A Motion Planning Framework for Skill Coordination and Learning

Marcelo Kallmann and Xiaoxi Jiang

To appear in Motion Planning for Humanoid Robots
Harada, Kensuke; Yoshida, Eiichi; Yokoi, Kazuhito (Eds.), Springer, 2010

Abstract Coordinated whole-body motions are key for achieving the full potential of humanoids performing tasks in human environments and in cooperation with humans. We present a multi-skill motion planning and learning framework which is able to address several complex whole-body coordinations and as well detection and imitation of motion constraints. The framework is composed of three main parts: first, a minimal set of basic motion skills is defined in order to achieve basic stepping, balance and reaching capabilities. A multi-skill motion planner is then developed for coordinating motion skills in order to solve generic mobile manipulation tasks. Finally, learning strategies are presented for improving skill execution and for learning constraints from imitation. The framework is able to coordinate basic skills in order to solve complex whole-body humanoid tasks and also integrates learning mechanisms for achieving humanlike performances in realistic environments.

1 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 typical humanoid tasks require coordination of different types of motions or skills and

Marcelo Kallmann
University of California, Merced; 5200 N. Lake Road, Merced CA 95343
e-mail: mkallmann@ucmerced.edu

Xiaoxi Jiang
University of California, Merced; 5200 N. Lake Road, Merced CA 95343
e-mail: janexip@gmail.com

therefore cannot be solved by a single planning strategy. One additional challenge is to achieve interactive performances: planning motions from scratch is often computationally intensive and therefore learning has to be addressed in an integrated fashion.

Consider for example the coordination of stepping and grasping. While stepping is mainly concerned with balance maintenance for mobility, reaching and grasping skills control the motion of the arms and hands assuming the humanoid to be in a well balanced standing position. Figure 1 illustrates a typical scenario where complex body positioning is required in order to support common book relocations in a shelf. It is possible to note that complex torso motions are both used for providing balance and for enlarging the reachable space of the arms. When body motion is not enough, stepping is also employed.

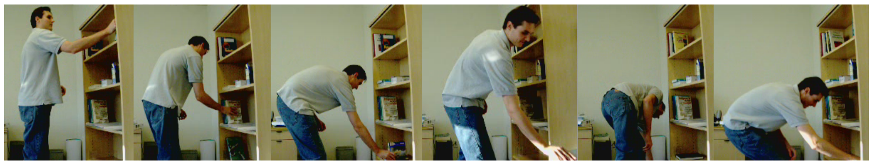

Fig. 1 Even in simple manipulation tasks complex leg-arm coordinations can be observed.

We address the whole-body motion generation problem for reaching tasks as a planning problem. We target object reaching tasks such as the ones illustrated in Figure 1. The goal is to plan the coordination of stepping, balance and reaching motion skills in order to reach given target locations with end-effectors.

Figure 2-left illustrates a situation where the fully articulated humanoid character is not able to grasp the book using its arm because the book is not within reachable range. This problem is even more critical in humanoid robots as they usually have less degrees of freedom (DOFs) and with less range of motion, therefore requiring more complex coordinations. Depending on the motion skills available, humanoids can solve given tasks in different ways, for instance: by performing some steps until the object becomes reachable, by adjusting the body posture (without stepping) until the target is reachable (for example by bending the knees and the torso while keeping in balance), etc. In the proposed framework, a motion planner is employed for searching for a solution and for evaluating the best solution whenever several possibilities exist.

It is important to notice that the problem of skill coordination appears frequently in common tasks such as for relocating objects, opening doors, interacting with machines and appliances, etc. Most importantly this class of

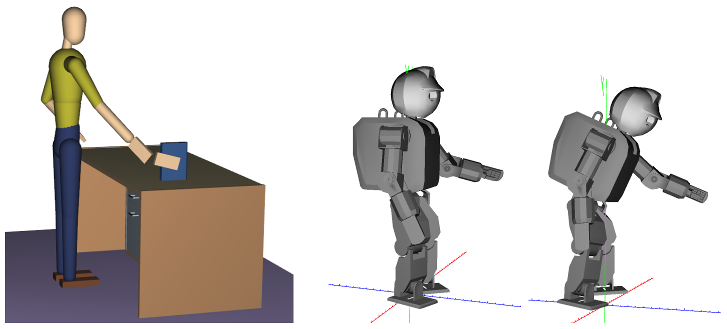

Fig. 2 Left: example of a common grasping task which cannot be solved with a simple arm reaching motion even by a fully articulated humanoid model. Right: in most humanoid robots a balanced torso motion will be needed for enlarging the reachable workspace of the arms. Whole-body coordination is in particular important for less articulated arms as for example in the shown Fujitsu's HOAP-3 humanoid model which has only 5 degrees of freedom in each arm. In such cases body positioning has a critical role in successfully reaching target locations.

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.

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 for humanoid assistants interacting and collaborating with people, as for instance: for assistance with physical therapy [49], for performance of collaborative work [6], and for assistance in space exploration [16].

We are in particular interested in developing techniques capable of solving the motion generation problem similarly to how humans do, i.e., achieving humanlike performances. Therefore learning is a critical component which has to be addressed. Each time a task is planned and solved it should be analyzed and stored for improving the performance in subsequent similar tasks. In this way motions which have been learned should be quickly executed in subsequent tasks which are similar with the learned ones, similarly to how humans perform tasks. Although achieving generic and robust learning techniques remains a challenge, the proposed framework integrates learning in different ways: for improving skill execution, for learning constraints from imitation and as well for learning higher-level motion skills.

1.1 Related Work

Traditional motion planning approaches [43, 44, 46] are fundamentally based on systematic search in configuration spaces. Among the several techniques, sampling-based methods such as Probabilistic Roadmaps (PRMs) [35] and Rapidly-Exploring Random Trees (RRTs) [41, 45] 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 [11, 12, 40], arm reaching motions for manipulation [4, 15, 17, 30, 31, 39], 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 [37, 38, 47] are well suited for finding the best sequence of actions to be taken. No single planning approach is best in all cases. For instance, while discrete planning is well suited for sequencing stepping motions, arm reaching is better addressed by searching in the continuous configuration space of the arm.

The issue of producing whole-body motions from combination of skills has also been addressed without an explicit motion planner, for example including balance maintenance, energy minimization and friction [59], and also using dynamic controllers designed for animating characters in physics-based simulations [19, 28].

1.1.1 Multi-Modal Planning

Multi-modal planning has recently emerged for solving humanoid problems and typically it addresses both discrete and continuous search strategies in an integrated fashion. Multi-modal planning has been in particular developed for achieving locomotion and climbing in difficult terrains [8, 25, 26, 34] but also to sequentially coordinate walking and pushing [27]. Reactive approaches have also been employed for coordinating specific problems such as walking while carrying a large object [18].

We have also addressed the whole-body motion planning problem in previous work. A whole-body reaching skill was developed including coupled spine rotations and knee bending [33], and two main aspects of the coordination problem have also been proposed: 1) the sequencing of movement primitives for climbing around obstacles [34], and 2) the synchronization of concurrent primitives for simultaneous locomotion and object manipulation [60]. The sequencing problem was solved by a discrete search over multi-modal connections identified by a RRT on the parametric space of each primitive controller. The concurrency problem was solved by including the locomotion time parameterization as one additional (monotone) variable in the planning search space.

The multi-skill framework described here is an evolution of these previous works and aims at achieving a generic framework able to coordinate generic motion skills and as well to incorporate learning.

1.1.2 Learning for Motion Planning

Learning has been included in motion planning in a variety of different ways, for example: to select the best planning technique to be applied in a given situation [51], to concentrate sampling toward difficult regions of the configuration space [9], etc. These methods are mainly tailored for improving planning in difficult problems, and not for achieving humanlike performances in humanoid tasks. In contrast, learning reactive behaviors has been extensively addressed in the literature. For instance, reinforcement learning has been recently applied for learning motion transitions [50] and for mapping state-goal spaces for controlling walking with collision avoidance [64].

While no works have specifically focused on the development of a learning-capable motion planner framework based on motion skills, the concepts of motion modeling and motion knowledge are known strategies [42, 48], and the idea of learning complex motions from primitive skills [1] is supported by evidence obtained in several cognitive studies [20, 63].

Learning from demonstrations has also been addressed by imitation frameworks [7, 58] and its importance is supported by research on mirror neurons: neurons that enable humans and other primates to imitate actions [14, 21]. Imitation techniques are therefore especially relevant for controlling and programming tasks for humanoid agents [5, 53, 54, 56, 62], and represent a natural approach to human-computer interaction by analogy to the way humans naturally interact with each other. As a consequence, research on humanoid imitation from human motion is currently very popular [13, 23, 62]. An imitation framework based on motion capture is explored here as a way to acquire motion constraints to be associated with objects.

1.2 Framework Overview

The presented framework has been mainly designed based on graphical simulations of virtual humanoids and only taking into account kinematic and static balance tests. We believe that such an environment is suitable for planning as long as the motions have low energy, which is mostly the case in usual reaching and relocation tasks. As an example, we have also successfully applied results of our motion planning framework to control the HOAP-3 humanoid robot (as demonstrated in Figure 12).

Besides applications to humanoid robotics, we also believe that motion planning will soon provide important autonomy capabilities for interactive virtual humans, which can impact a number of applications in Virtual Reality, Education, Training and also Entertainment. A recent tutorial on motion planning for virtual humans well illustrates the potential of motion planning to virtual characters [55].

Figure 3 depicts the architecture of the proposed framework. Each box in the figure represents a main module of the architecture. These modules are described in the following paragraphs.

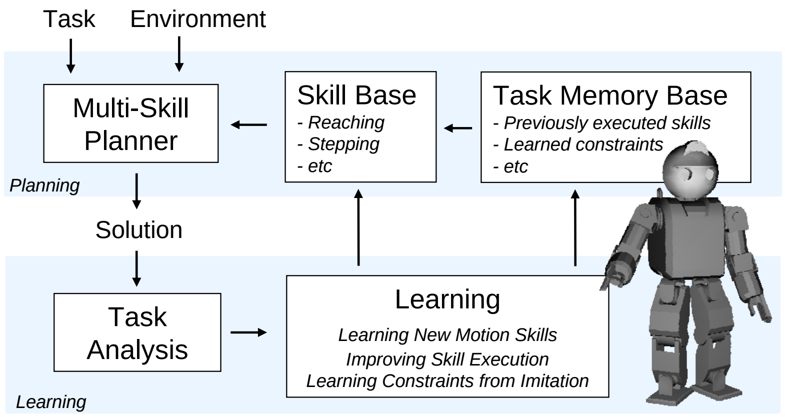

Fig. 3 Framework main modules.

- Skill Base: motion skills are organized in a skill base where each skill follows a unified representation and parameterization interface. Skills are designed to generically encapsulate different approaches for motion synthesis and to provide a unified control interface to the motion planner.
- Multi-Skill Planner: this module implements the planner which will search for coordinations between the available motion skills in order to achieve so-

lutions for given tasks. The output solution will be described as a sequence of instantiated skills needed to accomplish the task at hand.

- Task Analysis: once a solution is found for a given task, the solution is analyzed and classified in respect to the environment and task at hand, and as well in respect to previous solutions. The analysis will inform the learning module.
- Learning: this module includes several learning mechanisms supported by the framework. Learning may improve the knowledge of skills in the skill base, may form new skills by interpolation of similar solutions, and may also add learned features to the task memory base which will be used when planning similar future tasks.
- Task Memory Base: this module stores all relevant features learned from previously planned solutions. Two main aspects are addressed here: 1) learning attractor points for improving the planning of arm reaching motions in new but similar environments, and 2) for learning motion constraints automatically from motion demonstrations. Constraints are typically associated to objects and will inform if an object is supposed to move in a constrained way.

The reminder of this text is organized as follows: motion skills are described in the next section and the Multi-Skill Planner is described in Section 3. The task analysis, learning and the task memory base are presented in Section 4. Section 5 concludes this chapter.

2 Motion Skills

Motion skills are expected to produce specialized motions efficiently according to their own parameterization and one of the purposes of employing motion skills is their specialized ability to control the humanoid in a particular mode.

The discrete set M is used to represent all possible humanoid modes. A mode is represented by specifying the state of each end-effector. Three letters are used: f for free (or unconstrained), s for when the end-effector is being used to support the humanoid, and m for when it is being used for object manipulation. Therefore, assuming that four end-effectors are considered (two feet and two hands), a mode can be 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.

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.

Consider now that set $\mathcal{X}$ denotes 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. A function $m(x) \in M$ is also defined for retrieving the mode of the humanoid at state x.

The set $\mathcal{S}$ is defined to represent 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. 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 $\sigma^x : P_\sigma \times [0, 1] \to \mathcal{X}$, where P_σ is its parametric control space, and $[0, 1]$ is the normalized time parameterization of the produced motion, such that $\forall p \in P_\sigma$, $\sigma^x(p, 0) = x$. Therefore, once a skill is instantiated it can be evaluated in $[0, 1]$ in order to obtain the states traversed by the produced motion.

A skill is said to be explorative if it allows the re-instantiation at intermediate poses. Only explorative skills will allow the use of a sampling-based planning strategy to systematically expand a search tree in their parametric space. Skills which are not explorative will be only allowed to be concatenated and their parameterizations will only be traversed in a forward monotone fashion, allowing the encapsulation of algorithms based on forward simulations.

Finally, 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 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 $\mathcal{C}_{free}$.

In this work the specific case of coordinating body motions for object reaching is considered to demonstrate the framework. In this case, the goal of the humanoid is to reach with the hand a pre-defined hand target suitable for grasping the given object. Therefore the final set of goal states X_g represents all body postures with a hand precisely reaching the hand target.

Each skill defines its own parameterization. For instance a balance skill could be parameterized with a single real value for dynamically adjusting balance in one main direction, following the approach of *kinematic synergies* proposed by Hauser et al. [24]. As the main focus here is to address whole-body kinematic reaching problems, skills allowing the precise placement of joints are developed and they are parameterized by the target locations to be reached.

Three basic skills were developed and constitute the minimal set of skills for solving a variety of problems: a reaching skill, a stepping skill, and a body balance skill. These skills were implemented using an analytical Inverse Kinematics (IK) formulation [32] for arms and legs (see Figure 4). The analytical formulation provides a fast closed form solution for the control of end-effectors and is key for allowing the motion planner to quickly evaluate the motion generated by skills during the planning. The main limitation of the analytical formulation is that it cannot be applied for generic linkages and therefore the results presented here focus on controlling anthropomorphic limbs.

2.1 Reaching Skill

The IK-based reaching skill σ_{reach} is parameterized by the target location to be reached. Given a parameter p, σ_{reach} produces a motion from the current humanoid posture to a final pose with the hand exactly at the target position and orientation encoded in p.

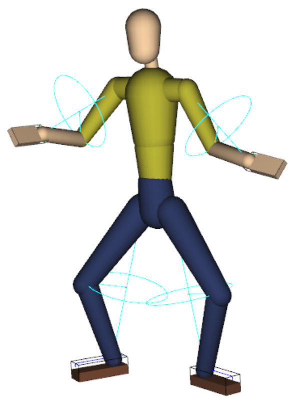
Fig. 4 The figure illustrates the analytical IK applied to 7-DOF arms and 7-DOF legs and the orbit circles containing the possible positions for the elbows and knees [32].

The reaching skill first uses the analytical IK to determine the final arm posture reaching location p. The determination of the final posture includes a fast search algorithm for determining the most suitable swivel angle along the orbit circle of the arm [32]. Once the final posture is determined, the motion between the current posture instantiated by the skill and the final posture is produced by interpolating, from the initial posture to the final posture, the following parameters: 1) the hand position, 2) the hand orientation and 3) the orbit angle. For each intermediate interpolated position and orbit angle, IK is again employed for computing the intermediate arm pose. Interpolation of the orientation is performed with quaternion spherical interpolation.

As a result, a realistic reaching motion with the hand describing a rectilinear trajectory in workspace is obtained (see Figure 5). 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 [57]. Anatomically-plausible joint parameterizations based on the swing-twist decomposition [22] and range limits based on spherical ellipses [3] are also included in order to represent human-like articulation limits.

This same skill has also been implemented for the HOAP-3 humanoid, the main difference is that the arms and legs of the robot have 5 and 6 DOFs respectively instead of 7 DOFs. The leg IK can therefore be exactly solved but without choice of the orbit angle. The arm IK however cannot be solved for arbitrary 6-DOFs targets and our current implementation assumes a predefined constrained target orientation to be solved.

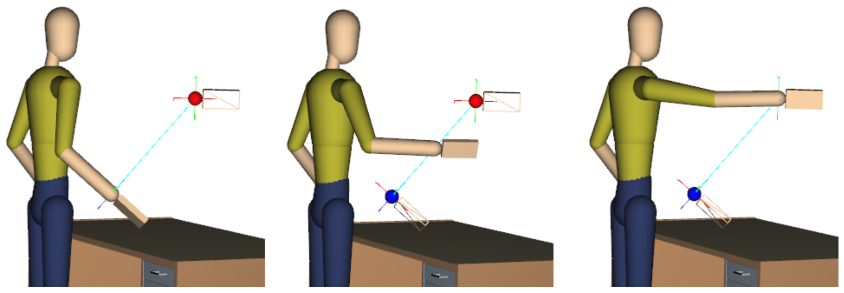

Fig. 5 The motion obtained by the reaching skill produces a straight-line trajectory of the end-effector in workspace.

2.2 Stepping Skill

The stepping skill σ_{step} controls one leg of the humanoid to perform the motion of one step towards another foot placement on the floor. Given a parameter p, σ_{step} produces a motion from the current humanoid posture to a final pose where the foot in control will exactly reach the target position and orientation encoded in p, where p is constrained to be a position where the foot aligns with the floor in a suitable manner to support the humanoid with that foot placement.

The motion generation follows the same interpolation strategies used in the reaching skill, however following a trajectory with a bell-like shape on the vertical plane, in order to achieve the one-leg step motion. Also, the stepping skill can only be instantiated when the leg being controlled is free to move. Therefore a step can only occur when the humanoid is in single support mode. Balance tests are computed by checking if the projection of the center of mass on the floor lies inside the support polygon. Only static balance is considered in the scope of this work.

2.3 Balance Skill

The balance skill σ_{bal} allows switching between support modes by adjusting the position of the body while maintaining feet placements. Its parameterization encodes the new target position and orientation for the root joint of the skeleton. Given a target location, the skill will then produce a motion that makes the root joint from the current location to the new desired location, while maintaining the same feet placements with IK.

The motion is computed as follows: first the new configurations of the legs maintaining the feet placements in respect to the target location of the root joint are computed with IK. The motion is then defined by interpolation between the initial configurations and final configurations, which are interpolated in workspace in order to ensure that the feet placements are maintained.

One interesting effect obtained for the 7-DOF legs solution with the orbit angle search mechanism of the analytical IK [32] is the automatic opening of the legs (for avoiding ankle joint limits) when the root joint is lowered. This however does not apply to the HOAP-3 legs, which have one less DOF.

The balance skill provides the 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 mode of the humanoid, and therefore also the set of applicable motion skills, i.e. the skills which can be instantiated at a given humanoid configuration.

Figure 6 shows two example motions obtained with σ_{bal}. The top sequence changes the humanoid mode from both-feet support "ssff" to single feet support "fsff". The bottom sequence changes the mode from "sfff" to "fsff". The balance motion skill can therefore be used to free one leg from supporting the humanoid so that a stepping skill can be applied to the free leg in order to place the (now free) foot in a new placement.

2.4 Other Skills and Extensions

The presented skills are enough for coordinating generic stepping and reaching motions. These skills also provide enough flexibility for planning whole-body coordinations for reaching tasks.

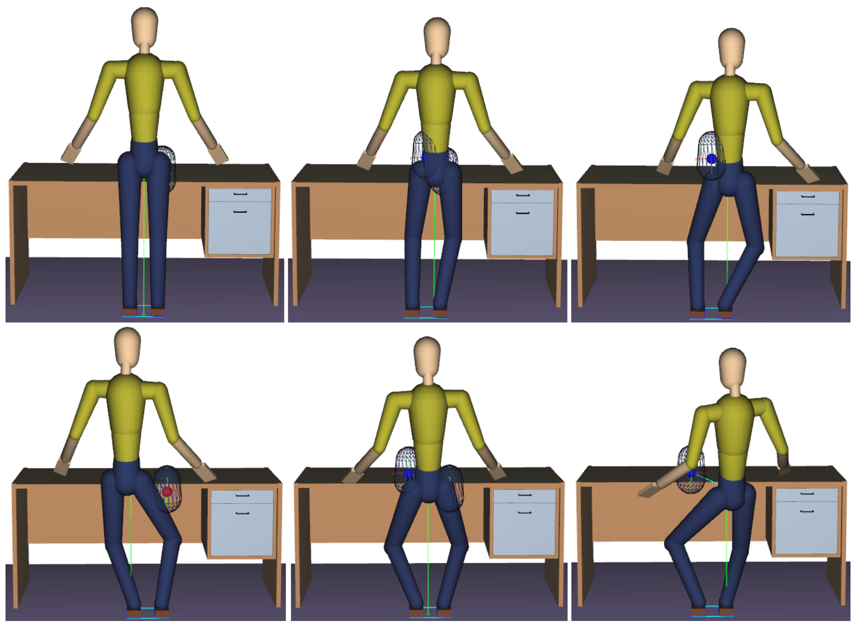

Fig. 6 Example of motions obtained with the balance skill. Top sequence: from standing pose to single foot support. Bottom sequence: transitioning the support mode from one foot to the other. The vertical line shows the projection of the center of mass to the floor. The intersection with the support polygon (also shown) reveals the support mode of the humanoid.

Note that if distant targets are given to the motion planner a long sequence of stepping and balance skills will be computed in order for the humanoid to achieve walking. The produced sequence would however not be specialized for walking. For achieving distant targets, a specialized walking skill could be included in the framework for providing optimized gait patterns for fast locomotion. Such a walking would be specifically implemented for the considered humanoid platform and would typically guarantee dynamic balance during the walking. The presented skills would then only be invoked when precise movement in constrained locations is needed.

Note also that the IK-based computations performed in the basic skills are specific to anthropomorphic limbs and cannot be easily extended to generic humanoid platforms. Generic Jacobian-based IK approaches [2, 10, 36] could be employed but would impose more expensive computations as they are iterative methods which require inversion of a Jacobian at every iteration. An alternate solution would be to employ fast *cyclic coordinate descent* iterations [65] for determining target poses and then generate the motions with interpolation in joint angle space.

3 Multi-Skill Planning

The main procedure of the multi-skill planner repeatedly invokes a skill expansion routine that will select and expand skills. The procedure stops when a solution motion is found, or if too much time has elapsed without success, in which case failure is reported. The pseudo-code for the main skill expansion routine is presented in Algorithm 1.

The multi-skill planner integrates in a single framework two types of search strategies depending on the skill type. If a skill σ is not explorative, k_2 samples from the parametric space of the skill are used to obtain k_2 motions entirely generated by σ without modifications. These motions are then included in a global discrete search tree T for continued expansion. As the examples illustrated in this work focus on arm manipulation, all mobility skills are set to be non-explorative. Therefore skills σ_{step} and σ_{bal} are only expanded by concatenation of sequences entirely generated by the skills.

Skill σ_{reach} is however set to be explorative and whenever a goal target is in reachable range, a bidirectional *RRT* exploration search tree is expanded by sampling intermediate configurations in the parametric space of σ_{reach}. One tree is rooted at the current state of the humanoid's arm, and the second tree is rooted at the goal arm state, which is computed by the skill. The bidirectional exploration determines if a motion in $\mathcal{C}_{free}$ between both states can be achieved and is set to explore only until k_1 nodes are achieved in the bidirectional search trees. If this limit is reached without finding a solution, the tree exploration is suspended and its source state re-inserted in the expansion front queue Q, allowing the expansion to possibly continue in the future. In this way, different sequencings of mobility and manipulation explorations are able to compete in search for the best compromise solution.

3.1 Algorithm Details

The Expand_Skill procedure basically selects skills and performs the search strategy suitable for each skill. It maintains a tree T storing all states traversed so far and a priority queue Q containing the states in the current expansion front. At each call, the procedure starts by removing the lowest cost (higher priority) state from Q and selecting all skills which are applicable (line 2), i.e. which can be instantiated at the current state x. For each applicable skill, the corresponding expansion method is selected and applied.

The priority queue Q store values according to a cost metric f_{cost} which can include different characteristics for guiding the search. The used cost function mainly 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 to the current node x. Edge costs represent the length of the motions represented in 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. Finally, term weighted by w_e 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 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 $\sigma^x$ in $\mathcal{S}(x)$ ) do
 4.   if ( $\sigma^x$ type is manipulation and $p_g$ in range ) then
 5.     $x_g \leftarrow \sigma^x(p, 1)$
 6.     if ( $x_g$ successfully generated by $\sigma^x$ ) then
 7.       grow_bidirectional_search ( $x$, $x_g$, $k_1$, $\sigma^x$ )
 8.       if ( connection found ) then
 9.         return SOLUTION_FOUND
10.       else
11.         attach the expanded bidirectional trees to $x$
12.         $n \leftarrow$ total number of nodes in the trees
13.         $Q$.insert ( $x$, $f_{cost}(x, p_g, n)$ )
14.       end if
15.     end if
16.   else
17.     // $\sigma^x$ is of mobility type
18.     for ( $k_2$ times ) do
19.       $p \leftarrow$ sample ( $P_\sigma$ )
20.       if ( motion generated by $\sigma^x(p)$ is valid ) then
21.         $x' \leftarrow \sigma^x(p, 1)$
22.         $T$.append_child ( $x$, $x'$, $\sigma^x$, $p$ )
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
```

Note also that the planner will request many sample motions from the available skills in order to progress with the search for a solution. Figure 7 illustrates some of the obtained motion samples from each of the skills.

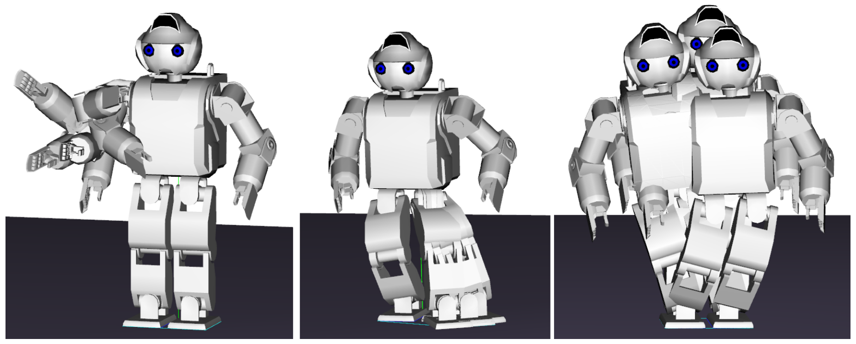

Fig. 7 Example of end postures from sampled motions. Motions are sampled from each of the considered skills: reaching (left), stepping (center), and balance (right). The HOAP-3 versions of the skills are used in this example.

In summary, the overall algorithm is able to integrate discrete skill expansion of mobility skills with configuration space exploration of manipulation skills in a single framework. The approach is able to naturally solve the trade-off between performing difficult manipulations or re-adjusting the body placement with mobility skills.

3.2 Results and Discussion

Figure 8 shows the coordination obtained by the algorithm for solving a given reaching problem. The first image (a) shows that the goal is not in the reachable range of the arm. The next two images (b,c) show the initial and final postures produced by the balance skill from the initial standing posture to a (non-explorative) sampled body posture favoring approximation to the hand target. The reaching skill is then recruited and instantiated at this posture and a bidirectional exploration tree is expanded (image d) for connecting the current posture to a posture reaching the target. The solution arm motion found by the reaching skill successfully avoids collisions with the table and reaches the goal, as depicted in the bottom sequence (d-f).

Further examples are illustrated in Figures 9, 10, 11 and 12. In all examples, the obtained solutions represent valid collision-free motions.

Note that the RRT exploration trees are implemented only using the generic interface for controlling motion skills. Therefore every expansion toward a sampled landmark implies a re-instantiation and application of the skill such that each expansion motion is produced by the skill itself.

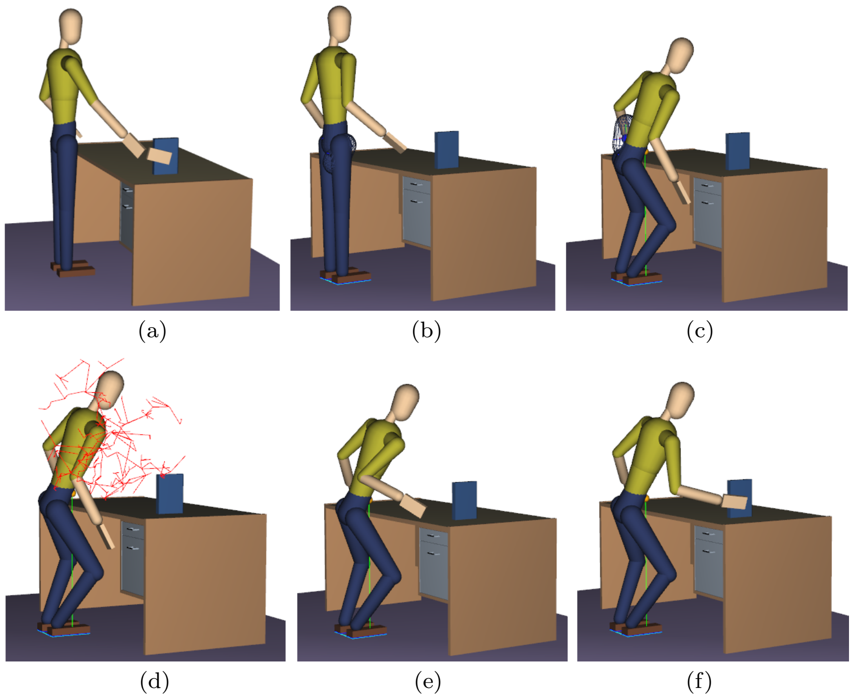

Fig. 8 Example of an obtained sequencing of stepping, balance and reaching skills. The lower left image (d) shows the generated exploration tree which was able to find the collision-free motion produced by the reaching skill.

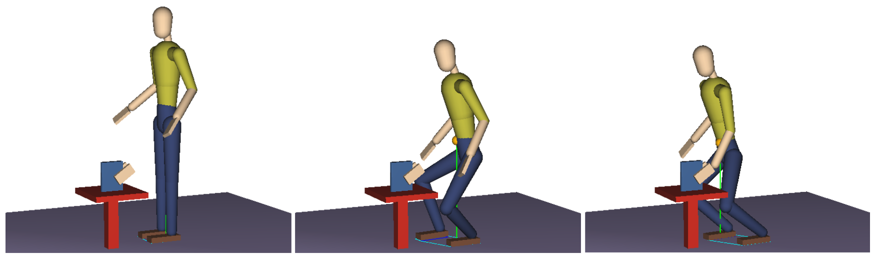

Fig. 9 Example of a solution coordinating mobility and manipulation skills in order to reach for a low target.

In its current version the algorithm is limited to sequencing motion skills. A possible extension is to allow a skill to be activated in parallel with the previous skill, for instance to allow reaching to start while stepping or balance is still being executed. These concurrent executions can also be computed as a post-optimization phase in order to reduce the time required to plan the motions. In most of the presented examples, the computation time taken by

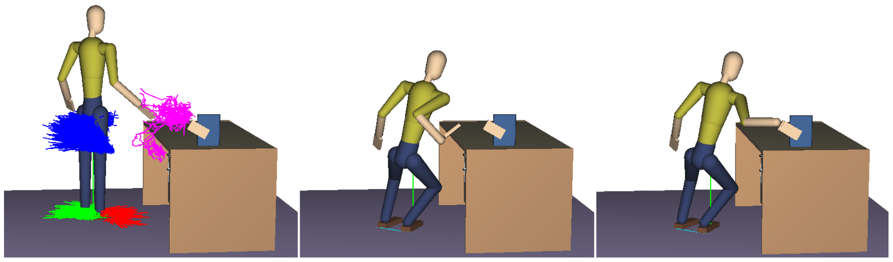

Fig. 10 In this example, a large k_1 value was used allowing many expansions in the bidirectional searches and leading to a relatively complex reaching motion being found. By using smaller values for k_1, more mobility skills would be recruited favoring reaching solutions needing fewer bidirectional expansions to reach the goal. The colored trajectories illustrate the locations traversed by the joints controlled by each skill instantiated in T.

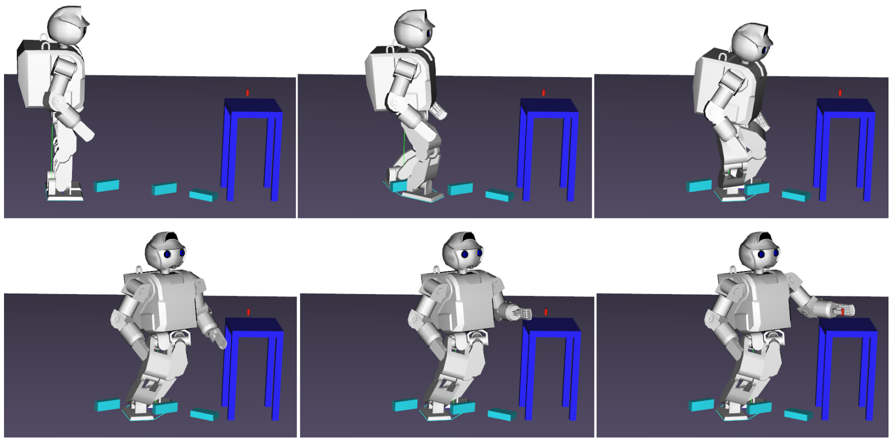

Fig. 11 Example of a solution computed for the HOAP-3 model which requires several steps among obstacles until the target can be reached.

the planner was in the range of 10 to 40 seconds. Solutions of longer duration, as the one shown in Figure 11, require several minutes of computation.

In the case of applying planned motions to the HOAP-3 humanoid platform (Figure 12) we also rely on a reactive controller running in the robot with the purpose to correct the main torso orientation in order to maximaze balance stability in response to readings from the feet pressure sensors. The several presented results demonstrate the capabilities of the multi-skill planning algorithm.

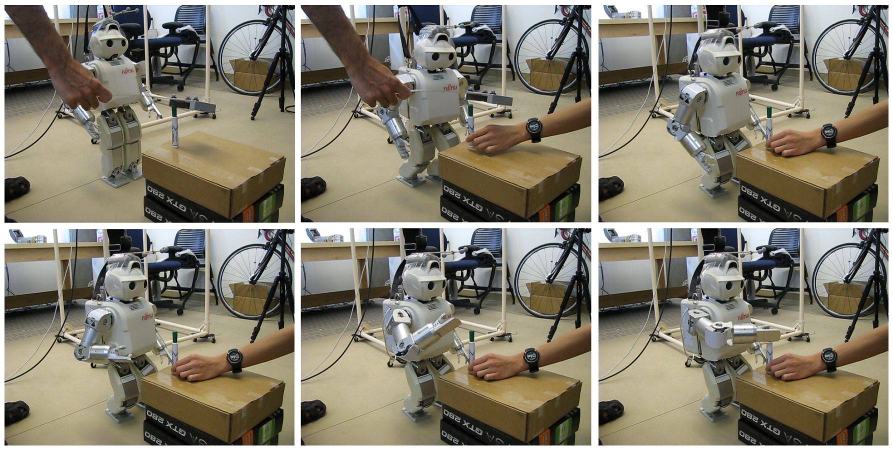

Fig. 12 Example of applying a planned motion to the HOAP-3 humanoid. The target location to reach is specified by markers tracked with motion capture cameras (first two images). The first three snapshots (top row) show the few steps planned for approaching the target, and the final three snapshots (bottom row) show the arm motion for finally reaching the target.

4 Learning

Learning is essential for improving motion planning. In our framework (Figure 3) learning starts by analyzing executed tasks in order to extract useful information for improving individual skills and as well for improving the overall planning process. Learned information may be directly sent to individual skills or may be stored in the task memory database, which will be accessible to each skill and as well to the multi skill planner.

We describe now our first experiments in designing such learning algorithms with an overview of the learning strategies employed in our *Attractor-Guided Planning* (AGP) approach for reaching tasks [29]. It includes a simple task analysis module for selecting previous tasks to be reused with attractor points which guide the planning of similar subsequent reaching motions.

The task analysis metric and the extraction of attractor points are discussed in the next sections and provide a good example of features which can be learned specifically for improving the exploration search of a reaching skill.

Finally, we also present an approach for extracting motion constraints from motion capture examples. Constraints will for instance inform that a certain object should be relocated only with specific types of motions. As a result, an imitation-based approach for learning motion constraints is achieved.

4.1 A Similarity Metric for Reaching Tasks

A similarity metric is needed for identifying previously planned reaching tasks which are similar to a new task to be solved. Consider the case of comparing the query task T with a previous task T' from the task memory database. Task T' is considered reusable only when its local environment and solution motion will share similar characteristics. The main question is what information should be taken into account. One possible approach is to define a metric only based on the distances between the initial and goal configurations of the tasks. Such a naive metric works well with dynamic queries in a static environment, where all the obstacles are not moving. When it comes to a dynamic environment, the change of obstacle positions may largely affect the structure of a solution path, as illustrated in Figure 13. This motivates us to include obstacle positions in the task comparison metric.

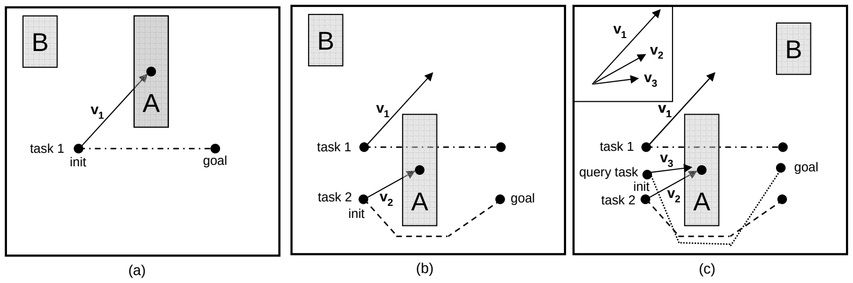

Fig. 13 A planning environment with moving obstacles A and B. Task 1 and task 2 are previously solved (image a and b), and the query task is selecting an example from the database. v1, v2 and v3 denote the local coordinates of obstacle A with respect to the three initial configurations. Note that obstacle B is out of the local coordinate system range, thus is not included in the comparison metric. Although the initial and goal configurations of task 1 are closer, the query task chooses task 2 as example because v2 is closer to v3.

When considering obstacles, one important factor is how much influence an obstacle has on the solution path. In Figure 13, obstacle B shows a great change of position from task 2 to the query task (images b,c), but since this change has limited impact on the solution path, this obstacle should not influence the comparison. Our comparison technique represents obstacles in local coordinates in respect to the initial and goal configurations.

For each task, two coordinate systems are built with origins respectively located at the initial and goal configurations. Obstacles that are too far away from an origin are not included in the respective coordinate system. The comparison metric accumulates the distance between each obstacle o from the query task and o' from the example task that is closest to o, computed in their local coordinates. The metric is described as follows:

$$dist(T,T') = h_1 * dist(c_{init}, c'_{init}) + h_1 * dist(c_{goal}, c'_{goal}) \\ +h_2 * \sum_{o'} \min_o dist(o,o').$$

The weights of the motion query distance and the obstacle distance are tuned by heuristics h_1 and h_2.

4.2 Learning Reaching Strategies

Once a previous similar task is identified from the task memory base, the stored motion is used to guide the planning of the new task at hand. Attractor points extracted from the previous motion are used. Since the focus here is on reaching tasks, we fit the path of the humanoid wrist approximately into a sequence of piecewise linear segments, and then define attractors to be the points where two segments connect. Given a set of 3D points denoting the path of the wrist joint in workspace, we employ a variation of a line tracking algorithm [61], which was further experimented and modified by Nguyen and colleagues [52].

The modified line tracking algorithm starts by constructing a window containing the first two points, and fits a line to them. Then it adds the next point to the current line model, and recomputes the new line parameters. If the new parameters satisfy the line condition up to a threshold, the window incrementally moves forward to include the next point. Otherwise, the new included data point is detected as an attractor point, and the window computation restarts from this attractor.

After a list of attractors is detected, a validation procedure is used to ensure that the attractor list defines a valid collision-free path. If a collision is detected between two consecutive attractors, the middle point on the original path is inserted in the attractor list and the process iterates until all of the straight segments are free of collisions. The goal is to detect valid attractors even in difficult regions. If the sequence of valid attractors is applied to guide the exact same task again, the solution path will be trivially solved without any search outside the guided path.

Once a set of attractors is decided to be reused, the sampling strategy in the motion planner is biased toward regions around the attractors. Taking advantage of the attractors is beneficial for two main reasons: first, it highly preserves the structure of a successful solution path. Second, whenever a new obstacle location collides with part of the solution path, a large portion of the path is still reusable. Note that since only similar tasks are used by the query, the environment does not differ much.

During the planning, dynamic Gaussian distributions are placed around each attractor point in the configuration space. The distributions are used as a replacement to the random sampling procedure of the exploration trees in the planner. The scale of the Gaussian distribution is initialized as zero, which means samples start as exactly the attractor point, guiding the search directly toward the current attractor. When samples are not able anymore to attract the exploration tree due variations in the new task, the scale of the Gaussian distribution is increased. In the limit the bias will gradually disappear and the sampling will return to a random sampling. When samples get very close to the attractor, the Gaussian centered on the next attractor point is then used. The process repeats for the whole list of attractors until a solution is found or the task is considered not solvable.

The same procedure can be applied to single or bidirectional search trees, in which case opposite attractor orders are taken in each tree (see Figure 14). Several experiments were conducted and they demonstrate a significant improvement in computation speed for dynamic environments which maintain some structure between queries [29].

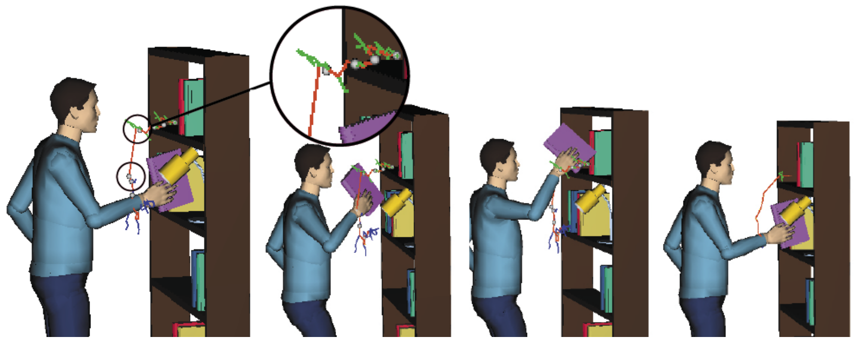

Fig. 14 Attractors (gray spheres) are extracted from a path (red line) computed by a bidirectional RRT tree expansion (green and blue branches). The attractors specifically represent postures which are trying to avoid collisions, as shown in the second and third images. The Attractor-Guided Planning results in less tree expansions and in a smoother solution path (last image) [29].

4.3 Learning Constraints from Imitation

Different kinds of motion constraints are important in object manipulation, for example: changes in orientation should not be allowed when relocating a cup of water. Moreover, the pattern of a movement is also meaningful, for example, whether the hand of the character is moving along a straight line on top a surface or just moving unconstrained in space.

Note that it is assumed here that an object being manipulated (or relocated) has a fixed attachment to the hand of the humanoid, and therefore finger manipulations are not considered, meaning that the hand and the object can be considered to be one single end-effector being controlled.

Motion constraints of objects or tasks have to be somehow programmed in the humanoid. We explore the approach of learning constraints from direct demonstrations, i.e. from imitation. The idea is that users can demonstrate how tasks should be performed and the observed motions are then analyzed and learned. We consider that a demonstrated motion is captured and represented as the trajectory of the user's hand. The demonstrated trajectory can then be analyzed and classified. A classification of the considered types of motion constraints is given below:

- Stationary: when both the position and the orientation of the end-effector remains the same. This happens for instance when holding still an object (like a photograph camera) while the body may be moving. Note that constraints are detected in global coordinates.
- Translation-only: when only the orientation of the end-effector remains the same. This will result in a motion only composed of translations (for example to relocate a cup of water).
- Rotation-only: when only the translation of the end-effector remains the same. This constraint happens when no translations are present.
- Patterned-rotation: this constraint refers to cases where the end-effector is rotating around an axis or a point. This constraint appears in particular rotational manipulations (like when opening a door).
- Patterned-translation: for the cases where the end-effector is translating in a plane or along a straight line. This constraint appears in particular translational manipulations (like painting a straight line).

The first three types of constraints are basic constraints. The stationary constraint has to be detected first. The translation-only constraint can exist at the same time with a patterned-translation constraint, but not with a patterned-rotation constraint. Similarly, rotation-only constraints can exist at the same time with patterned-rotation constraints, but not with patterned-translation constraints. Based on these observations, Figure 15 shows a decision tree for detecting constraints based on a series of tests.

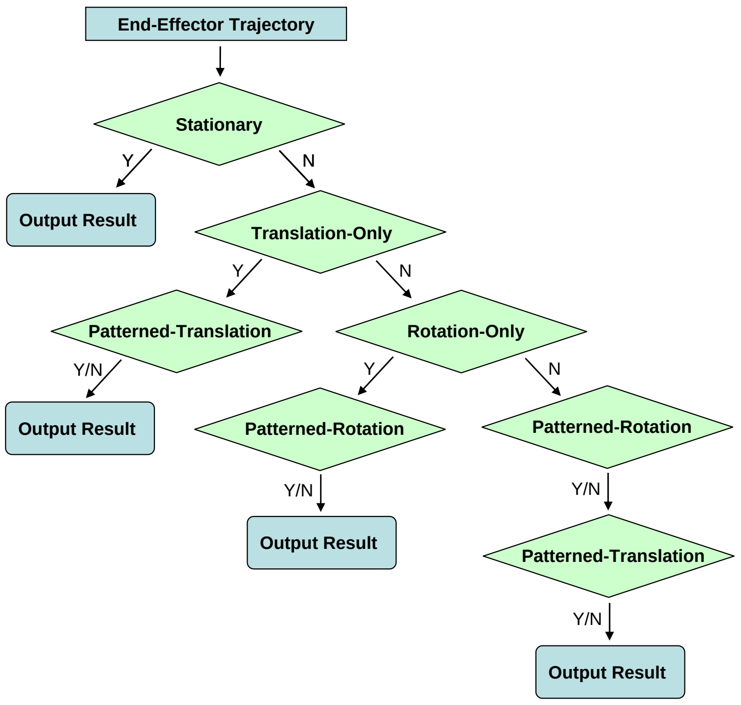

Fig. 15 Decision stages for detecting constraints. Letters Y or N denote whether the constraints can be detected or not.

A constraint detection algorithm can then be devised for analyzing the demonstrated motion. Given a specific time interval $[t_1, t_n]$, the demonstrated motion m of the end-effector is denoted as a sequence of captured frames $(f_1, f_2, ..., f_k, f_{k+1}, ..., f_n)$. Each frame f_k contains geometric information (position and orientation) about the end effector at that time step, i.e., the position and orientation of the end-effector with respect to the global coordinate system. Instantaneous constraints are first detected between each pair of frames. Then, the sequence of instantaneous constraints will be merged so that motion m is segmented by constraint type.

4.3.1 Detection of Instantaneous Constraints

An instantaneous constraint C_k is detected between time k and $k+1$. Given a pair of frames f_k and f_{k+1}, the definition of C_k depends on the 3D transformation from time k to time $k+1$, which is represented here by the displacement matrix D_k. Therefore if x_k denotes the demonstrated hand position at time k, then $x_{k+1} = D_k x_k$.

If the motion is stationary from time k to $k+1$, then D_k will be the identity transformation and $(D_k - I_4)x_k = 0$. Since the solutions to this equation include all the observed points on the end-effector, the elements in matrix (D_k-I_4) will contain very small values. Note that the translational values encoded in the homogeneous matrices have to be scaled to a compatible unit in respect to the rotational values, which are inside [-1,1].

The squared sum of all the elements in matrix (D_k-I_4) is then computed and called to be the *stationary value* of constraint C_k. This value denotes how much movement the end-effector has performed during this time step. If this value is smaller than a stationary threshold τ_s, then the end-effector is considered stationary from time k to time $k+1$.

Rotational movements can be similarly detected. First, as the end-effector can only perform rigid transformations, D_k can be re-written as:

$$D_k = \begin{bmatrix} R_k & t_k \\ 0_3 & 1 \end{bmatrix}$$

Where R_k and t_k are the rotational and translational components of D_k. The matrix (D_k-I_4) can then be written as:

$$D_k - I_4 = \begin{bmatrix} R_k - I_3 & t_k \\ 0_3 & 0 \end{bmatrix}$$

The squared sum of all the elements in matrix (R_k -I_3) is then called the *rotational value* of C_k. If it is smaller than a rotational threshold τ_r, then the movement will be identified as translation-only, i. e. with no rotation involved.

Similarly, for a rotation-only movement during one time step, the constraint can be detected by comparing the squared sum of vector t_k (the *translational value*) with a translational threshold τ_t.

In summary, the first three types of constraints are detected as follows:

- For an instantaneous constraint C_k, if its stationary value minus the squared sum of all the elements in (D_k-I_4) is smaller than threshold τ_s, then the end-effector is stationary between time k and $k+1$.
- For an instantaneous constraint C_k, if its rotational value minus the squared sum of all the elements in (R_k-I_3) is smaller than threshold τ_r, then the end-effector is not rotating between time k and $k+1$.
- For an instantaneous constraint C_k, if its translational value minus the squared sum of all the elements in t_k is smaller than threshold τ_t, then the end-effector is not translating between time k and $k+1$.

To detect the other two constraints (patterned-rotation and patterned-translation), successive transformations have to be merged and tested if they have similar patterns.

4.3.2 Merging Transformations

Transformation R_k denotes the rotational component of the transformation between time k and $k+1$. The rotation axis $d_k = (r_{21}$-r_{12}, r_{02}-r_{20}, r_{10} - $r_{01})$ can then be determined for each frame where r_{ij} denotes the elements of matrix R_k. Given a sequence of frames (f_1, f_2, ..., f_k, ..., f_n), the rotational axes (d_1,d_2, ...,d_k,d_{k+1}, ...,d_n) can be then computed. If they can be approximately fitted by a straight line L, then the motion is a rotation around the line L, and the transformations between time 1 and n can be merged together.

Similarly, if all the translational vectors can be approximately fitted to a straight line, then the end-effector will be translating along the line. Two new thresholds τ_{rfit} and τ_{tfit} are needed for deciding the rotational and translational fitting limits. The fitting algorithm employed is the same as the one used to detect attractors for the AGP planner (Section 4.2). Note that the same procedures can also be applied to detect translations in a plane.

Since real motion capture data is being analyzed, the existence of noise or performance imprecisions will create outliers, i.e., one or more frames that divide the motion into two parts, which should have the same constraint. To handle outliers, a frame buffer is used to store aberrant frames with a different motion pattern than in previous frames. A new motion will be created only when the number of aberrant frames in the frame buffer reaches a limit. Two limits are needed:

- Rotation buffer size: this threshold decides if a patterned rotation has ended and it is denoted as τ_{rbuf}. When the number of inconsistent frames in a patterned rotation is greater than τ_{rbuf}, a new motion constraint will be created.
- Translation buffer size: this threshold decides if a patterned translation has ended and it is denoted as τ_{tbuf}. When the number of inconsistent frames in a patterned translation is greater than τ_{tbuf}, a new motion constraint will be then created.

After a set of constraints are detected, they need to go through a final filter in order to ensure that the motion does not contain unnecessary segments. Two constraints are then merged together if they belong to the same constraint category and are close enough to each other.

4.3.3 Computing the Thresholds

A number of threshold parameters for detecting and merging constraints have been used: $\tau_s, \tau_r, \tau_t, \tau_{rfit}, \tau_{tfit}, \tau_{rbuf}, \tau_{tbuf}$. In order to perform a robust computation of the constraints, these values will be determined from template motions which can be provided for calibrating the constraint detection. Three types of calibration motions are needed: a stationary motion m_1, a rotation-only motion m_2 around an axis (and without translation), and a translation-only motion m_3 (without rotation) along a straight line.

From motion m_1 it is possible to determine the stationary threshold τ_s. First all stationary values for the frames of the motion are computed: $V_s = (\sigma_1, \sigma_2, ..., \sigma_k, \sigma_{k+1}, ..., \sigma_n)$. Due to the presence of noise and imprecisions from the human operator during the motion capture process it is not correct to simply take the highest σ value and assign it as the stationary threshold. There might be outliers in V_s as for instance a subtle shake of the hand could lead to a very high σ value.

In order to reduce the overestimation of the stationary threshold, outliers have to be detected and rejected. First, the median number σ_m from set V_s is computed. Then for each $\sigma_k \in V_s$, the squared distance $dist_k = (\sigma_k - \sigma_m)^2$ is computed. Let the smallest squared distance be $dist_{min}$. A σ_k is detected to be an outlier if $dist_k > h * d_{min}$, where h is a heuristic weight which is greater than 1. After the outliers are removed from the set of the stationary values, the stationary threshold τ_s is set to be the highest stationary value from the filtered V_s set.

The rotational threshold and the translational threshold are similarly computed. Since motion m_2 is a rotation-only motion without translation, most of its instantaneous constraints should be lower than the translational threshold τ_t. First, the translation vectors $(t_1, t_2, ..., t_k, t_{k+1}, ..., t_{n-1})$ are computed from each of the frames. After removing the outliers, τ_t is assigned the highest translational value. The rotational threshold τ_r can then be obtained from motion m_3 in the same way.

The line fitting thresholds τ_{rfit} and τ_{tfit} can also be obtained from m_2 and m_3. Since motion m_2 is a rotation around a specific axis, all detected instantaneous rotational axes can be fitted to an approximate straight line. After removing outliers, τ_{rfit} is set as the largest distance between the instantaneous rotational axes and the fitted line. The computation of τ_{tfit} is similarly performed from motion m_3. Finally, the maximum number of continuous outliers can then be assigned to τ_{rbuf} and τ_{lbuf}.

4.3.4 Reusing Detected Constraints in New Tasks

Object manipulation tasks are often complicated and will typically be composed of multiple motion constraints. For example, a water pouring task will typically include a translation without rotation followed by a rotation without translation. The described constraint segmentation procedures will divide the motion into a sequence of constrained sub-motions according to their constraint types. The connections between these sub-motions can then be represented by constrained attractors which will guide the execution of similar tasks, for example for pouring water in similar environments or to different target locations.

The attractor-guided planning described in Section 4.2 is then employed with the extension that configurations will always be sampled respecting the constraints associated with attractors. In this way constrained attractors will be able to guide the planning of future tasks which will respect constraints to be imitated. Even in a changing environment the constraints of the original motion will still be maintained.

4.4 Results and Discussion

A FASTRAK magnetic motion capture system was used for capturing example hand motions with constraints. Although a high frame rate is not necessary, the system is able to capture motions at 60Hz. First, template motions were processed in order to determine the needed thresholds and then several motion constraints could be successfully detected.

Figure 16 illustrates segmentation results of demonstrations containing different types of motions: 1) two translations, 2) translation and rotation, 3) translation, rotation and again translation, and 4) translation and random movements.

Each segment is shown as a pair of attractors (gray spheres). A segment represents a motion with a different constraint type than its neighbor segments. The described constraint detection procedure is capable of detecting all constraints. Stationary motions are also detected and capture a natural stop when the performer switches between translations and rotations.

Figure 17 shows the AGP planner being used to solve the constrained manipulation task of relocating a cup without rotating it. The demonstrated motion includes only translation. After recording the demonstration, an obstacle is placed inbetween the demonstrated path in order to force the imitation procedure to find an alternate path. The AGP approach is then able to find

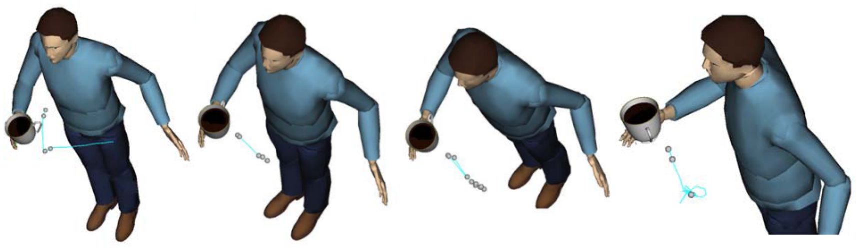

Fig. 16 The segmentations of four motions with different types of constraints. The attractors (denoted by gray spheres) represent a constraint type change and can then be used to guide the planning of future similar tasks.

a suitable solution maintaining the main characteristic of the demonstrated motion: the cup is never allowed to rotate during the relocation.

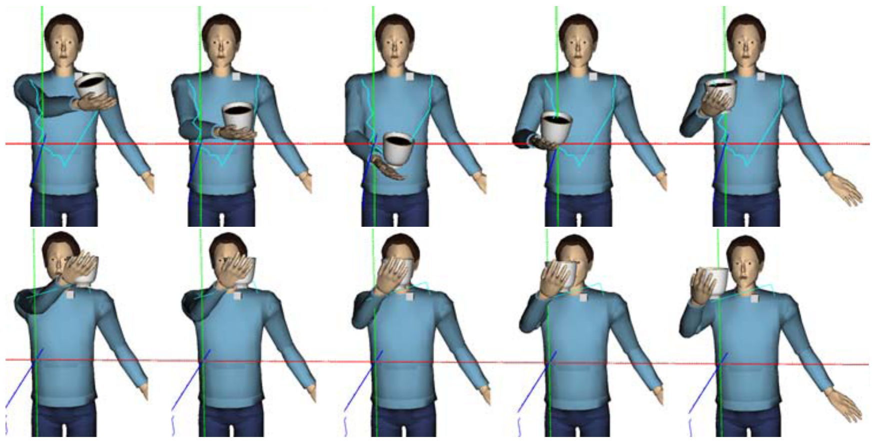

Fig. 17 The top sequence shows a solution path found by a non-constrained RRT planner. The bottom sequence shows a solution path found by the AGP planner with attractors constrained to translational motion. The small cube in the environment represents an obstacle and demonstrates the adaptation of the solution respecting the constraints. The image also illustrates that, due the much more constrained search procedure, AGP also favors finding shorter solutions.

5 Conclusion

We have described several pieces of a framework integrating learning with motion planning in different ways. Several results were also presented. We believe that the proposed approach is able to address a wide range of humanoid applications requiring autonomous object manipulations.

The presented framework also naturally accounts for the possibility of including new motion skills which could be automatically learned from experiences and then added in the skill base in order to provide new specialized skills to be used by the multi-skill motion planner. The possibility of achieving an automatic open-ended learning structure for accumulating motion skills and object-related constraints has the potential to empower the multi-skill planner with the capability to solve generic and increasingly complex tasks. The proposed framework presents our first steps in this direction.

Acknowledgments: This work has been partially supported by NSF Awards IIS-0915665 and BCS-0821766.

References

[1] M. A. Arbib. Perceptual structures and distributed motor control. In V. B. Brooks, editor, *Handbook of Physiology, Section 2: The Nervous System Vol. II, Motor Control, Part 1*, pages 1449–1480. American Physiological Society, Bethesda, MD, 1981.

[2] P. Baerlocher. *Inverse Kinematics Techniques for the Interactive Posture Control of Articulated Figures.* PhD thesis, Swiss Federal Institute of Technology, EPFL, 2001. Thesis number 2383.

[3] P. Baerlocher and R. Boulic. Parametrization and range of motion of the ball-and-socket joint. In *Proceedings of the AVATARS conference*, Lausanne, Switzerland, 2000.

[4] 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.

[5] A. Billard and M. J. Matarić. Learning human arm movements by imitation: Evaluation of a biologically inspired connectionist architecture. *Robotics and Autonomous Systems*, 37:2-3:145–160, November, 30 2001.

[6] C. Breazeal, A. Brooks, D. Chilongo, J. Gray, G. Hoffman, C. Kidd, H. Lee, J. Lieberman, and A. Lockerd. Working collaboratively with humanoid robots. In *Proceedings of Humanoids*, Los Angeles, CA, 2004.

[7] C. Breazeal, D. Buchsbaum, J. Gray, D. Gatenby, and D. Blumberg. Learning from and about others: Towards using imitation to bootstrap

the social understanding of others by robots. *Artificial Life*, 11:1–2, 2005.

[8] 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.

[9] B. Burns and O. Brock. Sampling-based motion planning using predictive models. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, pages 3120– 3125, Marina del Rey, CA, April 18-22 2005.

[10] S. R. Buss and J.-S. Kim. Selectively damped least squares for inverse kinematics. *Journal of Graphics Tools*, 10(3):37–49, 2005.

[11] J. Chestnutt, M. Lau, K. M. Cheung, J. Kuffner, J. K. Hodgins, and T. Kanade. Footstep planning for the honda asimo humanoid. In *Proceedings of the IEEE International Conference on Robotics and Automation*, April 2005.

[12] 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 (Humanoids)*, 2007.

[13] B. Dariush, M. Gienger, B. Jian, C. Goerick, and K. Fujimura. whole body humanoid control from human descriptors. In *Proceedings of the International Conference on Robotics and Automation (ICRA)*, pages 3677–2684, May 19-23 2008.

[14] J. Decety. Do imagined and executed actions share the same neural substrate? *Cognitive Brain Research*, 3:87–93, 1996.

[15] 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.

[16] M. A. Diftler and R. O. Ambrose. Robonaut, a robotic astronaut assistant. In *International Symposium on Artificial Intelligence, Robotics and Automation in Space (ISAIRAS)*, Montreal Canada, June, 18 2001.

[17] 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.

[18] C. Esteves, G. Arechavaleta, J. Pettré, and J.-P. Laumond. Animation planning for virtual characters cooperation. *ACM Transaction on Graphics*, 25(2):319–339, 2006. ISSN 0730-0301.

[19] P. Faloutsos, M. van de Panne, and D. Terzopoulos. Composable controllers for physics-based character animation. In *Proceedings of SIGGRAPH*, pages 251–260, New York, NY, USA, 2001. ACM Press. ISBN 1-58113-374-X.

[20] S. F. Giszter, F. A. Mussa-Ivaldi, and E. Bizzi. Convergent force fields organized in the frog's spinal cord. *Journal of Neuroscience*, 13(2):467–491, 1993.

[21] A. Goldman and V. Gallese. Mirror neurons and the simulation theory of mind-reading. *Trends in Cognitive Science*, 2(12):493–501, 1998.

[22] S. Grassia. Practical parameterization of rotations using the exponential map. *Journal of Graphics Tools*, 3(3):29–48, 1998.

[23] D. B. Grimes, D. R. Rashid, and R. P. N. Rao. Learning nonparametric models for probabilistic imitation. In *Neural Information Processing Systems (NIPS)*, pages 521–528.

[24] H. Hauser, G. Neumann, A. J. Ijspeert, and W. Maass. Biologically inspired kinematic synergies provide a new paradigm for balance control of humanoid robots. In *Proceedings of the IEEE-RAS International Conference on Humanoid Robotics (Humanoids)*, 2007.

[25] K. Hauser, T. Bretl, and J. Latombe. Non-gaited humanoid locomotion planning. In *Proceedings of the IEEE-RAS International Conference on Humanoid Robotics (Humanoids)*, pages 2641– 2648, December 2005.

[26] 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.

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

[28] J. K. Hodgins, W. L. Wooten, D. C. Brogan, and J. F. O'Brien. Animating human athletics. In *Proceedings of ACM SIGGRAPH*, pages 71–78, New York, NY, USA, 1995. ACM Press. ISBN 0-89791-701-4.

[29] X. Jiang and M. Kallmann. Learning humanoid reaching tasks in dynamic environments. In *Proceedings of the IEEE International Conference on Intelligent Robots and Systems (IROS)*, San Diego CA, 2007.

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

[31] 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.

[32] M. Kallmann. Analytical inverse kinematics with body posture control. *Computer Animation and Virtual Worlds*, 19(2):79–91, 2008.

[33] M. Kallmann, A. Aubel, T. Abaci, and D. Thalmann. Planning collision-free reaching motions for interactive object manipulation and grasping. *Computer graphics Forum (Proceedings of Eurographics'03)*, 22(3):313–322, September 2003.

[34] 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.

[35] 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.

[36] C. A. Klein and C.-H. Huang. Review of pseudoinverse control for use with kinematically redundant manipulators. *IEEE Transactions on Systems, Man, and Cybernetics*, SMC-13(3):245–250, March-April 1983.

[37] 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.

[38] S. Koenig and M. Likhachev. Real-time adaptive a*. In *Proceedings of the International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS)*, pages 281–288, 2006.

[39] 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. ISBN 0-89791-667-0.

[40] 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.

[41] 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.

[42] K. Kurashige, T. Fukuda, and H. Hoshino. Motion planning based on hierarchical knowledge for a six legged locomotion robot. In *Proceedings of IEEE International Conference on Systems, Man, and Cybernetics*, volume 6, pages 924–929, 1999.

[43] J.-C. Latombe. *Robot Motion Planning*. Kluwer Academic Publisher, December 1990. ISBN 0-79-239129-2.

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

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

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

[47] 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.

[48] V. Manikonda, P. Krishnaprasad, and J. Hendler. A motion description language and hybrid architecure for motion planning with nonholonomic robots. In *Proceedings of IEEE International Conference on Robotics and Automation (ICRA)*, May 1995.

[49] M. J. Matarić. Socially assistive robotics. *IEEE Intelligent Systems*, August 2006.

[50] J. McCann and N. S. Pollard. Responsive characters from motion fragments. *ACM Transactions on Graphics (SIGGRAPH 2007)*, 26(3), Aug. 2007.

[51] M. Morales, L. Tapia, R. Pearce, S. Rodriguez, and N. Amato. A machine learning approach for feature-sensitive motion planning. In *Proceedings of the Workshop on the Algorithmic Foundations of Robotics*, 2004.

[52] V. T. Nguyen, A. Martinelli, N. Tomatis, and R. Siegwart. A comparison of line extraction algorithms using 2d laser rangefinder for indoor mobile robotics. In *International Conference on Intelligent Robots and Systems (IROS05)*, Edmonton, Canada, 2005.

[53] M. N. Nicolescu and M. J. Matarić. Natural methods for robots task learning: Instructive demonstration, generalization and practice. In *Proceedings of the 2nd International Joint Conference on Autonomous Agents and Multi-Agent Systems (AAMAS)*, Melbourne, Australia, 2003.

[54] A. Olenderski, M. Nicolescu, and S. Louis. Robot learning by demonstration using forward models of schema-based behaviors. In *Proceedings,*

International Conference on Informatics in Control, Automation and Robotics, pages 14–17, Barcelona, Spain, September 2005.

[55] J. Pettré, M. Kallmann, M. C. Lin, J. Kuffner, M. Gleicher, C. Esteves, and J.-P. Laumond. Motion planning and autonomy for virtual humans. In *SIGGRAPH'08 Class Notes*, 2008.

[56] A. Ramesh and M. J. Matarić. Learning movement sequences from demonstration. In *Proceedings of the International Conference on Development and Learning (ICDL)*, pages 302–306, MIT, Cambridge, MA, 2002.

[57] 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.

[58] S. Schaal, A. Ijspeert, and A. Billard. Computational approaches to motor learning by imitation. *The Neuroscience of Social Interaction*, 1431:199–218, 2003.

[59] L. Sentis and O. Khatib. A whole-body control framework for humanoids operating in human environments. In *Proceedings of the International Conference on Robotics and Automation (ICRA)*, pages 2641– 2648, May 15-19 2006.

[60] A. Shapiro, M. Kallmann, and P. Faloutsos. Interactive motion correction and object manipulation. In *ACM SIGGRAPH Symposium on Interactive 3D graphics and Games (I3D)*, Seattle, April 30 - May 2 2007.

[61] A. Siadat, A. Kaske, S. Klausmann, M. Dufaut, and R. Husson. An optimized segmentation method for a 2d laser-scanner applied to mobile robot navigation. In *Proceedings of the 3rd IFAC Symposium on Intelligent Components and Instruments for Control Applications*, 1997.

[62] W. Suleiman, E. Yoshida, F. Kanehiro, J.-P. Laumond, and A. Monin. on human motion imitation by humanoid robot. In *Proceedings of the International Conference on Robotics and Automation (ICRA)*, pages 2697–2704, May 19-23 2008.

[63] K. A. Thoroughman and R. Shadmehr. Learning of action through combination of motor primitives. *Nature*, 407:742–747, 2000.

[64] A. Treuille, Y. Lee, and Z. Popović. Near-optimal character animation with continuous control. In *Proceedings of ACM SIGGRAPH.* ACM Press, 2007.

[65] L. C. T. Wang and C. C. Chen. A combined optimization method for solving the inverse kinematics problems of mechanical manipulators. *IEEE Transactions on Robotics and Automation*, 7(4):489–499, 1991.