To appear in the Proc. of Humanoids 2000: First RAS/IEEE Int'l Conf. on Humanoid Robots

Dynamically-stable Motion Planning for Humanoid Robots

James Kuffner, Jr., Satoshi Kagami, Masayuki Inaba, and Hirochika Inoue

Dept. of Mechano-Informatics, The University of Tokyo
7-3-1 Hongo, Bunkyo-ku, Tokyo 113-8656 JAPAN
{kuffner,kagami,inaba,inoue}@jsk.t.u-tokyo.ac.jp
http://www.jsk.t.u-tokyo.ac.jp/~kuffner/humanoid/

Abstract. We present an algorithm for computing stable collision-free motions for humanoid robots given full-body posture goals. The motion planner is part of a simulation environment under development for providing high-level software control for humanoid robots. Given a robot's internal model of the environment and a statically-stable desired posture, we use a randomized path planner to search the configuration space of the robot for a collision-free path. Balance constraints are imposed on incremental search motions in order to maintain the overall dynamic stability of the computed trajectories. The algorithm is presented along with preliminary results using an experimental implementation on a dynamic model of the H5 humanoid robot.

1 Introduction

Research involving humanoid robots has increased during recent years. Advances in computing hardware and software have enabled the implementation of sophisticated motion control strategies. In particular, dynamic simulation software [WO82, Bar89, Mir96] has assisted in the realization of dynamic walking in several humanoid robots [Hir97, YINT98, NII99].

As the technology and algorithms for real-time 3D vision and tactile sensing improve, humanoid robots will be able to perform tasks that involve complex interactions with the environment (e.g. grasping and manipulating objects). The enabling software for such tasks includes motion planning for obstacle avoidance, and integrating planning with visual and tactile sensing data.

To facilitate the deployment of such software, we are currently developing a graphical simulation environment for the H5 dynamic humanoid robot [KKII00]. The project builds upon a software framework that was originally developed for the high-level control of computer animated characters in 3D virtual environments[Kuf99]. The software automatically computes object grasping and manipulation trajectories through a combination of inverse kinematics and randomized holonomic path planning. Feasible kinematic trajectories can be computed at interactive rates for single-arm reaching and object manipulation tasks.

The algorithm described in this paper represents our first attempt at automatically generating collision-free dynamically-stable motions from full-body posture goals. Our approach is to adapt techniques from an existing, successful path planner [KL00] by transforming incremental motions used during the search by a filter function that maintains dynamic balance constraints [KKT^{+}00]. Provided the initial and goal configurations correspond to statically-stable body postures, the path returned by the planner can be transformed into a collision-free and dynamically-stable trajectory for the entire body.

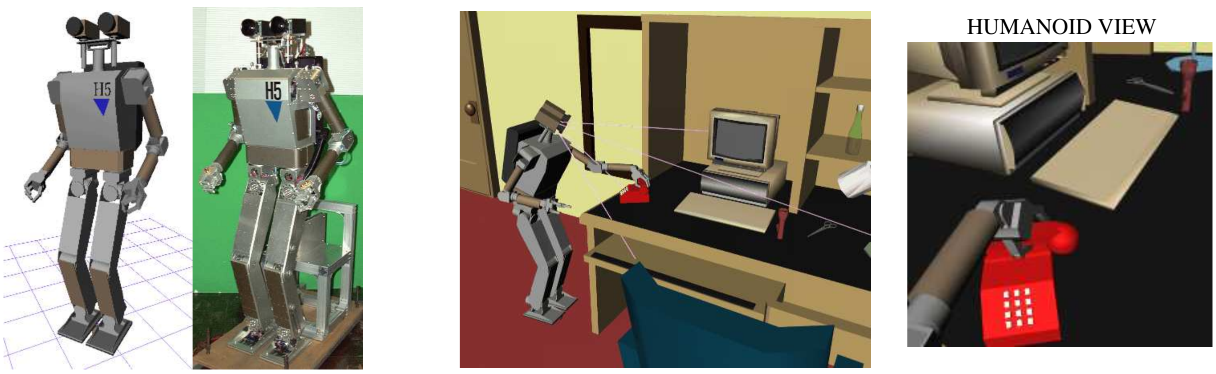

Fig. 1. The virtual and real dynamic humanoid 'H5' (left), and answering the phone in a simulated world (right).

Although the current implementation of the planner is limited to body posture goals, and a fixed position for either one or both feet, research is underway to extend the method to handle more complex body posture repositioning. It is our hope that through the use of such kinds of task-level planning algorithms and interactive simulation software, the current and future capabilities of humanoid and other complex robotic systems can be improved.

2 Background

Motion planning problems typically involve searching the system configuration space of one or more complicated geometric bodies for a collision-free path that connects a given start and goal configuration amidst environment obstacles. Complete algorithms exist for the general class of problems [SS83, Can88, HP00], but their computational complexity limits their use to low-dimensional configuration spaces.

This limitation, lower-bound hardness results [Rei79], and strong motivation to handle practical planning problems have stimulated the development and success of many path planning methods that use randomization (e.g., [BL90, Ove92, Sve93, CG93, KL93, HST94, KKKL94, HLM97, AW96, KŠLO96, BKL+97, MAB98, BOvdS99, KL00]). The accepted tradeoff is that the methods are incomplete, but will find a solution with any probability given sufficient running time. The goal is to develop randomized methods that converge quickly in practice, yet are simple enough to yield consistent behavior and analysis.

Due to the curse of dimensionality, developing practical motion planning algorithms for humanoid robots is a daunting task. Humanoid robots such as H5 (Figure 1) have 30 or more degrees of freedom. The problem is further complicated by the fact that humanoid robots must be controlled very carefully in order to maintain overall static and dynamic stability. These constraints severely restrict the set of allowable configurations and prohibit the direct application of existing randomized path planning techniques. Although efficient techniques have been developed for maintaining dynamic balance for biped robots [Rai86, VBSS90, PP99, KKT+00], none consider obstacle avoidance.

Recently, randomized motion planning algorithms that account for system dynamics have been developed for 2D and 3D rigid bodies with state spaces of up to twelve dimensions [LK99, LK00], for a 3D helicopter model [FDF99], and for a circular disc in 2D among circular moving obstacles [KHLR00]. These methods have yet to be applied to complex articulated models such as humanoid robots. It has been suggested to limit the active body degrees of freedom for humanoid robot path planning and balance control, though these ideas are still under development [Hir00].

Our approach is to adapt a variation of the randomized planner described in [KL00] to compute full-body motions for humanoid robots that are both dynamically-stable and collision-free. This planner (RRT-Connect) and its variants utilize Rapidly-exploring Random Trees (RRTs) [LaV98] combined with a simple greedy heuristic that aggressively tries to connect two search trees, one from the initial configuration and the other from the goal. These methods have been shown to be efficient in practice and converge towards a uniform exploration of the search space.

3 Robot Model and Assumptions

We have based our experiments on an approximate model of the H5 dynamic humanoid robot, including the kinematics and dynamic properties of the links (see Figure 1). The model of the link dynamics was successfully used to generate dynamically-stable stepping motions [NII99], and for online balance compensation [KKT+00]. Along with the existence of the dynamic model of the humanoid robot, we make the following assumptions:

1. We assume that the robot has access to a 3D model of the surrounding environment to be used for collision checking. For implementation on a real robot platform, an approximate model can be acquired using stereo vision or other means. In this case, the model need not be exact, provided that a conservative model of sensing error is taken into account by the planner to avoid potential collisions with obstacles.

2. The robot is currently balanced in a statically-stable posture supported by either one or both feet.

3. The location of the supporting foot or feet (in the case of dual-leg support) does not change during the planned motion.

4. The robot is given a full-body goal posture that is statically-stable. The goal posture may be set explicitly by a human operator, or computed via inverse kinematics or other means (e.g. for reaching a limb towards a target location or object).

4 Problem Formulation

To make things more precise, we now give a more formal formulation of the stable-posture motion planning problem for humanoid robots. The notation adopted here is loosely based on the conventions used in [Lat91], which are often used in the robotics and motion planning literature.

1. The robot is called $\mathcal{A}$.

2. The 3D environment (workspace) in which the robot moves is denoted by $\mathcal{W}$, and is modeled as the Euclidean space $\Re^3$ ($\Re$ is the set of real numbers).

3. $\mathcal{A}$ is a collection of p links $\mathcal{L}_i$ $(i = 1, \ldots, p)$ organized in a kinematic hierarchy with Cartesian frames $\mathcal{F}_i$ attached to each link. We denote the position of the center of mass c_i of link $\mathcal{L}_i$ relative to $\mathcal{F}_i$.

4. A *configuration* or *pose* of the robot is denoted by the set $\mathcal{P} = \{T_1, T_2, \ldots, T_p\}$, of p relative transformations for each of the links $\mathcal{L}_i$ as defined by the frame $\mathcal{F}_i$ relative to its parent link's frame. The *base* or *root* link transformation T_1 is defined relative to some world Cartesian frame $\mathcal{F}_{world}$.

5. Let n denote the number of generalized coordinates or *degrees of freedom* (DOFs) of $\mathcal{A}$. Note that n is in general *not equal* to p.

6. A *configuration* is denoted by $q \in \mathcal{C}$, a vector of n real numbers specifying values for each of the generalized coordinates of $\mathcal{A}$.

7. Let $\mathcal{C}$ be the *configuration space* or $\mathcal{C}$-space of $\mathcal{A}$. $\mathcal{C}$ is a space of dimension n.

8. Let $\text{FORWARD}(q)$ be a *forward kinematics* function mapping values of q to a particular pose $\mathcal{P}$. $\text{FORWARD}(q)$ can be used to compute the global transformation G_i of a given link frame $\mathcal{F}_i$ relative to the world frame $\mathcal{F}_{world}$.

9. The set of obstacles in the environment $\mathcal{W}$ is denoted by $\mathcal{B}$, where $\mathcal{B}_k$ $(k = 1, 2, \ldots)$ represents an individual obstacle.

10. We define the *$\mathcal{C}$-obstacle region* $\mathcal{C}B \subset \mathcal{C}$ as the set of all configurations $q \in \mathcal{C}$ where one or more of the links of $\mathcal{A}$ intersect (are in collision) with another link of $\mathcal{A}$, any of the obstacles $\mathcal{B}_k$. We also regard configurations $q \in \mathcal{C}$ where one or more *joint limits* are violated as part of the $\mathcal{C}$-obstacle region $\mathcal{C}B$.

11. The open subset $\mathcal{C} \setminus \mathcal{C}B$ is denoted by $\mathcal{C}_{free}$ and its closure by $cl(\mathcal{C}_{free})$, and it represents the *space of collision-free configurations* in $\mathcal{C}$ of the robot $\mathcal{A}$.

12. Let $\mathcal{X}(q)$ be a vector relative to $\mathcal{F}_{world}$ representing the global position of the center of mass of $\mathcal{A}$ while in the configuration q.

13. A configuration q is *statically-stable* if the projection of $\mathcal{X}(q)$ along the gravity vector g lies within the area of support $\mathcal{S}P$ (i.e. the convex hull of all points of contact between $\mathcal{A}$ and the support surface in $\mathcal{W}$).

14. Let $\mathcal{C}_{stable} \subset \mathcal{C}$ be the subset of *statically-stable configurations* of $\mathcal{A}$.

15. Let $\mathcal{C}_{valid} = \mathcal{C}_{stable} \cap \mathcal{C}_{free}$ denote the subset of configurations that are both *collision-free* and *statically-stable* postures of the robot $\mathcal{A}$. $\mathcal{C}_{valid}$ is called the set of *valid* configurations.

16. Let $\tau : \mathcal{I} \mapsto \mathcal{C}$ where $\mathcal{I}$ is an interval $[t_0, t_1]$, denote a motion trajectory or *path* for $\mathcal{A}$ expressed as a function of time. $\tau(t)$ represents the configuration q of $\mathcal{A}$ at time t, where $t \in \mathcal{I}$.

17. A trajectory τ is said to be *collision-free* if $\tau(t) \in \mathcal{C}_{free}$ for all $t \in \mathcal{I}$.

18. A trajectory τ is said to be both *collision-free* and *statically-stable* if $\tau(t) \in \mathcal{C}_{valid}$ for all $t \in \mathcal{I}$.

Given $q_{init} \in \mathcal{C}_{valid}$ and $q_{init} \in \mathcal{C}_{valid}$, we wish to compute a continuous motion trajectory τ such that $\forall t \in [t_0, t_1]$, $\tau(t) \in \mathcal{C}_{valid}$, and $\tau(t_0) = q_{init}$ and $\tau(t_1) = q_{goal}$. We refer to such a trajectory as a *statically-stable* trajectory.

Any statically-stable trajectory can be transformed into a dynamically-stable trajectory by arbitrarily slowing down the motion. For these experiments, we utilize the online balance compensation scheme described in

[KKT+00] as a method of generating a final dynamically-stable trajectory.The details of this technique are not described in this paper. Other methods of generating dynamically-stable trajectories from a given input motion are also potentially possible to apply here [Hir97, YINT98, YN00].

4.1 Planning Query

Note that in general, if a dynamically-stable solution trajectory exists for a given path planning query, there will be many such solution trajectories. Let Φ denote the set of all dynamically-stable solution trajectories for a given problem. A *planning query* is given as follows:

$$Planner(\mathcal{A}, \mathcal{B}, q_{init}, q_{goal}) \quad \longrightarrow \quad \tau \tag{1}$$

Given a model of the robot $\mathcal{A}$, obstacles in the environment $\mathcal{B}$, and an initial and goal posture, the planner returns a solution trajectory $\tau \in \Phi$. Currently, we require the planning software to compute only one solution. If the planner fails to find a solution, τ will be empty (a null trajectory).

5 Path Search

Unfortunately, there are no currently known methods for explicitly representing $\mathcal{C}_{valid}$. The obstacles are modeled completely in $\mathcal{W}$, thus an explicit representation of $\mathcal{C}_{free}$ is also not available. However, using a collision detection algorithm, a given $q \in \mathcal{C}$ can be tested to determine whether $q \in \mathcal{C}_{free}$. Testing whether $q \in \mathcal{C}_{stable}$ can also be checked by computing $\mathcal{X}(q)$ and verifying that its projection along g is contained within the boundary of $\mathcal{SP}$.

5.1 Distance Metric

As with the most planning algorithms in high-dimensions, a metric ρ is defined on $\mathcal{C}$. The function $\rho(q, r)$ returns some measure of the distance between the pair of configurations q and r). Some axes in $\mathcal{C}$ may be weighted relative to each other, but the general idea is to measure the “closeness” of pairs of configurations with a scalar function.

For the H5 humanoid, we use a metric that assigns higher relative weights to the generalized coordinates of links with greater mass and proximity to the trunk (torso):

$$\rho(q, r) = \sum_{i=1}^{n} w_i |q_i - r_i| \tag{2}$$

This choice of metric function attempts to heuristically encode a general relative measure of how much the variation of an individual joint parameter affects the overall body posture. Additional experimentation is needed in order to evaluate the efficacy of the many different metric functions possible.

5.2 Modified RRT-Connect

The Rapidly-exploring Random Tree (RRT) was introduced in [LaV98] as an efficient data structure and sampling scheme to quickly search high-dimensional spaces that have both algebraic constraints (arising from obstacles) and differential constraints (arising from nonholonomy and dynamics). The key idea is to bias the exploration toward unexplored portions of the space (an objective also shared by the planners in [HLM97] and [MAB98]).

In [LK99] an RRT-based approach to path planning was presented that generated and connected two RRTs in a state space, which generalizes $\mathcal{C}$. A holonomic variant of this planner that adds a greedy heuristic to guide searches that can be conducted in $\mathcal{C}$ was presented in [KL00]. For implementation details and analysis of these algorithms, the reader is referred to the original papers or a summary in [LK00].

The planner described in this paper is similar to the method in [KL00] in that it performs its search in $\mathcal{C}$. However, it uses a balance compensation method to enforce dynamic constraints imposed upon the ZMP (zero moment point) trajectory [KKT+00].

In particular, we modify the planner variant that employs symmetric calls to the *EXTEND* function as follows:

1. The *NEW_CONFIG* function in the *EXTEND* operation first generates a target configuration q_{target} by making an incremental step motion towards q from q_{near} as before. However, q_{new} is generated by filtering the straight-line path connecting q_{near} and q_{target} through the dynamic balance compensator. This creates an incremental dynamically-stable trajectory from q_{near} towards q_{target}. The filter simulation terminates when the

configuration converges or after a preset time limit is exceeded. If no collision occurs prior to termination, the most recent configuration output by the filter becomes q_{new} and is added to the tree $\mathcal{T}$. In this way, we are *guaranteed* that $q_{new} \in \mathcal{C}_{valid}$.

2. Rather than picking a purely random configuration $q_{rand} \in \mathcal{C}$ at every planning iteration, we pick a random configuration that also happens to correspond to a statically-stable posture of the robot (i.e. $q_{rand} \in \mathcal{C}_{stable}$).

A diagram depicting the modified *EXTEND* operation is given in Figure 2. Pseudocode for the complete modified *RRT_CONNECT_STABLE* algorithm is given in Figure 3. The main planning loop involves performing a simple iteration in which each step attempts to extend the RRT by adding a new vertex that is biased by a randomly-selected stable configuration.

EXTEND selects the nearest vertex already in the RRT to the given sample configuration, q. The function *NEW_CONFIG* makes a dynamically-stable motion toward q as outlined previously using some fixed incremental distance ϵ to generate q_{target}. Three situations can occur: *Reached*, in which q is directly added to the RRT, *Advanced*, in which a new vertex $q_{new} \neq q$ is added to the RRT; *Trapped*, in which no new vertex is added due to the inability of the balance compensator to generate an incremental trajectory that lies in $\mathcal{C}_{valid}$.

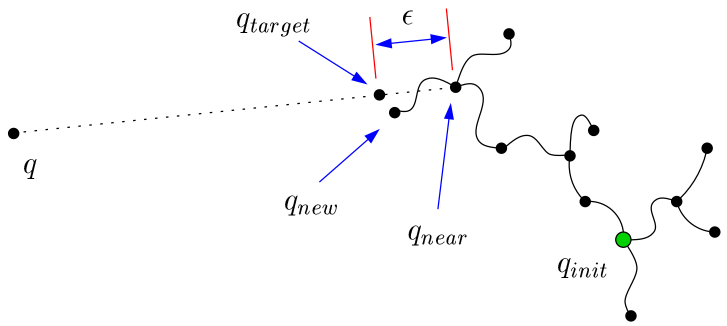

Fig. 2. The modified EXTEND operation.

EXTEND($\mathcal{T}, q$)
$q_{near} \leftarrow$ NEAREST_NEIGHBOR($q, \mathcal{T}$);
if NEW_CONFIG(q, q_{near}, q_{new}) **then**
    $\mathcal{T}$.add_vertex(q_{new});
    $\mathcal{T}$.add_edge(q_{near}, q_{new});
    if $q_{new} = q$ **then**
        Return *Reached*;
    else
        Return *Advanced*;
Return *Trapped*;

RRT_CONNECT_STABLE(q_{init}, q_{goal})
$\mathcal{T}_a$.init(q_{init}); $\mathcal{T}_b$.init(q_{goal});
for $k = 1$ **to** K **do**
    $q_{rand} \leftarrow$ RANDOM_STABLE_CONFIG();
    if not (EXTEND($\mathcal{T}_a, q_{rand}$) =*Trapped*) **then**
        if (EXTEND($\mathcal{T}_b, q_{new}$) =*Reached*) **then**
            Return PATH($\mathcal{T}_a, \mathcal{T}_b$);
    SWAP($\mathcal{T}_a, \mathcal{T}_b$);
Return *Failure*

Fig. 3. The modified RRT-Connect algorithm for generating dynamically-stable motions.

One of the key differences between *RRT_CONNECT_STABLE* and the classic RRT-Connect is that instead of uniformly sampling $\mathcal{C}$ and growing trees that lie entirely in $\mathcal{C}_{free}$, it attempts to uniformly sample $\mathcal{C}_{stable}$ and grow trees that lie within $\mathcal{C}_{valid}$.

5.3 Convergence and Completeness

Although not proven here, it is expected that arguments similar to those given in [KL00, LK00] could potentially be constructed to show uniform coverage and convergence over $\mathcal{C}_{valid}$, provided that appropriate convergence and performance guarantees can be derived for the balance compensation filter.

Ideally, we would like to build a *complete* planning algorithm. That is, the planner always returns a solution trajectory if one exists, and indicates failure if no solution exists. As mentioned in Section 2, implementing a practical complete planner is a daunting task for even low-dimensional configuration spaces (see [HP00]). Thus, we typically trade off completeness for practical performance by adopting heuristics (e.g. randomization).

The planning algorithm implemented here is incomplete in that it returns failure after a preset time limit is exceeded. Thus, if the planner returns failure, we cannot conclude whether or not a solution exists for the given planning query, only that our planner was unable to find one in the allotted time. Uniform coverage and convergence proofs, though only theoretical, at least help to provide some measure of confidence that when an algorithm fails to find a solution, it is likely that no solution exists. This is an area of ongoing research.

6 Random Statically-stable Postures

For our algorithm to work, we require a method of generating random statically-stable postures (i.e. random point samples of $\mathcal{C}_{stable}$). Although it is trivial to generate random configurations in $\mathcal{C}$, it is not so easy to generate them in $\mathcal{C}_{stable}$, since it encompasses a much smaller subset of the configuration space.

In our current implementation, a set $\mathcal{Q}_{stable} \subset \mathcal{C}_{stable}$ of N samples of $\mathcal{C}_{stable}$ is generated as a preprocessing step. This computation is specific to a particular robot and support-leg configuration, and need only be performed once. The collection of stable postures is saved to a file and loaded into memory when the planner is initialized[1].

A sample series of dual-leg and single-leg stable postures for the H5 humanoid robot are shown in Figure 4 (perspective view), Figure 5 (front view), and Figure 6 (left view).

However, we currently restrict the set of samples $q \in \mathcal{Q}_{stable}$ to belong to a unique dual-leg support posture at a fixed relative foot location. We hope to ultimately incorporate single-leg postures, multiple relative positions for dual-leg postures, and identify appropriate transitions between them in future implementations. For the rest of this discussion, we will assume that $\mathcal{Q}_{stable}$ contains only dual-leg support postures at a fixed relative position for the feet; specifically, statically-stable body configurations supported by both feet planted parallel and shoulder-width apart.

Populating $\mathcal{Q}_{stable}$ is very similar to the problem of sampling the configuration space of a constrained closed-chain system (e.g. closed-chain manipulator robots or molecular conformations [LYK99, HA00]). We employ similar techniques here.

The set $\mathcal{Q}_{stable}$ is populated with these fixed-position dual-leg support postures as follows:

1. The configuration space of the robot $\mathcal{C}$ is sampled by generating a random body configuration $q_{rand} \in \mathcal{C}$. q_{rand} can include either random or fixed positions for the arm and head joints. Although fixed positions for these limbs slightly decreases the full generality of the planner by reducing the set of possible full-body motions to be used for obstacle avoidance, the tradeoff in planning efficency can be worthwhile. Since the arms and head of the H5 robot have a relatively small mass, their joint variables are currently fixed at a canonical rest position in our current implementation.

2. Holding the right leg fixed at its random configuration, inverse kinematics is used to attempt to position the other foot at the required relative position to generate the body configuration q_{right}. If it succeeds, then q_{right} is tested for membership in $\mathcal{C}_{valid}$ (i.e. static stability and no self-collision).

3. An identical procedure is performed to generate q_{left} by holding the left leg fixed at its random configuration derived from q_{rand}, using inverse kinematics to position the right leg, and testing for membership in $\mathcal{C}_{valid}$.

[1] Since the H5 humanoid has 30 DOF, storing a 4-byte float for each joint variable corresponds to roughly 1.2MB of storage per $N = 10,000$ sample configurations. However, this memory usage can be significantly reduced by adopting fixed-point representations for the joint variables. This has not implemented in our current planner.

4. If either $q_{right} \in \mathcal{C}_{valid}$ or $q_{left} \in \mathcal{C}_{valid}$, we employ the following trick to effectively double the samples discovered: since most humanoid robots have left-right symmetry, additional stable postures can be derived by mirroring the generated stable configurations.

Although stable configurations could be generated "on-the-fly" at the same time the planner performs the search, pre-calculating $\mathcal{Q}_{stable}$ is preferred for efficiency. After pre-generating the set $\mathcal{Q}_{stable}$, the results are saved to a file for instant retrieval the next time the planner is initialized. In addition, multiple stable-configuration set files can be saved independently. If the planner fails to find a path after a certain number of samples have been removed from the currently active $\mathcal{Q}_{stable}$ set, a new one can be loaded with different samples.

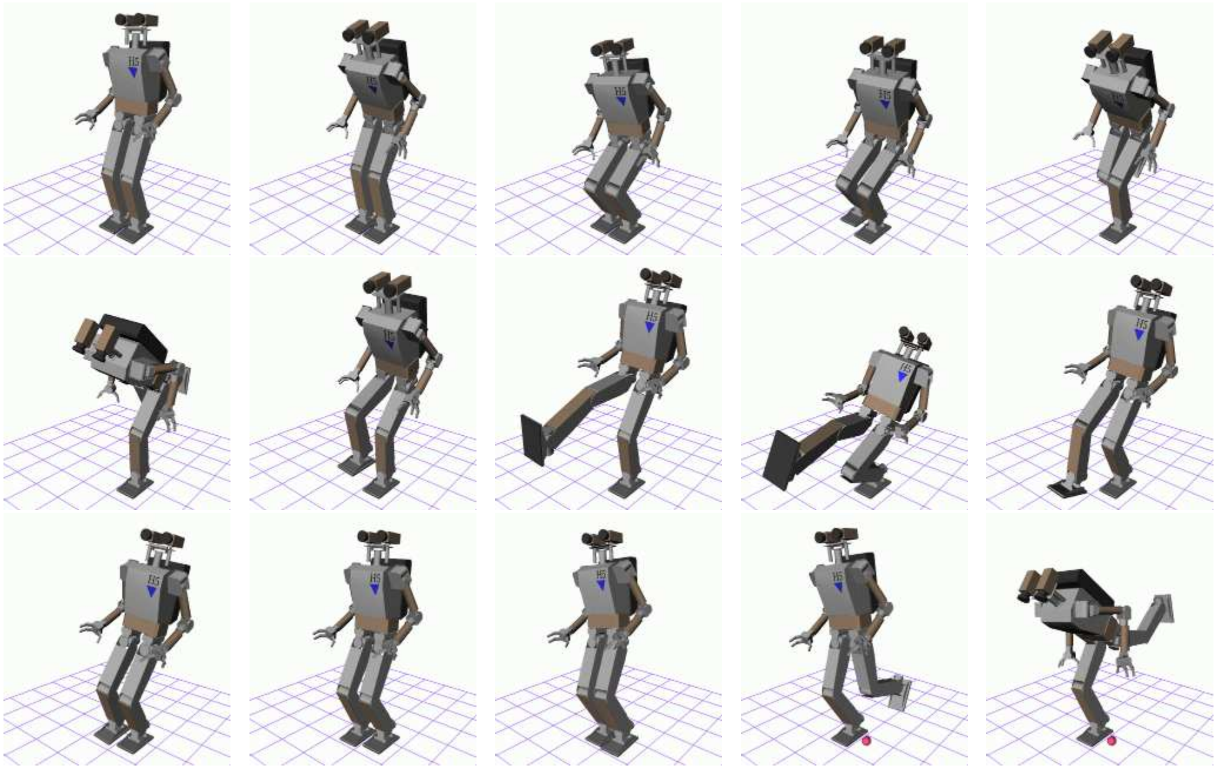

Fig. 4. Dual-leg and single-leg stable postures for the H5 humanoid robot (perspective view).

7 Experiments

This section presents some preliminary experiments performed on a 270 MHz SGI O2 (R12000) workstation. We have implemented a prototype planner that runs within a graphical simulation environment. An operator can position individual joints or use inverse kinematics to specify body postures for the virtual robot. The filter function can be run interactively to ensure that the goal configuration is statically-stable. After specifying the goal, the planner is invoked to attempt to compute a dynamically-stable trajectory connecting the goal configuration to the robot's initial configuration (assumed to be a stable posture).

This section presents some preliminary experiments performed on a 270 MHz SGI O2 (R12000) workstation. We have implemented a prototype planner that runs within a graphical simulation environment using a dynamic model of the H5 humanoid robot (30-DOF). Through a graphical user interface, an operator can position individual joints or use inverse kinematics to specify body postures. The filter function can be run interactively to ensure that the goal configuration is statically-stable.

After specifying the goal, the planner is invoked to attempt to compute a dynamically-stable trajectory connecting the goal configuration to the robot's initial configuration (assumed to be a stable posture).

Figure 7 shows a computed dynamically-stable motion for the H5 robot moving from a neutral standing position to a low crouching posture.

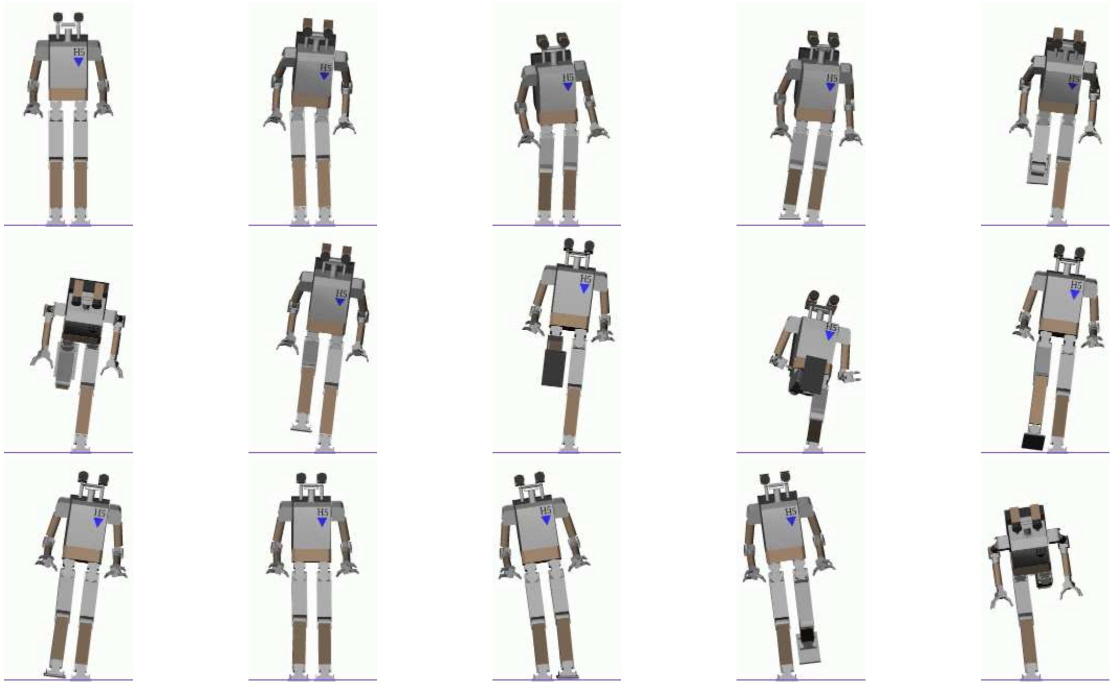

Fig. 5. Dual-leg and single-leg stable postures for the H5 humanoid robot (front view).

Fig. 6. Dual-leg and single-leg stable postures for the H5 humanoid robot (left view).

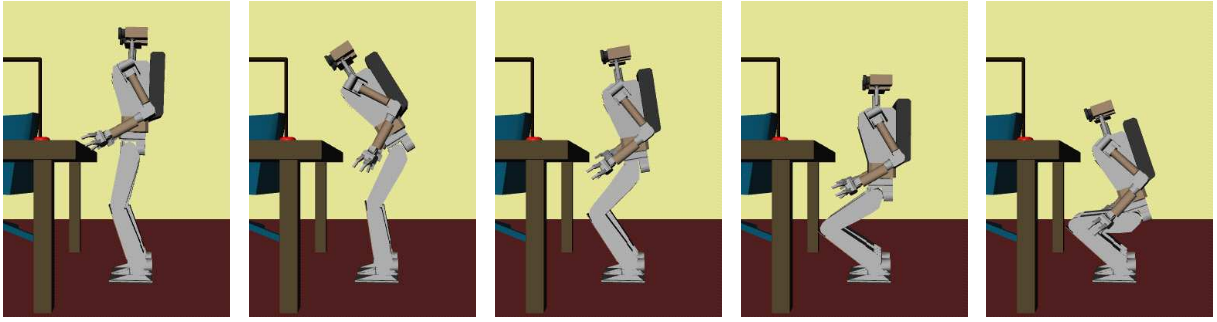

Fig. 7. Dynamically-stable planned trajectory for a crouching motion.

This scene contains over 11,300 triangle primitives. The 3D collision checking software used for these experiments was the RAPID library based on OBB-Trees developed by the University of North Carolina[GLM96]. Since we used a non-incremental 3D collision checking algorithm, performance could potentially be improved significantly by using an alternate algorithm (for example [LC91, Mir97, GHZ99]).

The total wall time elapsed in solving these queries ranges from approximately 3 to 12 minutes. A summary of the computation times for repeated runs of 100 trials is shown in Table 1.

Task Description	Computation Time (seconds)			
	min	max	avg	stdev
Crouch near table	176	620	304	133

Table 1. Performance statistics ($N = 100$ trials).

8 Discussion

This paper presents an algorithm for computing dynamically-stable collision-free motions for humanoid robots given full-body posture goals. Balance constraints are imposed upon incremental search motions computed by a randomized planner in order to maintain the overall dynamic stability of the computed trajectories.

There are many potential uses for such software, with the primary one being a high-level control interface for automatically solving complex tasks for humanoid robots that involve simultaneous obstacle-avoidance and overall dynamic stability.

By using a graphical simulation environment, sophisticated motion generation algorithms can be efficiently developed and debugged, reducing the costs and safety risks involved in testing software for humanoid robots.

We are aware of several current limitations in our current algorithm and implementation that forms the basis for future research:

1. The locations of the body supports (the foot positions) are currently fixed.
2. Efficient nearest-neighbor techniques [AMN+98, IM98] can be used to reduce computation time for n sample points from the obvious $O(n)$ algorithm to near-logarithmic time.
3. Incremental collision-detection software can be used to improve the collision-checking performance (e.g. [LC91, Mir97, GHZ99]).
4. Additional examples need to be tested and analyzed to obtain a better profile of the planner and potential bottlenecks.
5. Analysis of the coverage of $\mathcal{C}_{valid}$ and rates of convergence needs to be investigated, including a clearer understanding of how the filter function may or may not affect convergence.

6. The effectiveness of different configuration space distance metrics needs to be investigated.

7. We currently have no method for integrating visual or tactile feedback.

Acknowledgments

Many thanks to Fumio Kanehiro for his help in debugging and integrating the AutoBalancer software library. We also thank Koichi Nishiwaki for helping with the H5 model, and Yukiharu Tamiya, one of the original authors of the AutoBalancer software. We are grateful to Steven LaValle, David Hsu, Lydia Kavraki, Nancy Amato and Jean-Claude Latombe, with whom many discussions regarding path planning helped form a foundation for this work. This research is supported in part by a Japan Society for the Promotion of Science Postdoctoral Fellowship for Foreign Scholars in Science and Engineering. Many thanks to the Stanford University Dept. of Computer Science, where the initial development of portions of the simulation software was conducted.

References

[AMN+98] S. Arya, D. M. Mount, N. S. Netanyahu, R. Silverman, and A. Y. Wu. An optimal algorithm for approximate nearest neighbor searching. *Journal of the ACM*, 45:891–923, 1998.

[AW96] N. Amato and Y. Wu. A randomized roadmap method for path and manipuation planning. In *Proc. of IEEE Int. Conf. Robotics and Automation*, pages 113–120, Minneapolis, MN, 1996.

[Bar89] D. Baraff. Analytical methods for dynamic simulation of non-penetrating rigid bodies. In *Proc. SIGGRAPH '89*, pages 223–231, 1989.

[BKL+97] J. Barraquand, L.E. Kavraki, J.C. Latombe, T.Y. Li, R. Motwani, and P. Raghavan. A random sampling scheme for path planning. *Int. J. Robot. Res.*, 16(6):759–774, 1997.

[BL90] J. Barraquand and J.-C. Latombe. Robot motion planning: A distributed representation approach. *Int. J. Robot. Res.*, 10(6):628–649, December 1990.

[BOvdS99] V. Boor, M. Overmars, and A.F. van der Stappen. The gaussian sampling strategy for probabilistic roadmap planners. In *Proc. of IEEE Int. Conf. Robotics and Automation*, Detroit, MI, 1999.

[Can88] J.F. Canny. *The Complexity of Robot Motion Planning*. MIT Press, Cambridge, MA, 1988.

[CG93] D. Chalou and M. Gini. Parallel robot motion planning. In *Proc. of IEEE Int. Conf. Robotics and Automation*, pages 24–51, Atlanta, GA, 1993.

[FDF99] E. Frazzoli, M.A. Dahleh, and E. Feron. Robust hybrid control for autonomous vehicles motion planning. Technical report, Laboratory for Information and Decision Systems, Massachusetts Institute of Technology, Cambridge, MA, 1999. Technical report LIDS-P-2468.

[GHZ99] L. J. Guibas, D. Hsu, and L. Zhang. H-Walk: Hierarchical distance computation for moving convex bodies. In *Proc. ACM Symposium on Computational Geometry*, pages 265–273, 1999.

[GLM96] S. Gottschalk, M. C. Lin, and D. Manocha. Obbtree: A hierarchical structure for rapid interference detection. In *SIGGRAPH '96 Proc.*, 1996.

[HA00] Li Han and Nancy M. Amato. A kinematics-based probabilistic roadmap method for closed chain systems. In *In Proc. of Workshop on Algorithmic Foundations of Robotics (WAFR'00)*. A. K. Peters, March 2000.

[Hir97] Kazuo Hirai. Current and future perspective of honda humanoid robot. In *In Proc. of 1997 IEEE/RSJ Int. conf. on Intelligent Robots and Systems (IROS'97)*, pages 500–508, 1997.

[Hir00] H. Hirukawa. Motion planning algorithm of a arm of a humanoid robot, 2000. *Personal communication*.

[HLM97] D. Hsu, J.-C. Latombe, and R. Motwani. Path planning in expansive configuration spaces. *Int. J. of Computational Geometry and Applications*, 9(4-5):495–512, 1997.

[HP00] H. Hirukawa and Y. Papegay. Motion planning of objects in contact by the silhouette algorithm. In *IEEE Int. Conf. Robot. & Autom.*, pages 722–729, April 2000.

[HST94] T. Horsch, F. Schwarz, and H. Tolle. Motion planning for many degrees of freedom : Random reflections at c-space obstacles. In *Proc. of the IEEE Int. Conf. on Robotics and Automation (ICRA'94)*, pages 3318–3323, San Diego, CA, April 1994.

[IM98] P. Indyk and R. Motwani. Approximate nearest neighbors: Towards removing the curse of dimensionality. In *Proceedings of the 30th Annual ACM Symposium on Theory of Computing*, 1998.

[KHLR00] R. Kindel, D. Hsu, J.C. Latombe, and S. Rock. Kinodynamic motion planning amidst moving obstacles. In *IEEE Int. Conf. Robot. & Autom.*, April 2000.

[KKII00] J.J. Kuffner, S. Kagami, M. Inaba, and H. Inoue. Simulating high-level robot behaviors. In *In Proc. of RSJ 5th Annual Robotics Symposium of Japan*, March 2000.

[KKKL94] Y. Koga, K. Kondo, J. Kuffner, and J.-C. Latombe. Planning motions with intentions. In *Proc. SIGGRAPH '94*, pages 395–408, 1994.

[KKT+00] S. Kagami, F. Kanehiro, Y. Tamiya, M. Inaba, and H. Inoue. Autobalancer: An online dynamic balance compensation scheme for humanoid robots. In *Robotics: The Algorithmic Perspective, Workshop on Algorithmic Foundations of Robotics*. A K Peters, Hanover, NH, March 2000.

[KL93] L.E. Kavraki and J.-C. Latombe. Randomized preprocessing of configuration space for fast path planning. Technical report, Dept. of Computer Science, Stanford University, September 1993.

[KL00] J.J. Kuffner and S.M. LaValle. RRT-Connect: An efficient approach to single-query path planning. In *In Proc. IEEE Int'l Conf. on Robotics and Automation (ICRA'2000)*, San Francisco, CA, April 2000.

[KŠLO96] L. Kavraki, P. Švestka, J. C. Latombe, and M. H. Overmars. Probabilistic roadmaps for path planning in high-dimensional configuration space. *IEEE Trans. Robot. & Autom.*, 12(4):566–580, 1996.

[Kuf99] J.J. Kuffner Jr. *Autonomous Agents for Real-time Animation*. PhD thesis, Stanford University, 1999.

[Lat91] J. C. Latombe. *Robot Motion Planning*. Kluwer Academic Publishers, Boston, MA, 1991.

[LaV98] S. M. LaValle. Rapidly-exploring random trees: A new tool for path planning. TR 98-11, Computer Science Dept., Iowa State Univ. <`http://janowiec.cs.iastate.edu/papers/rrt.ps`>, Oct. 1998.

[LC91] M. C. Lin and J. F. Canny. Efficient algorithms for incremental distance computation. In *IEEE Int. Conf. Robot. & Autom.*, 1991.

[LK99] S.M. LaValle and J.J Kuffner. Randomized kinodynamic planning. In *Proc. IEEE Int'l Conf. on Robotics and Automation (ICRA'99)*, Detroit, MI, May 1999.

[LK00] S.M. LaValle and J.J Kuffner. Rapidly-exploring random trees: Progress and prospects. In *Proc. 2000 Workshop on the Algorithmic Foundations of Robotics.*, Hanover, NH, March 2000.

[LYK99] S.M. LaValle, J.H. Yakey, and L.E. Kavraki. A probabilistic roadmap approach for systems with closed kinematic chains. In *IEEE Int. Conf. Robot. & Autom.*, 1999.

[MAB98] E. Mazer, J. M. Ahuactzin, and P. Bessière. The Ariadne's clew algorithm. *J. Artificial Intell. Res.*, 9:295–316, November 1998.

[Mir96] B. Mirtich. *Impulse-Based Dynamic Simulation of Rigid Body Systems*. PhD thesis, University of California, Berkeley, CA, 1996.

[Mir97] B. Mirtich. V-Clip: Fast and robust polyhedral collision detection. Technical Report TR97-05, Mitsubishi Electronics Research Laboratory, 1997.

[NII99] K. Nagasaka, M. Inaba, and H. Inoue. Walking pattern generation for a humanoid robot based on optimal gradient method. In *In Proc. of 1999 IEEE Int. Conf. on Systems, Man, and Cybernetics*, 1999.

[Ove92] M. Overmars. A random approach to motion planning. Technical report, Dept. Computer Science, Utrect University, Utrect, The Netherlands, October 1992.

[PP99] J. Pratt and G. Pratt. Exploiting natural dynamics in the control of a 3d bipedal walking simulation. In *In Proc. of Int. Conf. on Climbing and Walking Robots (CLAWAR99)*, Portsmouth, UK, September 1999.

[Rai86] Marc Raibert. *Legged Robots that Balance*. MIT Press, Cambridge, MA, 1986.

[Rei79] J. H. Reif. Complexity of the mover's problem and generalizations. In *Proc. 20th IEEE Symp. on Foundations of Computer Science (FOCS)*, pages 421–427, 1979.

[SS83] J. T. Schwartz and M. Sharir. On the 'piano movers' problem: Ii. general techniques for computing topological properties of real algebraic manifolds. *Advances in applied Mathematics*, 4:298–351, 1983.

[Sve93] P. Svestka. A probabilistic approcach to motion planning for car-like robots. Technical report, Dept. Computer Science, Utrect Univ., Utrect, The Netherlands, April 1993.

[VBSS90] M. Vukobratovic, B. Borovac, D. Surla, and D. Stokie. *Biped Locomotion: Dynamics, Stability, Control, and Applications*. Springer-Verlag, Berlin, 1990.

[WO82] M.W. Walker and D.E. Orin. Efficient dynamic computer simulation of robotic mechanisms. *ASME J. of Dynamic Systems, Measurement, Control*, 104:205–211, 1982.

[YINT98] J. Yamaguchi, S. Inoue, D. Nishino, and A. Takanishi. Development of a bipedal humanoid robot having antagonistic driven joints and three dof trunk. In *In Proc of 1998 IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS'98)*, pages 96–101, 1998.

[YN00] K. Yamane and Y. Nakamura. Dynamics filter - concept and implementation of on-line motion generator for human figures. In *IEEE Int. Conf. Robot. & Autom.*, April 2000.