zero $\Delta\mathbf{x}$ is the result.

$$\Delta\mathbf{x} = \begin{cases} 0 & \text{if} \quad |\mathbf{p}| > d_{yz} \\ (d_{yz}\frac{1}{|\mathbf{p}|} - 1)\mathbf{p} & \text{else} \end{cases} \tag{5}$$

The virtual force $\mathbf{f}_{virtual}$ is given as

$$\mathbf{f}_{virtual} = k\Delta\mathbf{x} \tag{6}$$

where k is the spring constant.

By means of the Jacobian transform and the virtual work principle we get

$$\boldsymbol{\tau} = {\mathbf{J}_\mathbf{p}}^T\mathbf{f} \tag{7}$$

where $\boldsymbol{\tau}$ is the joint torque vector and ${\mathbf{J}_\mathbf{p}}^T$ is identified as the transpose of the Jacobian of $\mathbf{p}$. This equation relates forces in cartesian space and torques in joint space.

The torque $\boldsymbol{\tau}_{virtual}$ is given by

$$\boldsymbol{\tau}_{virtual} = \mathbf{C}\Delta\mathbf{q} \tag{8}$$

where $\mathbf{C}$ is a compliance matrix for the virtual springs of the joints and $\Delta\mathbf{q}$ is used to describe the deviation between the actual joint vector and the joint center vector $\tilde{\mathbf{q}}$.

Then we get following equation for $\Delta\mathbf{q}$:

$$\Delta\mathbf{q} = \mathbf{C}^{-1}{\mathbf{J}_\mathbf{p}}^T k\Delta\mathbf{x} \quad . \tag{9}$$

The Jacobian is used for position control by the distance computation. e.g. control of the hand segment position with respect to the body.

This equation converts deviations of positions to deviations of joints without singularities since it doesn't use a pseudo inverse Jacobian but the transpose Jacobian. Thus the method is able to change the posture of the robot by applying a virtual force in task space and transforming it to a movement in joint space.

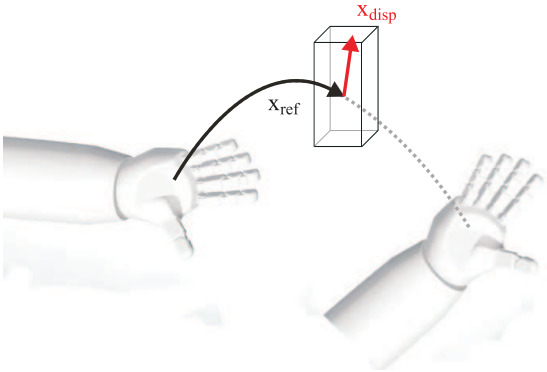


Fig. 2. Virtual displacement cuboid

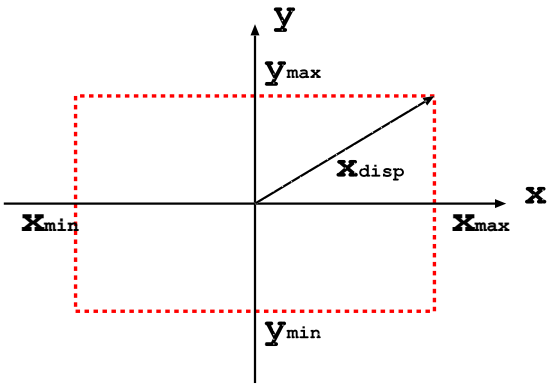


Fig. 3. Task interval in 2-dimensional case

C. Task Interval

For many robot tasks, movement targets do not necessarily need to be specific cartesian points and orientations, e.g. for dancing or making gestures. Thus for some tasks, it is useful to specify the robot's movement targets as volumes in space, orientation intervals or generally speaking task intervals [4]. In Fig. 2, a task interval for the hand position is depicted. The cuboid can be conceived as a virtual box around the reference point, in which the effector is allowed to move.

If the effector reference point is within the task interval, a cost function gradient is mapped into the task space. An arbitarary cost fuction can be used. Fig. 3 illustrates the 2-dimensional case. When the displacement is outside of the task interval, it is clipped to the boundary of the interval. The displacement is computed by a cost function gradient and the pseudo inverse Jacobian as follows,

$$\begin{aligned} \delta\mathbf{x}_{disp} &= \alpha\frac{\partial H(\mathbf{q})}{\partial \mathbf{x}} \\ &= \alpha\frac{\partial H(\mathbf{q})}{\partial \mathbf{q}}\,\frac{\partial \mathbf{q}}{\partial \mathbf{x}} \\ &= \alpha\nabla\mathbf{H}(\mathbf{q})^T\,\mathbf{J}^{\#} \end{aligned} \tag{10}$$

where α is a convergence step width.

We choose the joint limit cost function. Therefore the displacement is computed by the joint center by means of the equation (4), (9) and (10). The size of the task interval is determined by the displacement. Accordingly the joint limit avoidance is always mapped into task space. e.g. $\mathbf{x_{max}}$ and $\mathbf{y_{max}}$ are determined by the displacement $\mathbf{x}_{disp}$in Fig.3. The actual target position is changed from $\mathbf{x}_{task}$ to $\mathbf{x}_{task}+\mathbf{x}_{disp}$ by task interval. As far as the respective segment of the robot stays within the task interval, the robot follows the joint center which is determined by the virtual force.

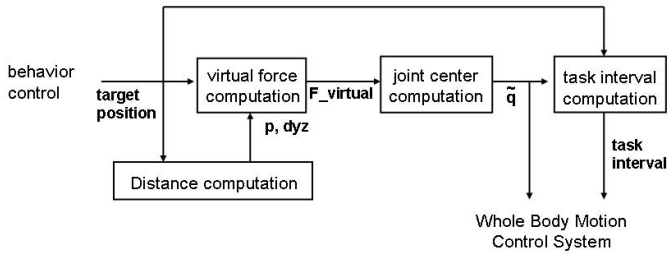


Fig. 4. Architecture of the collision avoidance and its interaction with the underlying whole body motion control system.

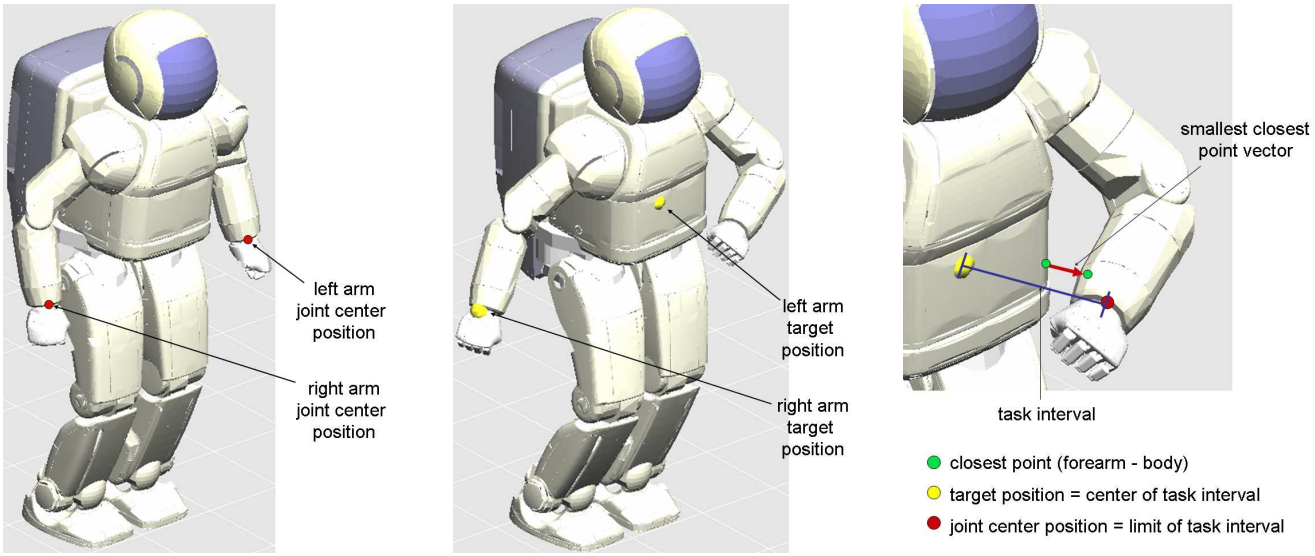
The advantage of a collision avoidance using these methods is that the target command is not modified explicitly, but it indirectly influences the control system by means of the joint center and the task interval.

Fig. 5 exemplifies the system on Asimo. Fig. 5(a) shows the robot in a resting position which is determined exclusively by the joint center since there is no target and thus no task space. In Fig. 5(b) targets are assigned to both arms. While the right arm has reached its target, the left arm did not because an arm segment violates the yellow zone and the collision avoidance is activated. Fig. 5(c) is a closeup of Fig. 5(b). The forearm violates the yellow zone and a virtual force is generated based on the smallest closest point vector $\mathbf{p}$. This force pushes the arm back to the limit of the task interval. Note that the target position is not modified but the actual position stays within the task interval. If the target position moves out of the yellow zone, the task interval becomes zero and the arm moves to the target position accordingly.

IV. System

ASIMO is a humanoid robot which has 5 degrees of freedom in each arm, 2 degrees of freedom in the head and 6 degrees of freedom in each leg. Our whole body motion control system uses 21 degrees of freedom including virtual degrees of freedom that describe the robot's legs. The system of collision avoidance and its interaction with the control system is shown in Fig. 4. Distance computation and avoidance are computed on ASIMO's internal computers because they are fundamental safety functions. Arm trajectories are generated by low pass filtering the difference between the target and the actual position. The task interval output from the collision avoidance is also low pass filtered before it is used in the control system to avoid jumping task space conditions.

When a target is given, the robot moves to the target without walking. But if the target cannot be reached and the position



(a) Resting position (b) Collision avoidance active (c) Closeup

Fig. 5. Asimo and its control and avoidance system in some typical equilibrium states. (a) resting position, no targets given, only the default nullspace criteria apply. (b) the left arm target is inside the yellow zone, but virtual force and task interval ensure that the hand does not hit the upper leg. (c) closeup of (b), the blue line indicates the relevant part of the task interval

error exceeds a threshold , the robot starts to walk to the target. The threshold is 50mm in case of our experiments.

V. Results

We tested our method by using both kinematics-only and dynamical simulations of Asimo. We ran extensive tests with over 100 randomly generated arm targets in our dynamics simulator but found no cases of red zone violation (emergency stop) or targets that could not be reached.

In our experiments we set the zones as follows:

- The *warning zone* distance is set to 100mm; entering it switches the distance computation from coarse to fine models.
- The *yellow zone* distance is set to 50mm; entering it activates collision avoidance.
- The *red zone* distance is set to 3mm; entering it triggers the emergency stop.

We will discuss two typical cases of our experiments below. Fig. 6 shows an example of a left hand trajectory crossing the yellow zone. The walking is not activated in this example because the walking threshold is not exceeded. The start posture and the target posture do not violate the yellow zone, but between time t1 and t2, the arm limb violates the yellow zone as shown in Fig. 6(a). Therefore the virtual force depicted in the center of Fig. 6(b) is generated between t1 and t2. The task interval is depicted in the bottom of Fig. 6(b). It is also generated from t1 but after t2 it does not immediately become zero because of the filtering discussed above. At time t3, the closest point model pair is switched as shown in top of Fig. 6(b). At time t4, the actual position reaches the limit of the task interval as can be seen in the bottom of Fig. 6(a). From there on the arm moves directly to the target. The arm trajectory is modified by the collision avoidance. The arm moves to the final target even if it partially violates the yellow zone and finally reaches the target without stopping.

A second example - this time including walking - is given in Fig. 7.

The yellow spheres denote the given target positions. The image sequence shows the following steps:

(a) The target for the left arm is set. The target posture would violate the yellow zone.

(b) The robot tries to reach the target but fails, because the arm violates the yellow zone of the body. Thus the robot starts walking when the deviation threshold is reached.

(c) The robot continues to reach for the target. The yellow sphere close to the right hand is the target for the right arm.

(d) The robot stops walking and finally reaches the target.

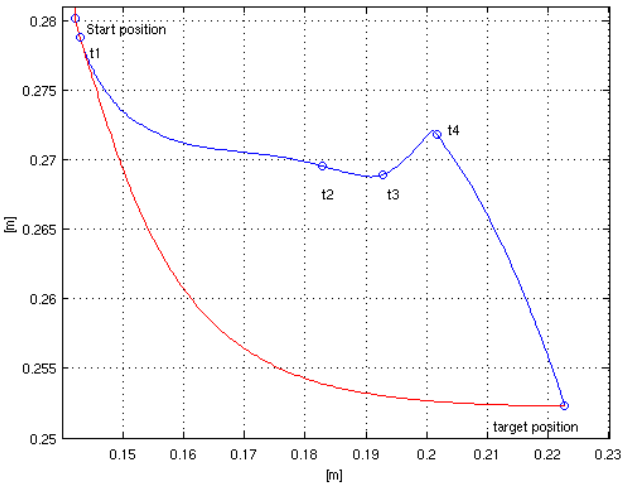
In this example the collision avoidance initally prohibits reaching the target but by means of causing a difference between the actual position and trajectory indirectly triggers walking so that the target can finally be reached.

VI. Conclusion and Outlook

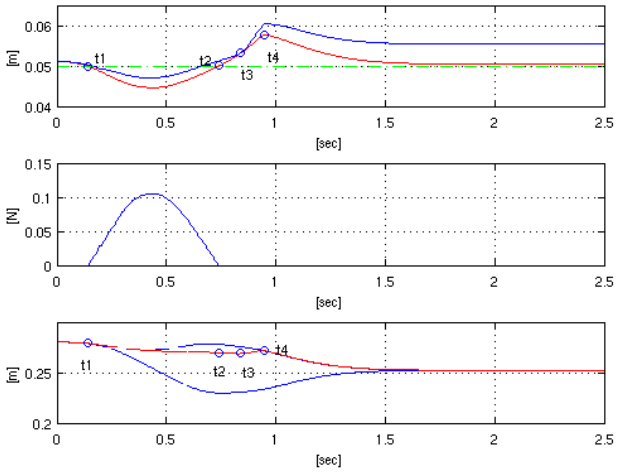
We described a collision avoidance method which uses virtual forces and task intervals. According to the results of our simulations, our robot moves smoothly without colliding when given various targets interactively.

We are currently working on extending the system to include external objects as detected by the vision system as obstacles.

Also we will adjust the robot segment model in real-time to increase the precision of the distance computation without increasing the computational load.

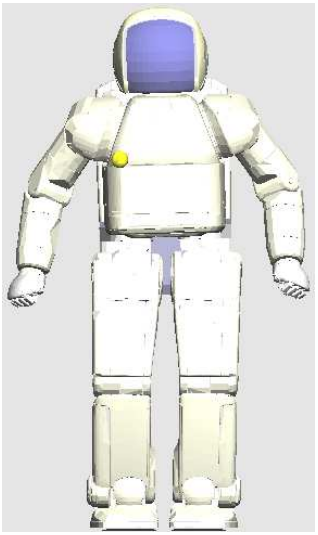


(a) Trajectory in XY-plane

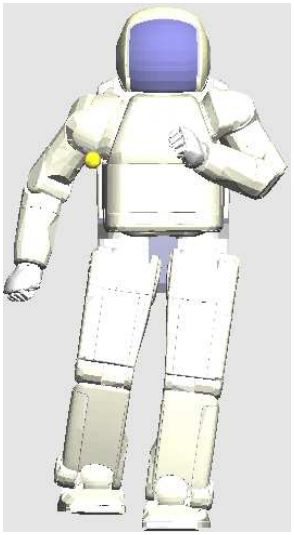


(b) Top: closest distance, Center: virtual force, Bottom: task interval

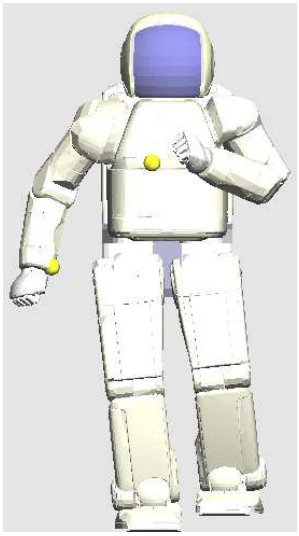
Fig. 6. Typical example of a trajectory crossing the yellow zone. t1 is the time of entering the yellow zone, t2 is the time of leaving the yellow zone, t3 is the time of switching between the closest point model pairs and t4 is time of the actual wrist position reaching the limits of the task interval. (a) XY-Plot of the left wrist position with (blue) and without (red) collision avoidance. (b) top: distance between closest points of the two closest model pairs hand - body1 (blue) and hand - body2 (red). (b) center: virtual force (b) bottom: actual Y position of the left wrist (red) and task interval borders (blue)



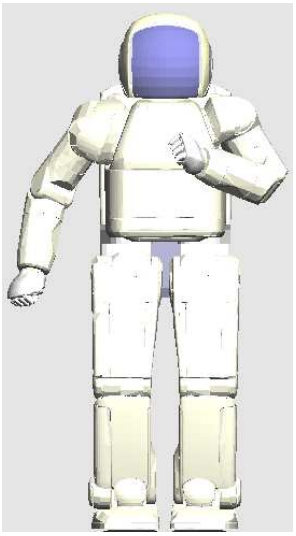
(a) Start: resting



(b) First step



(c) Second step



(d) Stop: target reached

Fig. 7. Simple example of a target that is inside the body and requires walking. The yellow spheres are the target positions.

REFERENCES

[1] Oliver Brock, Oussama Khatib, and Sriram Viji. Task consistent obstacle avoidance and motion behavior for mobile manipulation. In *proceedings of the IEEE International Conference of Robotics and Automation*, 2002.

[2] Christer Ericson. *Real-time collision detection*. The Morgan Kaufman Publishers, 2005.

[3] Michael Gienger, Herbert Janssen, and Christian Goerick. Task-oriented whole body motion for humanoid robots. In *proceedings of the IEEE-RAS International Conference on Humanoid Robots*, 2005.

[4] Michael Gienger, Herbert Janssen, and Christian Goerick. Exploiting task intervals for whole body robot control. In *proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2006.

[5] Hideo Hanafusa, Tsuneo Yoshikawa, and Yoshihiko Nakamura. Redundancy analysis of articulated robot arms and its utilization for task with priority. In *SICE*, volume 19, pages 421–426, 1983.

[6] Ioannis Iossifidis and Gregor Schoener. Autonomous reaching and obstacle avoidance with the anthropomorphic arm of a robotic assistant using the attractor dynamics approach. In *proceedings of the IEEE International Conference on Robotics and Automation*, 2004.

[7] Oussama Khatib. Real-time obstacle avoidance for manipulations and mobile robots. In *The international Journal of Robotics Research*, volume 5, pages 90–98, 1986.

[8] James Kuffner, Koichi Nishiwaki, Satoshi Kagami, Yasuo Kuniyoshi, Masayuki, Inaba, and Hirochika Inoue. Self-collision detection and prevention for humanoid robots. In *proceedings of the IEEE International Conference on Robotics and Automation*, 2002.

[9] Tomás Lozano-Pérez and Michael A. Wesly. An algorithm for planning collision-free paths among polyhedral obstacles. In *Communications of the ACM*, volume 22, pages 560–570, 1979.

[10] Anthony A. Maciejewski and Charles A. Klein. Obstacle avoidance for kinematically redundant manipulators in dynamically varying environments. In *The international Journal of Robotics Research*, volume 4, pages 109–117, 1985.

[11] Yoshihiko Nakamura. *Advanced Robotics: Redundancy and Optimization*. Addison Wesley Publishing Company, 1991.

[12] Fumi Seto, Kazuhiro Kosuge, and Yasuhisa Hirata. Self-collision avoidance motion control for human robot cooperation system using robe. In *proceedings of the IEEE/RSJ Internatiional Conference on Intelligent Robots and Systems*, 2005.

[13] Toshio Tsuji, Seiya Nakayama, Atsushi Araki, and Koji Ito. Instantaneous inverse kinematic solution for redundant manipulators based on virtual arms and its application to winding control. In *JSME International Journal*, volume 38, pages 87–93, 1995.

[14] Gino van den Bergen. *Collision Detection in Interactive 3D Environments*. The Morgan Kaufman Publishers, 2004.