A Skill-Based Motion Planning Framework for Humanoids

Marcelo Kallmann, Yazhou Huang and Robert Backman
(In Proceedings of ICRA 2010)

***Abstract*— This paper presents a multi-skill motion planner which is able to sequentially synchronize parameterized motion skills in order to achieve humanoid motions exhibiting complex whole-body coordination. The proposed approach integrates sampling-based motion planning in continuous parametric spaces with discrete search over skill choices, selecting the search strategy according to the functional type of each skill being coordinated. As a result, the planner is able to sequence arbitrary motion skills (such as reaching, balance adjustment, stepping, etc) in order to achieve complex motions needed for solving humanoid reaching tasks in realistic environments. The proposed framework is applied to the HOAP-3 humanoid robot and several results are presented.**

I. Introduction

Despite several successes in the motion planning domain, achieving whole-body coordinated humanoid motions for solving manipulation tasks remains a challenge. The problem is in particular difficult because most of humanoid tasks require coordination of different types of motion skills, which have to be addressed in an integrated fashion. This paper proposes a generic framework for addressing this problem.

The proposed framework is applied to the particular problem of coordinating stepping and reaching for the HOAP-3 humanoid platform. The robot has a relatively small reachable space for the arms and therefore coordination with stepping is critical even for reaching simple targets. Figure 1 illustrates a situation where the humanoid is not able to reach the target with a simple arm motion because the target is outside reachable range, and therefore stepping and body adjustments are required in order to solve the task.

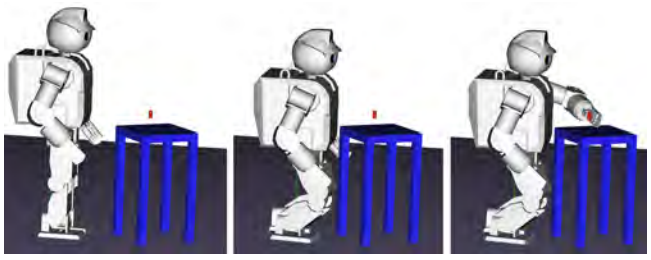

Fig. 1. Example of a simple reaching task which is only solvable after the robot performs a few stepping motions and body adjustments.

It is important to notice that the problem of skill coordination appears frequently in common tasks such as: relocating objects, opening doors, pushing buttons, etc. Most importantly this class of problems addresses a broad range of real-life tasks and represents a large proportion of people's daily activities. Deciding the sequence of skills to employ for such tasks is not trivial due the large number of possible coordinations and the different capabilities and constraints of each available skill. However solving the coordination of motion skills is critical for several reasons: 1) for achieving optimal movements exploring the full potential of the humanoid structure, 2) for enabling complex mobile manipulations in cluttered environments, and 3) for achieving human-like motions which are better suited for humanoid assistants interacting and collaborating with people.

Instead of relying on a series of pre-programmed controllers with reactive behaviors, the proposed framework is designed to explore and evaluate many possible motion strategies for each given problem. The planner is capable of exploring both the parameterization space of individual motion skills and the possible coordination points for switching between skills. The planner is based on a generic representation of motion skills which is able to encapsulate different algorithmic strategies for motion control in a common integrated framework. As a result the planner is able to evaluate several solutions for the sequencing of stepping and reaching motions in order to reach for a given target. The presented framework is inspired by cognitive theories advocating that humans select learned motor programs (here called motion skills) each time a task has to be solved [26].

II. Related Work

Traditional motion planning approaches [20, 21, 23] are based on the systematic search in configuration spaces. Among the several techniques, sampling-based methods such as the Probabilistic Roadmaps (PRMs) [14] and Rapidly-Exploring Random Trees (RRTs) [19, 22] have become extremely popular for planning in continuous configuration spaces. These and other methods have been applied to humanoid structures, however, as whole-body motion planning for humanoids is inherently a multi-modal problem, most of the approaches have been developed for particular modes or skills, for instance: footstep planning for precise locomotion around obstacles [3, 4, 18], reaching motions for manipulation [1, 5, 6, 11, 12, 17], etc.

When planning is limited to a discrete selection among possible actions or predefined motion patterns, discrete planners based on A* and its several variations [15, 16, 24] are well suited for finding the best sequence of actions to be taken. No single planning method is best in all cases. For instance,

The authors are with the School of Engineering of the University of California, Merced, N. Lake Road, Merced CA 95343. {mkallmann,yhuang,rbackman}@ucmerced.edu

while discrete planning is well suited for sequencing stepping motions, arm reaching is best addressed by searching the continuous configuration space of the arm.

Multi-modal planning has recently emerged for humanoids and has been in particular developed for achieving locomotion and climbing in difficult terrains [2, 8, 9, 13] and also to sequentially coordinate walking and pushing [10]. With a focus on locomotion, extensions to the basic PRM method for handling multi-modal problems have also been proposed [7]. However, less attention has been given for the specific purpose of coordinating locomotion with reaching motions, which is the main topic addressed in our present work.

The multi-skill framework proposed in this paper addresses the problem of coordinating stepping and reaching motions as a multi-modal planning problem. In doing so the presented algorithm proposes a novel hybrid mechanism for solving multi-modal problems, integrating two types of search: 1) sampling of motion variations for concatenation of mobility skills and 2) bidirectional systematic exploration of manipulation skills for precisely reaching given targets. The overall approach is therefore able to explore the concatenation of locomotion skills until the target manipulation can be rapidly solved with a bidirectional search. The presented results demonstrate the suitability of the approach for solving several complex humanoid reaching tasks.

III. Motion Skills

The presented framework considers that each available motion skill is able to produce specialized motions efficiently according to its own parameterization scheme.

Let $\mathcal{C}$ be the d-dimensional configuration space of the humanoid being controlled and $\mathcal{C}_{free}$ the subspace representing the valid configurations. Configurations in $\mathcal{C}_{free}$ are collision-free, in balance and respect articulation constraints.

One of the purposes of using motion skills is their specialized ability to control the humanoid in a particular mode. The discrete set M is used to represent humanoid modes by specifying the state of each end-effector. Four letters are used: f for free (or unconstrained), m for when the end-effector is being used for object manipulation, s for when it is being used to support the humanoid, and o for when the end-effector can be optionally used as a support. Optional supports are in particular detected when a foot is correctly placed on the floor but without supporting any weight, meaning that the foot can be used or not as support by a subsequent skill.

Assuming that the humanoid structure in consideration has four end-effectors (two feet and two hands), a mode is then represented as a four-letter string, such as: "$ssff$" for a standing rest pose, or "$ssmf$" for a standing pose with one hand grasping an object, etc.

$\mathcal{X}$ is the state space of the planning problem, which is defined as the Cartesian product $\mathcal{X} = \mathcal{C} \times \mathbb{R}^+$, where $\mathbb{R}^+$ is used to represent time. Therefore $x = (q, t) \in \mathcal{X}$ denotes a configuration q at time t. The function $m(x) \in M$ is used to compute the mode of the humanoid at state x in the current environment.

$\mathcal{S}$ represents the skill base of the humanoid and is the finite set of all available motion skills. $\mathcal{S}(x) \subset \mathcal{S}$ represents the subset of skills which can be instantiated at x, i.e., which are applicable to take control over the humanoid in state x and mode $m(x)$. Each skill is essentially a controller specialized to operate in a particular set of modes and is responsible for checking the feasibility of instantiation, for example: a foot placement skill will only be instantiated if there is an unconstrained foot to be controlled, etc.

A skill $\sigma^x \in \mathcal{S}(x)$ is a function of the type $\sigma^x : P_\sigma \times [0, 1] \rightarrow \mathcal{X}$, where P_σ is its parametric control space, and $[0, 1]$ is the normalized time parameterization of the produced motion, such that $\forall p \in P_\sigma$, $\sigma^x(p, 0) = x$, and $\sigma^x(p, 1)$ is the final pose produced by the skill. Therefore, once a skill is instantiated it can then be evaluated in $[0, 1]$ in order to obtain the states traversed by the produced motion. Note that if there is a $p \in P_\sigma$ such that $\sigma^x(p, 1) = y$, then σ^x can be used to create a motion between states x and y, and the notation $\sigma^x(y, t)$ may be used instead of $\sigma^x(p, t)$.

Skills are also classified according to their main purpose. Currently two types of skills are considered: mobility skills and manipulation skills. The type informs how each skill is supposed to be controlled by the multi-skill planner. While manipulation skills are responsible for controlling end-effectors in order to reach targets which are in reachable range, mobility skills are responsible for updating the position of the humanoid until the target becomes reachable.

Given an initial state x_i and a final set of goal states X_g, the goal of the multi-skill planner is to produce a sequence of n skills, together with their application parameters such that, after the sequential application of all the skills, the humanoid will have moved from x_i to a state $x_g \in X_g$ and only traversing states with configurations in $\mathcal{C}_{free}$. Note that in this way the search space considered by the multi-skill planner is restricted to the parametric spaces of the available skills.

In this paper, the specific case of coordinating body motions for object grasping is used to demonstrate the framework. The goal is to precisely reach given targets with the hand and therefore the final set of goal states X_g represents all body postures with the hand precisely reaching the given target placement. The motion skills developed for solving these reaching tasks are described below.

A. Reaching Skill

The reaching skill σ_{reach} is a manipulation skill parameterized by the target location to be reached by a hand or foot and was developed with analytical IK formulations for the arms and legs of the HOAP-3 robot.

Given a parameter p, σ_{reach} produces a motion from the current posture to a final pose with the hand or foot exactly reaching the target position encoded in p. The legs of the HOAP-3 platform have 6 degrees of freedom (DOFs) and therefore the analytical solution can precisely compute joint angles for placing the feet in given 6 DOFs targets (position and orientation). The arms only have 5 DOFs and so the IK formulation only ensures that the target position is met, leaving the end-effector with the best orientation possible.

Whenever the skill is instantiated in a given body posture, the reaching skill will first determine with the IK a final arm posture reaching the given location p. The motion between the initial posture and the final posture is produced by interpolating the current hand position to the target in workspace. Each intermediate pose is therefore also computed by IK. In this way a reaching motion with the hand describing a rectilinear trajectory in workspace is obtained (see Figure 2). The time parameterization is mapped to a spline curve in order to obtain a bell-shaped velocity profile. These characteristics encode observations of how realistic arm motions are performed by humans [25].

The use of an analytical IK formulation is important due its fast computation, as the motion planner will sample several variations of each motion skill. The fact that the analytical formulation only affects the arm (or leg) does not pose a problem since other skills (as the balance skill introduced next) will affect the other parts of the body. Note also that the IK formulation observes joint range limits and the planner will test for motion feasibility each time a skill is used.

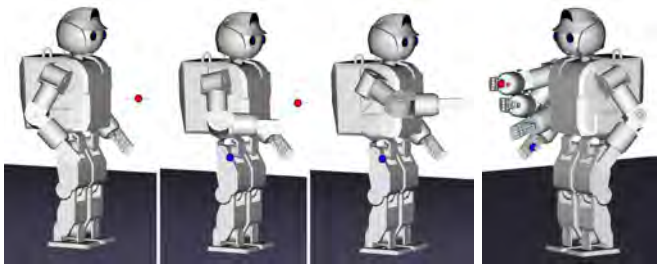

Fig. 2. The motion obtained by the reaching skill produces a straight-line trajectory of the end-effector in workspace.

B. Stepping Skill

The stepping skill σ_{step} is a mobility skill that moves each leg individually and is implemented using the same IK formulation as in σ_{reach}. The main difference is that σ_{step} is parameterized by a target position and orientation on the floor to be reached by the foot. Contacts between the feet and the environment are constantly monitored for determining the support mode and balance validity. In this paper only the ground is considered to be a valid surface for support.

Given the current mode $m(x)$, σ_{step} can only be instantiated for a leg which is unconstrained, and two versions exist: σ_{lstep} controls the left leg and σ_{rstep} controls the right leg. Also note that in order to produce valid motions between the current foot position to a new placement, σ_{step} generates an arc-shaped trajectory in the vertical plane such that the foot will not slide on the ground between the two placements. Figure 3 illustrates the generated motion.

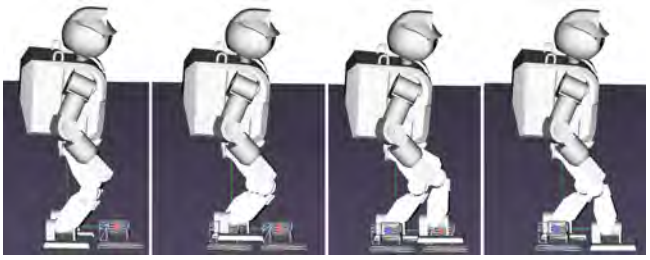

Fig. 3. Lateral view of a leg motion produced by the stepping skill.

C. Balance Skill

The balance skill σ_{bal} is another mobility skill but which has as main purpose to vary the support mode in order to allow different stepping skills to be instantiated. It is parameterized by a displacement (in position and orientation) to be achieved by the root joint of the robot. The desired motion is achieved by moving the joint angles of the legs towards new configurations which will lead to the desired root joint displacement. The analytical IK formulation is again used to determine the joint angle variations of the legs while exactly maintaining the feet placements.

The balance skill provides the key capability of transitioning between different leg support modes. The support mode is constantly monitored during the application of motion skills as it will influence the set of applicable motion skills, i.e. the skills which can be instantiated at a given humanoid configuration. The balance skill will in particular be responsible to free one leg from supporting the humanoid, allowing it to take a step towards a new placement. Figure 4 illustrates two different motions generated by the balance skill.

D. Sampling Skill Variations

Skills also determine bounds for their parameters in order to allow meaningful sampling of motions produced by instantiated skills. Figure 5 illustrates few variations obtained when sampling the parametric spaces of the skills considered in this work. The multi-skill planner will use this sampling functionality in order to search for concatenations between skills, and in the case of manipulation skills, to also search for intermediate motions around obstacles when collisions with obstacles are detected.

IV. Multi-Skill Planner

The multi-skill planner (MSP) maintains a search tree T of visited states and a priority queue Q with the nodes in the current expansion front. When MSP starts, T and Q are initialized with the current state of the humanoid. Queue

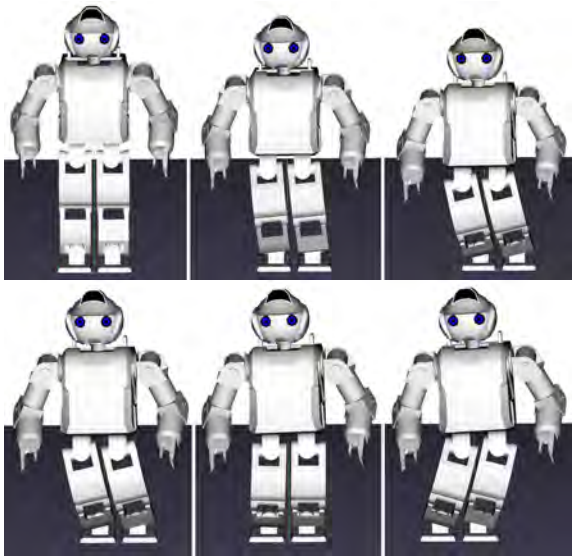

Fig. 4. Example of motions obtained with the balance skill. Top sequence: from standing pose (mode "ssff") to single foot support (mode "osff"). Bottom sequence: transitioning the support mode from "soff" to "osff". The vertical line shows the projection of the center of mass to the floor. The intersection with the support polygon (also shown) reveals the support mode of the humanoid.

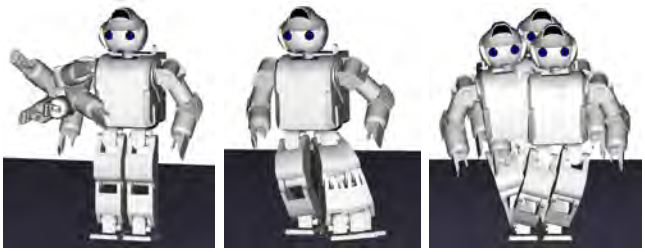

Fig. 5. Examples of the final postures obtained when sampling motions produced by σ_{reach} (left), σ_{lstep} (center), and σ_{bal} (right).

Q is prioritized by a cost associated with each enqueued state and the initial state receives cost 0. Then, given a goal position p_g and orientation q_g, the skill expansion procedure is repetitively called until the humanoid reaches a state in X_g, i.e. a state where one of its hand is exactly reaching target (p_g, q_g). If a given maximum elapsed time passes without success, the expansion stops and MSP reports failure.

The skill expansion routine is detailed in Algorithm 1. It basically selects skills and performs the search expansion strategy according to each skill type. At each call, the procedure removes the lowest cost (higher priority) state x from Q and selects all skills which are applicable (line 2), i.e. which can be instantiated at state x. For each applicable skill, the corresponding expansion method is then selected and applied. Note that skills will only be applicable if the controlled end-effector is free or optionally free (letters f or o in the mode encoding). Therefore, for each new state x being processed, the center of mass of the humanoid and its support polygon are computed in order to determine the mode description $m(x) \in M$. In the scope of this paper, only static balance tests based on the location of the projected center of mass onto the support polygon are computed.

Manipulation skills are only considered by the planner if the goal location is reachable by the skill from the current state. This test is performed in lines 5 and 6. When a manipulation skill is selected for expansion, a candidate goal state x_g reaching the goal has been already determined (line 5) and a bidirectional RRT-like exploration starts between x and x_g in order to determine if a valid motion joining the two states can be found, in which case the algorithm successfully terminates. If the two states can be connected by a single application of the manipulation skill, the bidirectional search will trivially find such solution.

Algorithm 1 Skill expansion of the multi-skill planner.

Expand_Skill (Q, T, p_g, q_g)

1. $x \leftarrow Q$.remove_lowest_cost ()
2. $\mathcal{S}(x) \leftarrow$ applicable_skills ($\mathcal{S}$)
3. **for** (each σ^x in $\mathcal{S}(x)$) **do**
4. **if** (σ^x type is manipulation) **then**
5. $x_g \leftarrow \sigma^x((p_g, q_g), 1)$
6. **if** ($x_g \neq null$) **then**
7. **for** (k_1 times) **do**
8. **Expand_Bidirectional_Search** (x, x_g, σ^x)
9. **if** (connection found) **then**
10. **return** SOLUTION_FOUND
11. **end if**
12. **end for**
13. attach the expanded bidirectional trees to x
14. $n \leftarrow$ total number of nodes in the trees
15. Q.insert (x, $f_{cost}(x, p_g, n)$)
16. **end if**
17. **else if** (σ^x respects its mobility sequence) **then**
18. **for** (k_2 times) **do**
19. $p \leftarrow$ sample (P_σ)
20. **if** (motion generated by $\sigma^x(p)$ is valid) **then**
21. $x' \leftarrow \sigma^x(p, 1)$
22. T.append_child (x, x', σ^x, p)
23. Q.insert (x',$f_{cost}(x', p_g, 0)$)
24. **end if**
25. **end for**
26. **end if**
27. **end for**
28. **return** NOT_YET_FOUND

Skill σ_{reach} is the only considered manipulation skill and two versions of it are available in $\mathcal{S}$: one for the right hand and another one for the left hand. In this way the planner will naturally select the most suitable hand to reach the goal. The bidirectional search for the manipulation skill is then expanded up to k_1 times (lines 7-12).

Algorithm 2 details the process: each bidirectional search is initialized with one tree rooted at the current state x and the second tree rooted at the goal state x_g. At each iteration, the algorithm tries to grow the trees in the direction of random

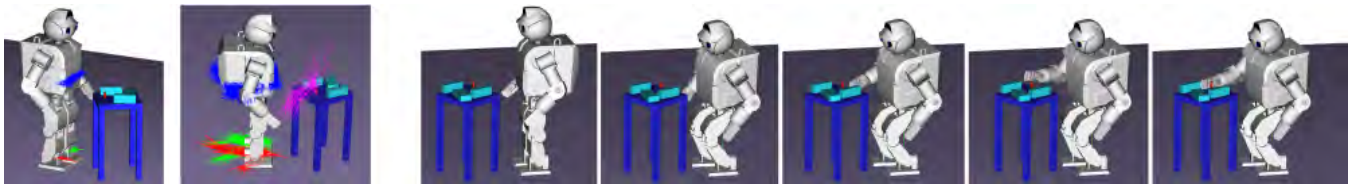

Fig. 6. The left-most image shows the edges expanded for finding the first solution for reaching the target among obstacles on top of the table. The second image illustrates the expansion of several solutions. The five right-most images, from left to right, show poses of the first solution found (left view).

landmarks x_{rand} by a motion of length δ. The iterations will eventually produce a collision-free path by concatenation of the landmarks, or in case k_1 iterations pass without success, the process stops.

Parameter k_1 controls the tradeoff between insisting in trying to reach the target from the current body placement, or suspending the search to let new manipulation instantiations to try reach the target from different body placements. When the k_1 iterations pass, the bidirectional search is suspended and attached to the current state x, which is re-inserted in Q with an updated cost encoding the time invested so far in that expansion. Note that if x is removed again from Q in a later expansion, the attached bidirectional search will be again executed for a maximum of k_1 additional expansions. In this way different instantiations of σ_{reach} compete with each other in order to find the overall minimum-cost solution for the task (see Figure 8).

Algorithm 2 Bidirectional expansion of manipulation skills

Expand_Bidirectional_Search (x, x_g, σ^x)

1. if no bidirectional trees attached to x, attach empty ones
2. $t_a \leftarrow$ attached tree rooted at x
3. $t_b \leftarrow$ attached tree rooted at x_g
4. $x_{rand} \leftarrow \sigma^x($ sample(P_σ) , 1)
5. $x_a \leftarrow$ closest configuration to x_{rand} in t_a
6. $x_b \leftarrow$ closest configuration to x_{rand} in t_b
7. **if** (motion generated by $\sigma^{x_a}(x_b)$ is valid) **then**
8. **return** CONNECTION_FOUND
9. **else**
10. try_to_expand (t_a, x_a, x_{rand}, δ)
11. try_to_expand (t_b, x_b, x_{rand}, δ)
12. **return** NOT_YET_FOUND
13. **end if**

Mobility skills are selected for expansion in line 17 of Algorithm 1. In order to ensure that the planner concatenates a meaningful sequence of mobility skills, each mobility skill provides a mobility sequence to be respected. Skill σ_{lstep} requires the grand-parent and the parent nodes of the current state x to have been generated by skills σ_{rstep} and σ_{bal} respectively. This avoids the planner to spend time with non-useful stepping sequences such as: σ_{lstep}, σ_{bal} and σ_{lstep}. Similarly, skill σ_{rstep} constraints the two previously applied skills to be σ_{lstep} and σ_{bal}. Finally skill σ_{bal} simply does not allow repetition, i. e. it restricts that the previous skill cannot be as well a σ_{bal} skill. Constraining sequences with per-skill specifications has greatly improved the overall performance of the planner.

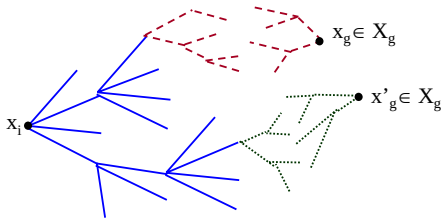

Fig. 8. The concatenation of mobility skills (continuous blue edges) will lead to several states which can initiate expansions of manipulation skills. Here the dashed red edges and the dotted green edges represent bidirectional expansions trying to reach the goal from different states. Each bidirectional expansion grows for up to k_1 times.

Selected mobility skills are expanded at most k_2 times (line 18 of Algorithm 1). Each expansion selects a target state by sampling a control parameter p from the parametric control space of the skill. Each time p is sampled, the generated motion between $\sigma^x(p,0)$ and $\sigma^x(p,1)$ is tested for validity (line 20). If the motion is valid, a new valid state $x' = \sigma^x(p,1)$ has been reached and x' is then added to T and Q.

The motion validity tests (Algorithm 1 line 20 and Algorithm 2 line 7) will evaluate the motion in several intermediate states by recursive bisection until a given precision is reached. At each evaluated state, joint limits compliance, collision detection and balance tests are performed. The continuous validity problem is therefore solved discretely and the motion is determined valid only if all intermediate states are valid.

Costs Every time a new state is inserted in Q (lines 15 and 23) of Algorithm 1, its cost is computed. The cost dictates which states are to be expanded first and its computation can take different metrics into consideration. The cost function used in this work encodes the following terms:

$$f_{cost}(x, p_g, n) = d_c(x_i, x) + w_g d_g(x, p_g) + w_e n.$$

Term $d_c(x_i, x)$ encodes the usual cost-to-come and is computed as the sum of the costs in all the edges in the T branch from the root node x_i to the current node x. Edge costs encode the displacement of the motion represented by each edge. Term $d_g(x, p_g)$ encodes the cost-to-go heuristic, and is set as the distance between the goal point and the mid-point between the shoulder joints of the humanoid at state x. This cost is weighted by w_g and makes states closer to the goal to be expanded first.

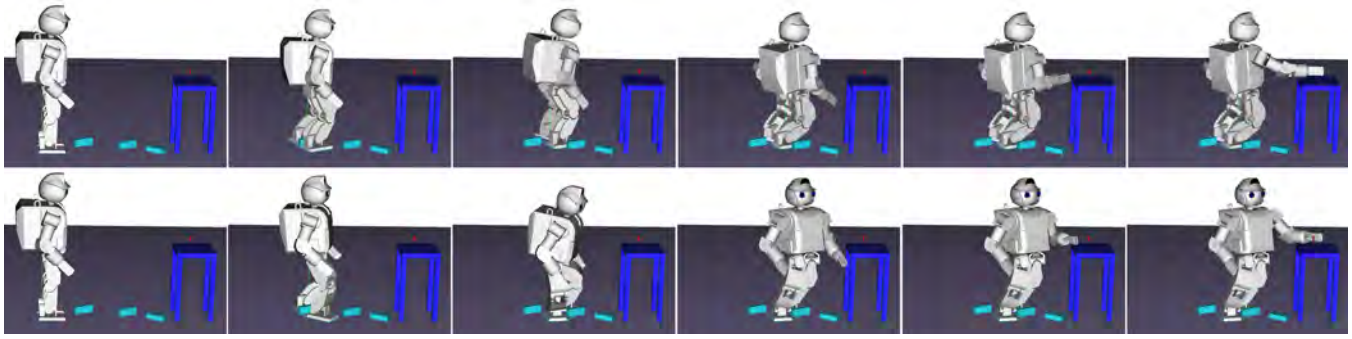

Fig. 7. The sequences show snapshots of two solutions for a reaching task requiring several steps among obstacles on the ground. The top sequence shows a solution with the right arm reaching the goal. The bottom sequence shows a solution with the left arm. These solutions had similar costs.

The final term is weighted by w_e and penalizes states which have already been expanded (by n expansions) in a bidirectional manipulation search. This term does not affect states reached by mobility skills which will always have $n = 0$ (Algorithm 1 line 23).

V. Results and Discussion

Figure 6 illustrates a typical solution obtained by MSP for reaching a target among a few obstacles in a table. In the example shown by Figure 7 the two versions of σ_{reach} are included in $\mathcal{S}$ (one for each hand) and the planner was able to find solutions using both hands. Figure 9 shows another solution obtained in a more constrained environment. In all cases, the obtained solutions represent coordinated, statically-stable and collision-free motions.

The first two images of Figure 6 show colored edges representing the expanded nodes of the search. The blue edges show the root position variation generated with the application of skill σ_{bal}. Red edges represent variations of the right foot position generated by instantiations of σ_{rstep}. Green edges represent variations of the left foot position generated by σ_{lstep}. The edges in magenta represent the hand trajectories controlled by σ_{reach}. These examples show that most of the solutions are composed of several body adjustments until arm reaching is feasible, in the same fashion depicted by the diagram of Figure 8.

The obtained results show that MSP is able to find suitable body placements for supporting manipulation tasks and that several solutions can be explored when the algorithm is not stopped at the first (minimum-cost) solution. Additional metrics can also be included in the cost function for taking into account: low energy consumption, security distance from obstacles, etc.

One main strength of the proposed approach is the ability to integrate the discrete expansion of mobility skills with the systematic and bidirectional expansion of manipulation skills, allowing both types of search to be concurrently expanded in order to explore and evaluate different concatenations of skills. The amount of expansions performed at each step can be controlled by parameters k_1 and k_2, which will directly control the branching factor of the search. Note that no mechanisms have been included to detect and prevent regions in $\mathcal{X}$ to be excessively visited. Such mechanisms could be integrated with the use of hash functions, and would be necessary in dense expansions.

Another important observation is that MSP is able to generate crude locomotion patterns. Locomotion patterns could be further optimized and parameterized, and even re-inserted as new higher-level skills to become available to the planner. The possibility of integrating an automated process of learning of higher-level skills could lead to a powerful knowledge-base learning approach for motion planning. Alternatively, hand-crafted skills for producing a walking gait could be easily integrated in the framework and would quickly bring the humanoid close to the object to be manipulated.

Note also that the computed solutions are described by a precise sequence of skills and their instantiation parameters. This allows motion optimization algorithms to be developed to operate on the parametric control space of each skill, instead of operating on the full state space of the humanoid. In our present work, optimization and smoothing techniques were not applied to the presented results. However several well-known techniques from the motion planning field can be readily integrated for further improving the computed motions.

Depending on the complexity of the problem being solved, the computational time required for finding a solution may vary from few seconds to several minutes. The solutions illustrated in Figures 1, 6, 7, and 9 took respectively: 1 second, 49 seconds, 7 minutes, and 3 hours on an Intel Q9450 CPU. In most of these examples parameters $k_1 = 60$ and $k_2 = 10$ were used. Note that the planner is not designed to plan long stepping sequences and long solutions imply excessive search of stepping combinations leading to many minutes of computation. Instead, the planner is designed to solve short concatenations of stepping and body adjustments for the purpose of supporting manipulation tasks. Furthermore, our prototype implementation can still be significantly improved with the purpose of reducing computation time, in particular in respect to precomputing skill samples and

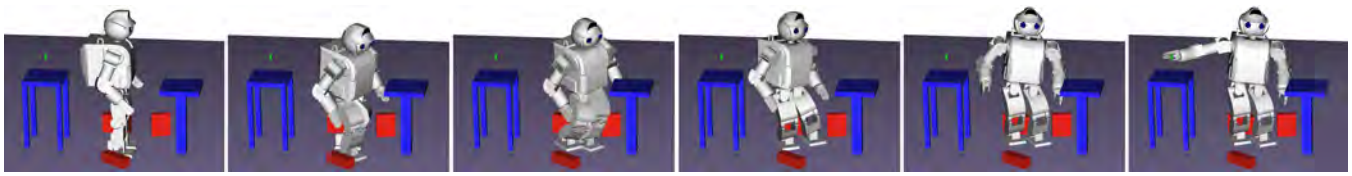

Fig. 9. In this example the target is behind the initial pose of the robot and there are few obstacles on the ground constraining the free space. This environment forces the robot to perform several rotational steps until it is able to reach the goal with the right hand.

employing specific continuous collision detection procedures based on simplified geometries.

The produced reaching motions have also been successfully transferred to the HOAP-3 platform for the realtime control of reaching tasks. Figure 10 shows snapshots of one of our experiments. In this example, markers tracked by motion capture cameras are used to record the position of the target to be reached. The planner then typically takes a few seconds for computing a solution (of relatively short duration), which can then be transferred and executed by the robot. Our realtime controller converts the sequence of joint angles of the solution motion into encoder values which are sent to the robot at about 30 Hz.

The presented experiments show that the produced plans have enough precision for being successfully executed by the humanoid robot. Several extensions are being carried out for improving the robustness of the humanoid control. For instance we are currently integrating reactive controllers monitoring the robot's sensors with the specific purpose of improving balance maintenance and target tracking while executing computed motions.

The video accompanying this paper presents animations of some of the obtained results and also demonstrations of planned motions being applied to the HOAP-3 humanoid.

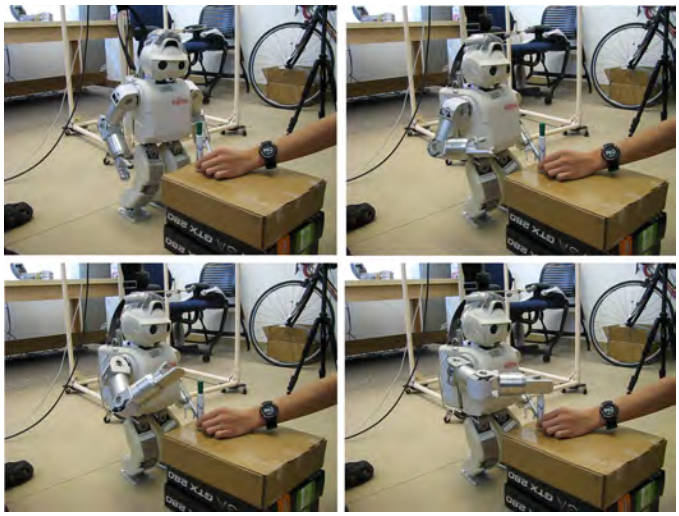

Fig. 10. Applying a solution motion to the robot.

VI. Conclusions and Future Work

This paper describes a multi-skill motion planning framework able to plan the sequencing of generic motion skills. The presented results demonstrate that the proposed framework is able to solve multi-modal problems involving mobility and manipulation.

This approach has the potential to lead to optimal whole-body humanoid performances which are closer to human-like strategies. Furthermore, the proposed skill-based approach has the potential to enable a human-like automated way of learning complex skills from basic ones. The presented framework has therefore the potential to be useful to a wide range of humanoid applications related to achieving intelligent autonomous assistants.

As future work, we are exploring several topics for improvement of the framework: 1) the use of higher-level skills, 2) skill parameterization properties for improving the planning, and 3) mechanisms for learning basic coordinations in order to solve repeated situations fast.

Acknowledgments This work was partially supported by NSF Award BCS-0821766.

References

[1] D. Bertram, J. Kuffner, R. Dillmann, and T. Asfour. An integrated approach to inverse kinematics and path planning for redundant manipulators. In *Proceedings of the IEEE International Conference on Robotics and Automation*, pages 1874–1879. IEEE, May 2006.

[2] T. Bretl. Motion planning of multi-limbed robots subject to equilibrium constraints: The free-climbing robot problem. *International Journal of Robotics Research*, 25(4):317–342, 2006. ISSN 0278-3649.

[3] J. Chestnutt, M. Lau, K. M. Cheung, J. Kuffner, J. K. Hodgins, and T. Kanade. Footstep planning for the honda asimo humanoid. In *ICRA*, April 2005.

[4] J. Chestnutt, K. Nishiwaki, J. Kuffner, and S. Kagami. An adaptive action model for legged navigation planning. In *Proceedings of the IEEE/RAS International Conference on Humanoid Robotics*, 2007.

[5] R. Diankov and J. Kuffner. randomized statistical path planning. In *Proceedings of the International Conference on Robotics and Automation (ICRA)*, pages 1–6, May 19-23 2008.

[6] E. Drumwright and V. Ng-Thow-Hing. Toward interactive reaching in static environments for humanoid

robots. In *Proceedings of the IEEE International Conference on Intelligent Robots and Systems (IROS)*, Beijing, China, October 2006.

[7] K. Hauser and J. Latombe. Multi-modal motion planning in non-expansive spaces. In *Proceedings of the 8th Workshop on Algorithmic Foundations of Robotics (WAFR)*, December 7-9 2008.

[8] K. Hauser, T. Bretl, and J. Latombe. Non-gaited humanoid locomotion planning. In *Humanoids*, pages 2641– 2648, December 2005.

[9] K. Hauser, T. Bretl, K. Harada, and J. Latombe. Using motion primitives in probabilistic sample-based planning for humanoid robots. In *Workshop on Algorithmic Foundations of Robotics (WAFR)*, pages 2641– 2648, July 2006.

[10] K. K. Hauser, V. Ng-Thowhing, Gonzalez-Baos, H. Mukai, and S. Kuriyama. Multi-modal motion planning for a humanoid robot manipulation task. In *International Symposium on Robotics Research*, 2007.

[11] S. Kagami, J. Kuffner, K. Nishiwaki, M. Inaba, and H. Inoue. Humanoid arm motion planning using stereo vision and rrt search. *Journal of Robotics and Mechatronics*, April 2003.

[12] M. Kallmann. Scalable solutions for interactive virtual humans that can manipulate objects. In *Proceedings of the Artificial Intelligence and Interactive Digital Entertainment (AIIDE'05)*, pages 69–74, Marina del Rey, CA, June 1-3 2005.

[13] M. Kallmann, R. Bargmann, and M. J. Matarić. Planning the sequencing of movement primitives. In *Proceedings of the International Conference on Simulation of Adaptive Behavior (SAB)*, pages 193–200, Santa Monica, CA, July 2004.

[14] L. Kavraki, P. Svestka, J.-C. Latombe, and M. Overmars. Probabilistic roadmaps for fast path planning in high-dimensional configuration spaces. *IEEE Transactions on Robotics and Automation*, 12:566–580, 1996.

[15] S. Koenig. A comparison of fast search methods for real-time situated agents. In *Proceedings of the International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS)*, pages 864–871, 2004.

[16] S. Koenig and M. Likhachev. Real-time adaptive A*. In *AAMAS*, pages 281–288, 2006.

[17] Y. Koga, K. Kondo, J. J. Kuffner, and J.-C. Latombe. Planning motions with intentions. In *Proceedings of SIGGRAPH*, pages 395–408. ACM Press, 1994.

[18] J. Kuffner, K. Nishiwaki, S. Kagami, M. Inaba, and H. Inoue. Motion planning for humanoid robots. In *Proceedings of the 11th International Symposium of Robotics Research (ISRR)*, November 2003.

[19] J. J. Kuffner and S. M. LaValle. RRT-Connect: An efficient approach to single-query path planning. In *Proceedings of IEEE International Conference on Robotics and Automation (ICRA)*, San Francisco, CA, April 2000.

[20] J.-C. Latombe. *Robot Motion Planning*. Kluwer Academic Publisher, December 1990.

[21] J.-P. P. Laumond. *Robot Motion Planning and Control*. Springer-Verlag New York, Inc., Secaucus, NJ, USA, 1998. ISBN 3540762191.

[22] S. LaValle. Rapidly-exploring random trees: A new tool for path planning. Technical Report 98-11, Iowa State University, Computer Science Department, October 1998.

[23] S. M. LaValle. *Planning Algorithms*. Cambridge University Press (available on-line), 2006. URL `msl.cs.uiuc.edu/planning/`.

[24] M. Likhachev, G. J. Gordon, and S. Thrun. ARA*: Anytime A* with provable bounds on sub-optimality. In S. Thrun, L. Saul, and B. Schölkopf, editors, *Advances in Neural Information Processing Systems 16*. MIT Press, Cambridge, MA, 2004.

[25] S. Schaal. Arm and hand movement control. In M. Arbib, editor, *The handbook of brain theory and neural networks*, pages 110–113. The MIT Press, second edition, 2002.

[26] R. Schmidt and T. Lee. *Motor Control and Learning: A Behavioral Emphasis*. 2005. ISBN 073604258X.