Reverse Control for Humanoid Robot Task Recognition

Sovannara Hak, Nicolas Mansard, Olivier Stasse, Jean Paul Laumond

***Abstract*—Efficient methods to perform motion recognition have been developed using statistical tools. Those methods rely on primitives learning in a *suitable space*, for example, the latent space of the joint-angle and/or adequate task spaces. Learned primitives are often sequential : a motion is segmented according to the time axis. When working with a humanoid robot, a motion can be decomposed into parallel sub-tasks. For example, in a waiter scenario, the robot has to keep some plates horizontal with one of its arms, while placing a plate on the table with its free hand. Recognition can thus not be limited to one task per consecutive segment of time. The method presented in this paper takes advantage of the knowledge of what tasks the robot is able to do and how the motion is generated from this set of known controllers, to perform a reverse engineering of an observed motion. This analysis is intended to recognize parallel tasks that have been used to generate a motion. The method relies on the task-function formalism and the projection operation into the null space of a task to decouple the controllers. The approach is successfully applied on a real robot to disambiguate motion in different scenarios where two motions look similar but have different purposes.**

***Index Terms*—Task recognition, task-function formalism, humanoid robot, inverse kinematic.**

I. Introduction

CURRENT promising developments of service robotics stimulate the research in human-robot interaction. In that context, understanding robot actions from observation is a challenge per se. While an intentional action originates at a planning level, its realization takes place in the real world via motions. How to recognize an action from observed motions? Defining methods to automatically recognize the goal pursued by a robot performing a given motion is a critical issue. If we consider mobile manipulators (e.g., PR2 robots), there is a clear separation between navigation functions and manipulation functions. The question of action recognition may be rather simple. Similarly, a humanoid robot can be divided in two distinctive parts, legs and upper body, which correspond to the navigation and manipulation functions. For example, consider the *Give me the purple ball* scenario [2] performed by the humanoid robot HRP-2 at LAAS-CNRS as shown in Fig. 1(a). To reach the assigned objective, HRP-2 decomposes its mission into elementary sub-tasks, each of them being addressed by a dedicated software module. For instance, to reach the ball, the robot has to walk to the ball. *Walking* appears as an elementary action that is a resource to solve the problem that is processed by a dedicated locomotion module.

Authors are with CNRS, Universtité de Toulouse, LAAS, 7, avenue du Colonel Roche, F-31400 Toulouse, France (`firstname.lastname@laas.fr`). This work was supported by a grant from the R-Blink Project, Contract ANR-08JCJC-0075-01. A preliminary version of this work has been published in [1].

However, in the second scenario (Fig. 1(b)), HRP-2 has to grasp the ball that is located between its feet [3]. To reach the objective, the robot has to step away from the ball and then grasp it. In this experiment there is no dedicated module in charge of *stepping*. *Stepping* is a direct consequence of *grasping*. The grasping action is totally embedded in the body, allowing the legs to naturally contribute to the action. Grasping appears as an embodied action generating a complex motion. Finally Fig. 1(c) introduces the purpose of this paper. In the case on the left side, the robot performs a single grasping task. In the case on the right side, the robot performs two grasping tasks in parallel. The ambiguity to distinguish both cases comes from the role played by the left arm. In the first case, the left arm contributes to the single grasping action by maintaining the balance of the robot. In the second one, the left arm performs another grasping task. Both motions are very similar.

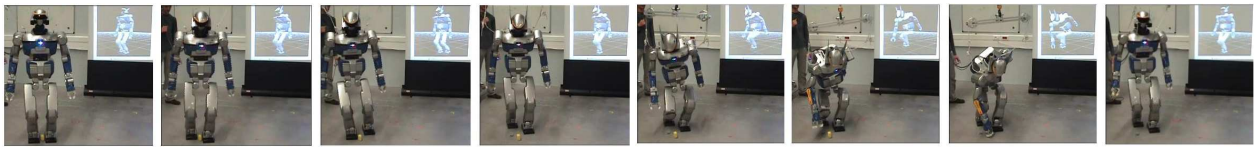
The works presented here tackles the problem of motion recognition and shows that it is possible to disambiguate both cases by focusing the analysis of the motion in the task spaces and on the behaviors of the controllers of the robot. The main assumption needed to disambiguate those cases is that the kinematic model of the observed robot has to be known. The experiments presented in this paper focus on the HRP-2 robot, but the method is generic and as long as the assumptions are respected, the method is theoretically valid for other robots.

II. Related work

Recognition spans a wide range of areas. In the vision community, this problem is generally looked from the unstructured motion point of view: no hypothesis are done even on the shape or rigidity of the moving body. The recognition can be applied to spot irregular events [4], or specific events such as visual indicators of drowning [5]. The recognition can also be focused on one person. The motion of the human can be analyzed in order to perform a human body tracking [6]. However, information from the environment is generally used and the recognition is mainly done from the context. For example, salient points in time and position (looking at the 2D video flow as a 3D function) [7]. These points are learned from a database, then matched during the demonstration. The environment is also used to perform a background extraction in order to extract a silhouette and perform, for example a gait recognition [8].

The structure of a moving multi rigid body is a strong assumption, that can boost the performances when this knowledge is available. It is often the case in the robotic context. For example the estimation of a humanoid pose trajectory in [9] is performed using optical motion-capture data to guide a known physical model. Recent vision developments prove that the

Scenario (a): The global task *Give me the ball* is decomposed into a sequence of sub-tasks [locate the ball], [walk to the ball], [grasp the ball], [locate the operator], [walk to the operator], and [give the ball]. The motions [walk to], [grasp], [give] appear as a sequence structuring the action (from [2]).



Scenario (b): To grasp the ball between its feet, the robot has to step away from the ball. In this experiment *stepping away* is not a software module. It is an integral part of the embodied action *grasping* (from [3]).

Scenario (c): To grasp the ball in front of it (left), the robot reaches a posture where the left arm is used to maintain its balance. In the figure on the right, the robot performs two actions in parallel: grasping a ball in front of it while grasping a ball behind (of course the ball behind has been intentionally placed at the end position of the left hand depicted on the left side). It is not possible to spot the difference between both postures. However, the question we address is: Is it possible to spot the difference between both *motions*?

Fig. 1. Introductory examples of embodied intelligence.

same knowledge could also be reliably be extracted using classical or RGB-D cameras [10]. In the remainder of this paper, we will consider that the motion of the rigid bodies is known. A typical use of a dedicated vision system to reconstruct the whole-body pose is presented in the experimental section.

The question is now how to extract information of higher levels from supposedly-known human whole-body movements. A direct example is signed languages which provide a way to convey meaning through the combination of hands, arms and face movement and configuration. In the same way, human gait has been studied from an information provider point of view. For example, the gait can be used like fingerprints to perform identification of human [11], [12], or to recognize emotions such as anger, sadness or happiness [13].

Statistics have been successfully applied to action recognition and motion analysis [14]. Statistical tools are used to create symbols, and by extension, detect those symbols in a motion. For example, a method for behavior-based control is proposed in [15], [16]. Behaviors are defined as a motion symbol (e.g. jab, hook, elbow, shield and uppercut). The behaviors are modeled by learning from series of examples. A dimensional reduction is then applied to have a significant clusterization. The recognition part is handled by a Bayesian classifier which recognizes a trajectory in joint or Cartesian space. The extension to the recognition is to perform an imitation. This is performed by interpolating known examples to obtain feasible trajectories. The introduction of partially-observable markov decision process or Bayesian inference [17] has renewed the topic of action modeling [18] in the last decade. Such techniques and related ones are now applied to motor skill learning in general [19], and to motion segmentation [20], [21] in particular. Hidden Markov Models (HMMs) have been extensively used, for example, to perform action and gait recognition [22] or to generate a human-like motions [23]. In these works, human capture data are used to build a basis elements in the joint space for each movement class using a dimension reduction technique. At the same time, HMMs are trained to capture the features of a movement class in the task space. The generation is obtained by finding the optimal linear combination of basis elements that maximize the probability of a trained HMM. Although the task space is considered to be the space where movement features have to be extracted, the method is limited to one specific task space per movement generated. In [24] variable-length markov models are used to learn atomic human actions. A sequence of atomic actions represent a complete behavior. Generally speaking, the efficiency of statistics-based recognition is ruled by the quality of the dataset built in the learning phase. Several demonstrations for each particular cases are needed in order to extract the invariants that will discriminate the tasks. The sparsity of the demonstrations can also limit the efficiency of the recognition. Finally, the sets of demonstrations have to be associated with the correct symbols, which are generally given to the learning algorithm.

Alternatively, recognition can be based on specific criteria that are a priori given to the system. In [25], only the robot trajectories are used to distinguish between various phases of motion. A task is a complete whole body motion within a temporal segment. The global motion is a sequence of tasks. Each task has its own parameters called *skills parameters*. The task recognition method is decomposed in two steps: first, for each tasks, find all the temporal segments in the observed motion corresponding to that task. The second step is the estimation of the skill parameters for each segment. Each task is detected by the analysis of a trajectory projected in a specific space. For example, a stepping task is detected by

analyzing the trajectory of a foot; a squatting task is detected by analyzing the vertical trajectory of the waist. The criteria used for detection and the associated dedicated projection spaces are built manually for a particular motion that has to be imitated by the robot. Similarly, [26] uses a set of specific spaces in which the observed motion is projected. Each task is associated with a specific criterion. These criteria are used to automatically choose the set of task spaces that will best represent and generalize a given movement, in order to focus a learning technique into that new space. The criteria used for the task-space selection are expressed by some score functions inspired from neuroscience: the saliency of the object that is manipulated, a variance of the dimension of a space during several demonstrations, and some heuristics that express that uncomfortable or exhausting motions reveal the presence of a task. Those decision tests address the problem of how to spot tasks that involve no motion. The method adds some higher level information to a purely statistical analysis and relies on task spaces as an appropriate space to represent movements. However the efficiency of the task space selection depends on the strength of the chosen decision tests.

The common approach of these last works are to project the observed motion in some specific reduced spaces, where the recognition is easier. These spaces can be chosen arbitrary [25], automatically selected [26] or learned [19]. Similarly, in control, smaller-size spaces are used to define the control objectives and modulate the robot behavior. For example the task-function approach [27] expresses generic control objectives in given n-dimensional task spaces. The approach has been extended to handle a hierarchical set of tasks [28], [29] using the redundancy of a system.

The originality of the method presented in this paper is to use the properties of the task-function to perform a task recognition. The main idea is to perform a reverse engineering of an observed motion, knowing the set of all possible tasks that can appear and using the control law in the task space as characteristic trajectories. Under the hypothesis that the motion has been generated by stacking a set of controllers, the motion is processed in order to seek the known behaviors in each task spaces. We named this reverse engineering algorithm approach *reverse control*. While all the approaches presented above can only recognize non-parallel task, we rely on the control redundancy principles to recognize sets of parallel tasks. Projections of the motion in the already-detected orthogonal task spaces ensure an efficient decoupling of the tasks performed by the robot. It has to be noted that our approach relies also on some strong assumptions, listed explicitly below, but that are directly related to the robot control framework.

We introduce in Section III the basics of what the work presented here relies on: the task-function approach. The proposed algorithm to perform the motion analysis is presented in Section IV. Finally, experimentations that validate the method are presented in simulation in Section V and with the real HRP-2 robot in Section VI.

III. Tasks and stack of tasks

The task-function framework [27] is an elegant approach to describe intuitively sensor-based control objectives. The advantage is that expressing the control law in the most suitable subspace, with respect to a given objective, simplifies its construction as well as its execution since the subspace is generally closely linked to the sensors of the robot. Based on the redundancy of the system, this approach can be extended to consider a hierarchical set of tasks [28]. Complex motion can then be composed from simple tasks seen as atomic bricks of motion. This composition mode, along with the obvious composition by temporal sequencing offers a real versatility: subparts of the motion can be used in very different situations without redesigning them for each case.

In the following, we consider that the robot input is the velocity $\dot{\mathbf{q}}$, where $\mathbf{q}$ is the robot configuration vector. A task is defined by a vector space $\mathbf{e}$ and by the reference behavior $\dot{\mathbf{e}}^*$ to be executed in the task space. The differential link between the error and the robot configuration is the Jacobian of the task and is noted $\mathbf{J} = \frac{\partial \mathbf{e}}{\partial \mathbf{q}}$. Various typical behaviour can be chosen for $\dot{\mathbf{e}}^*$. Typically, we will use in the following an exponential decrease, set by

$$\dot{\mathbf{e}}^* = -\lambda \mathbf{e} \tag{1}$$

where $\mathbf{e} = \mathbf{s} - \mathbf{s}^*$ is the error between a current observed value of a signal $\mathbf{s}$ and its arbitrary reference $\mathbf{s}^*$, and $\lambda > 0$ is the gain that tunes the speed of the regulation of $\mathbf{e}$ to 0. For example, the observed feature can be a 3D position $\mathbf{p}$ of one of the robot end-effectors, to be brought to a chosen position $\mathbf{p}^*$ and the Jacobian $\mathbf{J} = \frac{\partial \mathbf{p}}{\partial \mathbf{q}}$.

The control law is given by the least-square solution [30]:

$$\dot{\mathbf{q}} = \mathbf{J}^+ \dot{\mathbf{e}}^* + \mathbf{P}\mathbf{z} \tag{2}$$

where $\mathbf{J}^+$ is the least-square inverse of $\mathbf{J}$, $\mathbf{P} = \mathbf{I} - \mathbf{J}^+\mathbf{J}$ is the projection operator onto the null space of $\mathbf{J}$ and $\mathbf{z}$ is any secondary criterion. $\mathbf{P}$ ensures a decoupling of the task with respect to $\mathbf{z}$. Using $\mathbf{z}$ as a secondary input, the control can be extended recursively to a set of *n* tasks. Those *n* tasks are ordered by priority : task number 1 being the highest priority task, and task number *n* the lowest priority, $task_i$ should not disturb $task_j$ if $i > j$. The recursive formulation of the control law is proposed by [28] :

$$\dot{\mathbf{q}}_i = \dot{\mathbf{q}}_{i-1} + (\mathbf{J}_i \mathbf{P}^{\mathbf{A}}_{i-1})^+ (\dot{\mathbf{e}}^*_i - \mathbf{J}_i \dot{\mathbf{q}}_{i-1}), \quad i = 1 \dots n \tag{3}$$

with $\dot{\mathbf{q}}_0 = 0$ and $\mathbf{P}^{\mathbf{A}}_{i-1}$ is the projector onto the null space of the augmented Jacobian $\mathbf{J}^{\mathbf{A}}_i = (\mathbf{J}_1, \dots \mathbf{J}_i)$. Joint velocities realizing all the tasks is $\dot{\mathbf{q}}^* = \dot{\mathbf{q}}_n$. Each task can be used to generate a common pattern of motion in various situation. In that sense, a task is at the same time the controller that can generate a motion, as well as a descriptor of the motion that is currently executed. This descriptor is quasi symbolic, but comes also with additional parameters that characterize the way it is executed: for example, an exponential-decrease task comes with the parameters λ that characterize its decrease speed. A complete implementation of this approach is proposed in [31] under the name *Stack of Tasks* (SoT).

IV. Motion analysis for task recognition

In the previous section the classical widely-used task-function formalism was recalled. In this section, we propose to use tasks as a set of descriptors to recognize a demonstrated

motion, by identifying the set of tasks that have been used to generate the observed motion. The identified task set can then be used to characterize the observed motion, for example, to distinguish between similar-looking movements. In the following we list and justify the hypothesis considered, then we present an overview of the task recognition method and finally detail the main step of the method.

A. Hypothesis

The observed motion is given through joint trajectories[1]. It is supposed to have been generated using an unknown stack of tasks. However the tasks that may appear in a motion are known.

All the tasks involved in the demonstration are supposed compatible in the sense of the projection $\mathbf{P}$ defined in (2) (no algorithmic singularities [32]).

We call the set of possible tasks the *task pool* and we assume that the behavior model $\dot{\mathbf{e}}^*$ of each tasks is known. However, the parameters of the behaviors (like the velocity of movement, or the desired position) are not known. The knowledge of this set allows to turn the recognition problem into a selection in a finite set problem. Limiting the possible tasks to a finite pool does not dramatically impact on the expressiveness of motion because tasks can be executed in parallel. The expressiveness is then reflected by the different possible task combinations.

The set of active tasks has to be constant during the motion and the tasks start and complete at the same time. As a consequence the hierarchical and concurrent tasks with fuzzy or weighted levels of importance are not considered. Another consequence is that sequential movements and all inequalities and conditional constraints are not considered because such constraints involves a change in the stack of tasks used to generate the movement.

Finally, the kinematic model of the robot is supposed known: it is required to compute the Jacobian and the null-space projector that are used in the recognition part. As a consequence, uncertainties in the kinematic model will affect the performance of the method.

B. Overview

The input of the algorithm is the joint trajectories and the task pool. The algorithm is iterative: at each iteration of the algorithm, the task that seems the most relevant is selected. The selection of a task relies on a curve-fitting score, obtained by projecting the joint-angle trajectory in each task space. Assuming that the kinematic model of the robot is known, the projection of the motion in each task space reconstructs the trajectory in that space. For example, the trajectory of the center of mass (*CoM* task) is recovered from the joint trajectories by the projection of the joint trajectories into the *CoM* task space. The projected trajectories are compared to the theoretical trajectories which are characteristic of the execution of a task. The observed motion is then projected onto the null space of the selected task, cancelling the effect of the task in the observed motion. This projection into the null space relies on the control redundancy and is used to decouple the current tasks from the others. Another iteration of selection-projection is then executed, on the projected motion. The algorithm stops when the original motion is totally cancelled, by the iterative projections.

Algorithm 1 Task selection algorithm

Input: $\hat{\dot{\mathbf{q}}}(t)$
Output: $activePool$
$\mathbf{P}\dot{\mathbf{q}}(t) \leftarrow \hat{\dot{\mathbf{q}}}(t)$
while $\int \|\mathbf{P}\dot{\mathbf{q}}(t)\|^2 dt > \epsilon$ **do**
  for task $i = 1..n$ **do**
    $r_i \leftarrow \mathrm{taskFitting}(i, activePool)$
  end for
  $i_{select} \leftarrow \mathrm{argmin}(r_i)$
  $activePool.\mathrm{push}(i_{select})$
  $\mathbf{P}\dot{\mathbf{q}}(t) \leftarrow \mathrm{projection}(i_{select}, \mathbf{P}\dot{\mathbf{q}}(t))$
end while

The algorithm is showed in Alg. 1. The joint velocity trajectory of the observed motion is denoted $\hat{\dot{\mathbf{q}}}(t)$[2]. $\mathbf{P}\dot{\mathbf{q}}(t)$ denotes the successive projections of the reference motion. Before the first iteration, $\mathbf{P}\dot{\mathbf{q}}(t)$ is set to the reference motion. Then, each iteration projects it in the selected task null space. r_i denotes the score of the cost function of the curve fitting optimization. $activePool$ denotes the set of tasks selected during the algorithm.

If the observed motion is exactly generated by a SoT, the resulting trajectory $\mathbf{P}\dot{\mathbf{q}}$ after projection onto all the active-task spaces is null. However, in presence of noise (*ie* when acquiring motion through real sensors) a residue is systematically obtained, which implies to use a threshold as stop criteria: the loop ends when the residue is below the noise of the acquisition chain. ϵ denotes the threshold of the motion norm below which the algorithm stops.

The next subsections describe the two main functions of the algorithm. The procedure $\mathrm{projection}(i, \dot{\mathbf{q}}(t))$ computes the projection of velocity onto the null space of the task i and apply the projection to the motion. The procedure $\mathrm{taskFitting}(i, activePool)$ handles the curve fitting of the observed motion and the theoretical motion. The process is detailed in section IV-D.

C. Projection of the motion

The reconstruction of the trajectories in each task space from the joint trajectories is directly done by multiplying the joint trajectories by the task jacobian.

In order to cancel the effect of a detected task i_{select}, the joint trajectories are projected onto the null space of that task by multiplying it with the projector onto the null space of all

[1]The joint-angle trajectories are observed using a motion capture system as shown in Section VI-A.

[2]In the remainder of this article, observations and measures are denoted $\hat{\cdot}$ while references and desired values are denoted $\cdot^*$

tasks to cancel. For every time t of the motion time interval:

$$\mathbf{P}\dot{\mathbf{q}}(t) \leftarrow \mathbf{P}^{\mathbf{A}}_{\mathbf{i}\mathrm{select}}(t)\mathbf{P}\dot{\mathbf{q}}(t) \tag{4}$$

where $\mathbf{P}^{\mathbf{A}}(t)$ is jointly updated:

$$\mathbf{P}^{\mathbf{A}}(t) = \mathbf{P}^{\mathbf{A}}(t) - (\mathbf{J}_{\mathbf{i}\mathrm{select}}(t)\mathbf{P}^{\mathbf{A}}(t))^{+}(\mathbf{J}_{\mathbf{i}\mathrm{select}}(t)\mathbf{P}^{\mathbf{A}}(t))$$

The projector $\mathbf{P}^{\mathbf{A}}(t)$ is initialized to $\mathbf{P}^{\mathbf{A}}_{\mathbf{0}}(t) = \mathbf{I}$. The remaining motion after projection $\mathbf{P}\dot{\mathbf{q}}(t)$ is then analyzed to detect the potentially-remaining tasks.

The projection operation will nullify the effect of the motion in the selected-task space. It has in fact two different effects in the configuration space: on the first hand, it cancels the component of the motion that is independent with regard to the other tasks; on the other hand, it modifies by the way the part of the motion that is coupled with the effect of the remaining tasks. The first effect is beneficial because it avoids future false detection that could be caused by non-linear reflection of the currently-selected task in the remaining task space. However, care has to be taken with the modification of the coupled part, as explained by the following example.

Consider a motion composed of two arbitrary tasks $\mathbf{e}_a$ and $\mathbf{e}_b$. The control law to execute both tasks is given by:

$$\dot{\mathbf{q}} = \mathbf{J}_a^{+}\dot{\mathbf{e}}_a^{*} + (\mathbf{J}_b\mathbf{P}_a)^{+}(\dot{\mathbf{e}}_b^{*} - \mathbf{J}_b\mathbf{J}_a^{+}\dot{\mathbf{e}}_a^{*}) \tag{5}$$

If $\mathbf{e}_a$ is detected first, (4) is applied with $i_{select} = a$. Multiplying (5) by $\mathbf{P}_a$ cancels the motion in the Task a space:

$$\mathbf{P}_a\dot{\mathbf{q}} = \underbrace{\mathbf{P}_a\mathbf{J}_a^{+}\dot{\mathbf{e}}_a^{*}}_{\mathbf{0}} + \mathbf{P}_a(\mathbf{J}_b\mathbf{P}_a)^{+}(\dot{\mathbf{e}}_b^{*} - \mathbf{J}_b\mathbf{J}_a^{+}\dot{\mathbf{e}}_a^{*}) \tag{6}$$

The first term is null by definition of $\mathbf{P}_a$. The motion due to $\mathbf{P}_a\dot{\mathbf{q}}$ in the Task b space is obtained by multiplying by $\mathbf{J}_b$:

$$\mathbf{J}_b\mathbf{P}_a\dot{\mathbf{q}} = \dot{\mathbf{e}}_b^{*} - \mathbf{J}_b\mathbf{J}_a^{+}\dot{\mathbf{e}}_a^{*} \tag{7}$$

since $\mathbf{J}_b\mathbf{P}_a(\mathbf{J}_b\mathbf{P}_a)^{+} = \mathbf{I}$ by hypothesis. The first term is the independent component of the second task, and the second term is the component coupled with Task a. Therefore, the projection onto the null space of the discovered task at the first iteration will induce a coupling effect that has to be handled when trying to discover another task. This coupling will be handled in the next section.

D. Task fitting by optimization

We denote $\widehat{\dot{\mathbf{e}}}(t)$ the trajectory due to the current $\mathbf{P}\dot{\mathbf{q}}$ in the observed-task space. The quantification of the relevance of the given task $\mathbf{e}$, with respect to the current motion $\mathbf{P}\dot{\mathbf{q}}$, is achieved by applying a least-square optimization between the actual observed motion projected in the task space and the reference behavior of a task over the unknown parameters:

$$\hat{\mathbf{x}} = \arg\min_{\mathbf{x}} \frac{\int \|\widehat{\dot{\mathbf{e}}}(t) - \dot{\mathbf{e}}_{\mathbf{x}}(t)\|^2 \mathrm{d}t}{\int \|\widehat{\dot{\mathbf{e}}}(t)\|^2 \mathrm{d}t} \tag{8}$$

where $\widehat{\dot{\mathbf{e}}}(t)$ is the observed trajectory and $\dot{\mathbf{e}}_{\mathbf{x}}(t)$ is the trajectory generated by the model using the parameters $\mathbf{x}$. The model is generic and an example of a behavior model is an exponential decrease with the parameters $\mathbf{x} = [x_1, x_2, x_3]$:

$$p_x(t) = x_1 \mathrm{e}^{-x_2 t} + x_3$$

The score will thus be the residue after trying to obtain the best correspondence with the given model in the task space.

In some cases, the trajectory $\widehat{\dot{\mathbf{e}}}$ can be observed directly. For example, if the task space is the 3D position of the hand, a direct observation is possible. However, in general, a direct observation is not possible. For example, the center of mass of the robot is difficult to observe directly. In addition, it is not possible to observe directly the effect of the successive projection. $\widehat{\dot{\mathbf{e}}}$ is then obtained by the kinematic model and joint trajectories of the robot: $\mathbf{J}_i\hat{\dot{\mathbf{q}}}$. However, proceeding so would lead to (7), where the coupling with previously selected tasks appears. The projected motion is then augmented with coupling compensation.

$$\widehat{\dot{\mathbf{e}}}(t) = \mathbf{J}_i\mathbf{P}^{\mathbf{A}}\dot{\mathbf{q}}(t) + \mathbf{J}_i\mathbf{J}^{\mathbf{A}+}\widehat{\dot{\mathbf{e}}}_{\mathbf{A}}(t) \tag{9}$$

where $\mathbf{A}$ is the set of tasks that have already been detected and i is the task candidate. All the terms corresponding to $\mathbf{A}$ are known since identified in the previous iteration of the algorithm.

In practice, the observation is sampled and the integral is a sum. The optimization problem (8) is in general a non linear problem. To solve it numerically, the CFSQP solver has been used [33]. The result of the optimization produces at the same time the residue used as a criteria to select the most plausible task, and the numerical parameters of the task (for example the gain and the desired position, when considering a proportional task (1)).

E. Order of the nullification

The previous composition of detection-projection enables to remove the side effects of the previously detected tasks without introducing any coupling in the non-detected tasks. We prove now that the order of the detected tasks and the subsequent projection does not affect the detection of the remaining tasks.

Consider the observed motion to have been generated by a two-stages SoT (5). The reference motion in Task i space is denoted $\dot{\mathbf{e}}_i^{*}$. The motion induced in Task i space by the original motion $\dot{\mathbf{q}}$ is denoted $\widehat{\dot{\mathbf{e}}}_i$, while the motion induced in Task j after removing the motion from Task i $\mathbf{P}_j\dot{\mathbf{q}}$ is denoted $\widehat{\dot{\mathbf{e}}}_{j|i}$.

We prove that both tasks can be detected (and removed) in arbitrary orders.

Proposition IV.1. *Consider a motion generated by a two-stage SoT. Then $\widehat{\dot{\mathbf{e}}}_i = \widehat{\dot{\mathbf{e}}}_{i|j} = \dot{\mathbf{e}}_i^{*}$, for $i = a$ and $j = b$, and reciprocally, for $i = b$ and $j = a$.*

Proof:

1) Detecting Task a, then Task b: The direct implication is straight forward. At iteration 1, the observation in Task a space is directly $\widehat{\dot{\mathbf{e}}}_a = \mathbf{J}_a\hat{\dot{\mathbf{q}}} = \dot{\mathbf{e}}_a^{*}$. After the removal of Task a, the observation in Task b space is obtained from (9):

$$\begin{aligned}\widehat{\dot{\mathbf{e}}}_{b|a} &= \mathbf{J}_b\hat{\dot{\mathbf{q}}} + \mathbf{J}_b\mathbf{J}_a^{+}\widehat{\dot{\mathbf{e}}}_a \\ &= \underbrace{\mathbf{J}_b(\mathbf{J}_b\mathbf{P}_a)^{+}}_{\mathbf{I}}(\dot{\mathbf{e}}_b^{*} - \mathbf{J}_b\mathbf{J}_a^{+}\dot{\mathbf{e}}_a^{*}) + \mathbf{J}_b\mathbf{J}_a^{+}\dot{\mathbf{e}}_a^{*}\end{aligned}$$

implicating directly $\widehat{\dot{\mathbf{e}}}_{b|a} = \dot{\mathbf{e}}_b^{*}$, since $\mathbf{J}_b(\mathbf{J}_b\mathbf{P}_a)^{+} = \mathbf{J}_b\mathbf{P}_a(\mathbf{J}_b\mathbf{P}_a)^{+} = \mathbf{I}$ and $\widehat{\dot{\mathbf{e}}}_a = \dot{\mathbf{e}}_a^{*}$.

The observation obtained in Task b space is independent from the projection in the null space of Task a.

2) Detecting Task b, then Task a: At the first iteration, the observation in Task b space is directly

$$\widehat{\dot{\mathbf{e}}}_b = \mathbf{J}_b\mathbf{J}_a^+\dot{\mathbf{e}}_a^* + \mathbf{J}_b(\mathbf{J}_b\mathbf{P}_a)^+(\dot{\mathbf{e}}_b^* - \mathbf{J}_b\mathbf{J}_a^+\dot{\mathbf{e}}_a^*)$$

which implies $\widehat{\dot{\mathbf{e}}}_b = \dot{\mathbf{e}}_b^*$ since $\mathbf{J}_b(\mathbf{J}_b\mathbf{P}_a)^+ = \mathbf{I}$.

Projecting the original motion (5) in the null space of Task b at the end of the first iteration results in the following Task a observation:

$$\begin{aligned}\widehat{\dot{\mathbf{e}}}_{a|b} =& \mathbf{J}_a\mathbf{P}_b\mathbf{J}_a^+\dot{\mathbf{e}}_a^* + \mathbf{J}_a\mathbf{P}_b(\mathbf{J}_b\mathbf{P}_a)^+(\dot{\mathbf{e}}_b^* - \mathbf{J}_b\mathbf{J}_a^+\dot{\mathbf{e}}_a^*) + \mathbf{J}_a\mathbf{J}_b^+\widehat{\dot{\mathbf{e}}}_b \\ =& (\mathbf{J}_a\mathbf{P}_b\mathbf{J}_a^+ - \mathbf{J}_a\mathbf{P}_b(\mathbf{J}_b\mathbf{P}_a)^+\mathbf{J}_b\mathbf{J}_a^+)\dot{\mathbf{e}}_a^* \\ &+ (\mathbf{J}_a\mathbf{P}_b(\mathbf{J}_b\mathbf{P}_a)^+ + \mathbf{J}_a\mathbf{J}_b^+)\dot{\mathbf{e}}_b^*\end{aligned}$$

We show that this complex sum is in fact simply $\dot{\mathbf{e}}_a^*$ (see Appendix A). This proves that the same fitting is obtained independently of what task is detected first. ■

The proof can be extended using exactly the same arguments for a set of n tasks. Similarly, it is straightforward to prove that whatever the order of detection, the resulting motion after all the projections is null.

V. Results in simulation

This section details a series of experimentation in simulation to validate the recognition algorithm. Simulation allows us to emphasize the nominal behavior of the algorithm without any sensor noise. The first experiment simply validates the projection of the motion (section V-B). The second part compiles a set of experiments that validate the task recognition algorithm (section V-C). For each experiment of that set, the task recognition algorithm is applied to two similar-looking motions: the two motions have been artificially built to be ambiguous when compared to each other, in order to illustrate the efficiency of the algorithm regarding the precision of the recognition. Fig. 2, Fig. 3 and Fig. 4 show the final posture of the ambiguous motions used in those experiments. All motions involved in the experiments are summed up in Table I. The description of the tasks used to build these motions are detailed below.

A. Set-up

The reference motions have been generated by using the model of the humanoid robot HRP-2 having 30 actuated degrees of freedom plus six degrees of freedom on the freeflyer. Every motions start from the half-sitting pose. As classically done in inverse kinematics, the under-actuation of the freeflyer is resolved by constraining the left foot to be on the ground.

The set of tasks considered in those experiments are :

- *CoM* : the center of mass of the robot is constraint to maintain the static balance (3 DOF)
- *Gaze* : the robot looks at one point in the Cartesian space (2 DOF)
- *Twofeet*: both feet stay flat with regard to each other (6 DOF)

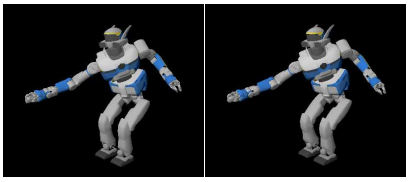


Fig. 2. Left: The final position of *motion 2.a*; Right: The final position of *motion 2.b*.

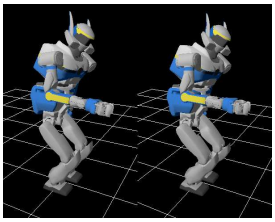


Fig. 3. Left: The final position of *motion 3.a*; Right: The final position of *motion 3.b*. The difference lying in the orientation of the right hand is difficult to spot. In *motion 3.b*, the right hand is parallel to the ground.

	(a)	(b)
Motion 1	*Right grab* *Left grab* *Gaze* *CoM* *Twofeet*	
Motion 2	*Right grab* *CoM* *Twofeet*	*Left grab* *Right grab* *CoM* *Twofeet*
Motion 3	*Gaze* *Right grab* *CoM* *Twofeet*	*Gaze* *Right Screw* *CoM* *Twofeet*
Motion 4	*Gaze* *CoM* *Twofeet*	*Left Screw* *CoM* *Twofeet*
Motion 5	*Right grab* *Gaze* *CoM* *Twofeet*	*Right grab* *Chest* *CoM* *Twofeet*

TABLE I
THE TABLE OF CONSIDERED MOTIONS AND TASKS.

- *Left/Right grab* : either the left or right hand of the robot reaches a point defined in the Cartesian space (3 DOF)
- *Screw* : similar to the *grab* task, but the desired position has to be reached with a defined orientation (6 DOF)
- *Head* : the head of the robot is constrained in position and orientation (6 DOF)
- *Chest* : the chest of the robot is constrained in orientation (3 DOF)

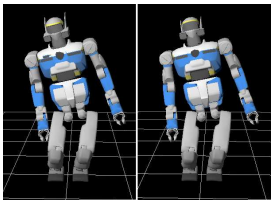


Fig. 4. Left: The final position of *motion 4.a*; Right: The final position of *motion 4.b*.

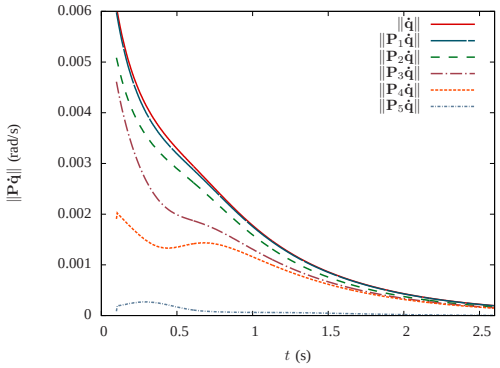


Fig. 6. Experiment 1: Evolution of the norm of the motion after successively projecting the motion in the tasks null spaces.

B. Experiment 1 : Preliminary validation

In this experiment, a basic motion is generated. Then the recognition by fitting, the projection and the termination condition of the algorithm are validated. The reference motion is *motion 1.a* (see Table I). The motion is given to the detection program which selects the tasks that fit the most closely by the optimization.

Fig. 5 shows snapshots of the original motion and of the motions after successive projections in the null spaces of the detected tasks : *Right grab*, *CoM*, *Gaze*, *Twofeet* and *Left grab*. Each projection cancels a part of the motion, and the robots motion becomes null when all projections are applied, which means that all relevant tasks have been detected. As we can see in the second line, the movement of the right hand is nullified. The cancellation of the *CoM* from the third line is more difficult to perceive. On the forth line, the cancellation of the head movement due to the gaze is very clear. On the fifth line, the cancellation of the task *Twofeet* removes the compensation done with the right leg. Finally, as shown on the last line, the cancellation of all tasks leads to a null motion. Fig. 6 shows the evolution of the norm of the motion, defined by the sum square of the joint-angle velocities of the robot. Each projection strictly decreases the norm of the velocity (*ie* the quantity of movement). After five projections, the movement is completely nullified, which confirms that all the active tasks have been discovered. The algorithm can then stop.

C. Experiment 2 : Distinction among similar-looking motions

An interesting challenge in motion detection is to make distinction between two motions involving different tasks but that produce very similar joint trajectories. Anthropomorphic

Reference	Detected	$\int \|\|\dot{q}(t)\|\|^2 dt$	$\int \|\|P\dot{q}(t)\|\|^2 dt$
CoM Right Hand Twofeet	CoM Right Hand Twofeet	0.364398	0.00159355
CoM Left Hand Right Hand Twofeet	CoM Left Hand Right Hand Twofeet	0.538329	0.0035343

TABLE II
RESULTS OF THE TASK SELECTION ALGORITHM FROM THE ANALYSIS OF *motion 2.a* AND *motion 2.b*.

algorithm would use the context to perform the disambiguation. The work presented in this paper shows that the fitting criterion is sufficient to disambiguate similar-looking motion. Three ambiguous pairs of movements are presented to illustrate this capability.

1) Reaching motions: In this section, two motions are considered: *motion 2.a* and *motion 2.b*. The first one is a far-reaching motion with the right hand. The reaching motion of the right hand has an influence on the left arm through the *CoM* task: to regulate its balance, the robot puts its left arm behind. The second motion is the same reaching task for the right hand, added with a second reaching task on the left hand. The desired position of the left hand is artificially set to the final position of the left hand obtained at the first motion.

The final states of the robot for the two motions are shown in the Fig. 1(c). A video showing those two motions on the HRP-2 is available[3]. The two motions look very similar, and it is very difficult to the human eye to tell which motion involves a left and right hand task without the context. In the first case, the motion of the left hand is due to *CoM* and *Right Hand* task, as it is a side effect to compensate the balance perturbation induced by the right hand. In the second case, the motion of the left hand is decoupled, since the left hand has been driven by its own goal. However, in the proper task spaces, those motions appear clearly different. Fig. 7 shows an example of the result of the task fitting of *Right Hand* and *Left Hand* task applied to *motion 2.a*. The residue of the optimization for *Left Hand* is higher than the residue for *Right Hand*. At this step, the movements analyzed is better explained by *Right Hand* than by *Left Hand*. The results of the detection algorithm are showed in Table II. The first column shows the tasks being used in the reference motion, the second column shows the tasks selected by the algorithm, the third column shows the norm of the reference motion (quantity of motion initially observed), and the last column shows the norm of the reference motion projected onto the null space of all selected tasks. The final quantity of movement after projection is very low for both motions compared to the threshold for the stop criterion $\int \|P\dot{q}(t)\|^2 dt > \epsilon$ that is empirically fixed to $\epsilon = 0.07$ in order to handle numerical noise due to successive projections (the threshold depends on the number of tasks involved in the movement).

Finally Fig. 8 shows how the norm of joint-angle velocities corresponding to the tasks *Right grab* and *Left grab* evolves after the projections. Projecting *motion 2.a* onto the null spaces of *Right grab* task will decrease its associated theoretical joint-

[3]http://homepages.laas.fr/shak/videos/

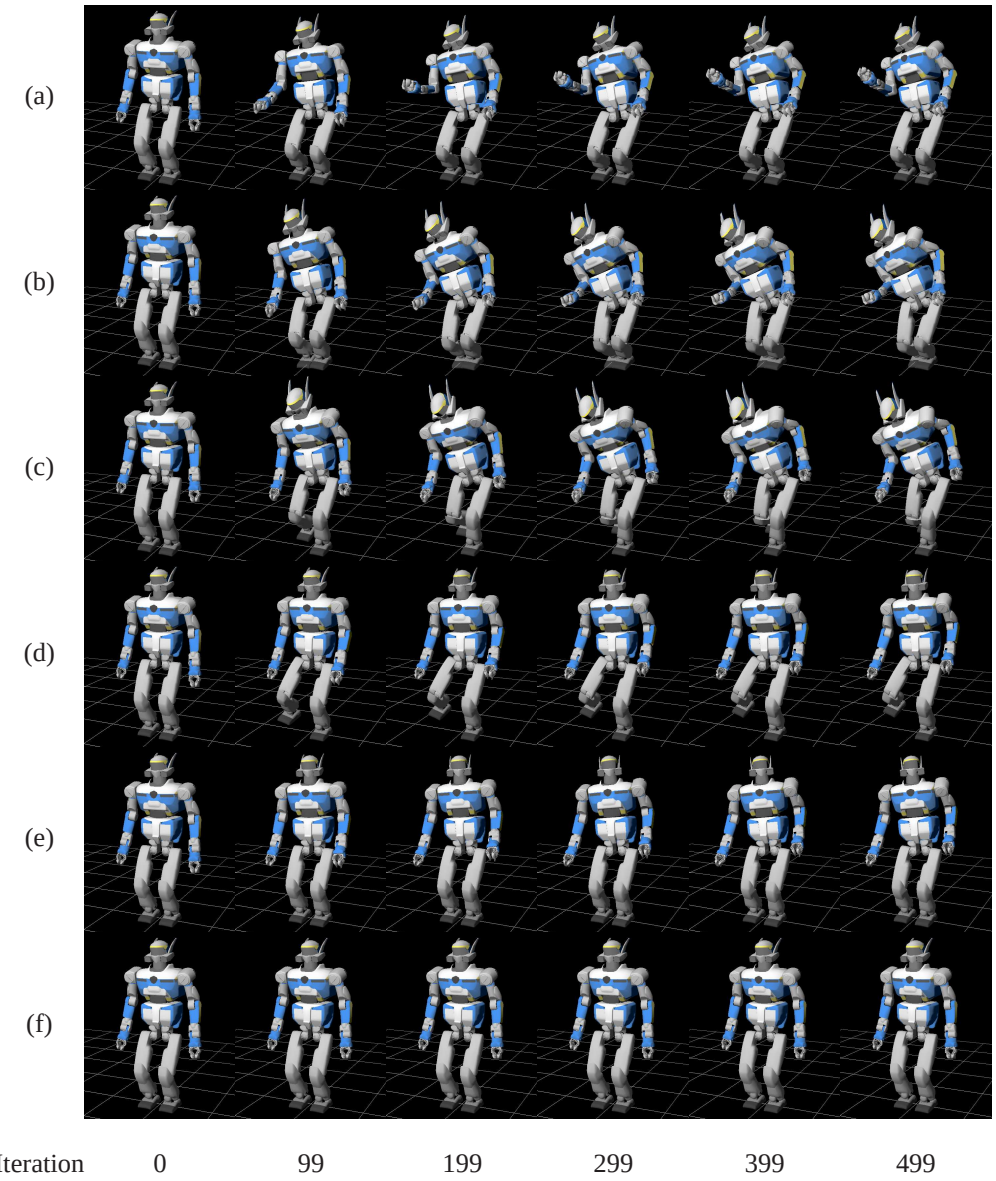


Fig. 5. Experiment 1: The motion generated by a stack of tasks involving : *Right and Left grab*, *CoM*, *Gaze*, *Twofeet* is represented in the row (a). The remaining rows represent the successive projections of the motion in the null spaces of the tasks, (b) Right grab, (c) CoM, (d) Gaze, (e) Twofeet, (f) Left grab.

angle velocities while leaving the *Left grab* one unchanged. Fig. 8(a) shows that the theoretical joint-angle velocities associated to *Left grab* task is decreased after the projection of the motion in the null space of *CoM* task. That explicits that the left arm of the robot was moved by *CoM* task. By the way, the projection prevent the left task to be detected at any further iteration of the algorithm. But, when projecting *motion 2.b* into the nullspace of *Right grab* and *CoM* task the joint-angle velocities associated to *Left grab* task is still significant (Fig. 8(b)). That means that *CoM* task has a little influence on the motion of the left arm, and that motion on the left arm is due to another task. Therefore, the task selection algorithm will keep looking for the task that has controlled the left arm. After the detection of the two (for *motion 2.a*) and three (for *motion 2.b*) main tasks, the norm of the last joint-angle velocities are not null, because the task selection algorithm has not finished and other tasks have not been selected yet. *Twofeet* task is then detected but the projection is not displayed on the figure for the sake of clarity. On the other hand Fig. 9 shows the task fitting for *motion 2.b*.

2) Grabbing VS Screwing: In this section, the two motions considered are *motion 3.a* and *motion 3.b*. Both motions share the same position target for the right hand. The only difference between the two demonstrated motions is the presence of an orientation constraint for the right hand in *motion 3.b*. The final positions of those motions are shown in the Fig. 3. Table III shows the results of the task-selection algorithm which performed successfully. As in the previous experiments, the task *Screw* can not be properly fitted on the trajectory generated on *motion 3.a*. In *motion 3.b*, both tasks *Grab* and *Screw* are detected, whereas the task *Screw* is not selected in *motion 3.a*. The selection of both tasks in *motion 3.b* is

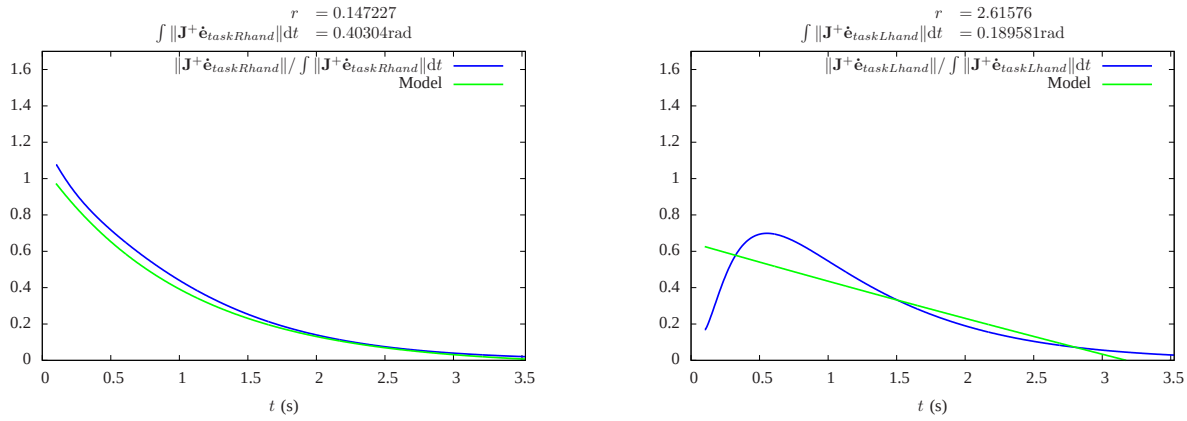
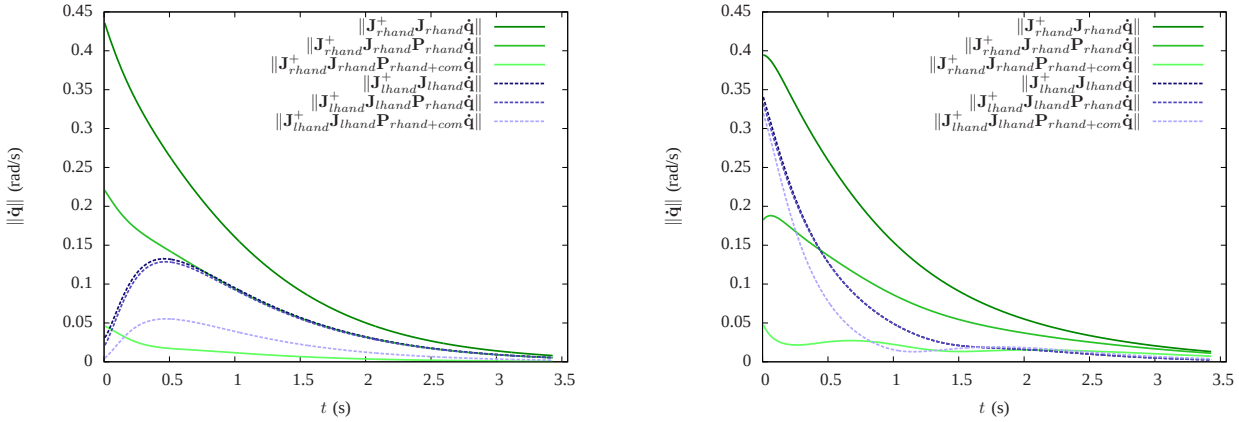


Fig. 7. Experiment 2: Task fitting on *motion 2.a* on the right and the Left grab tasks. Variable r is the residue, *ie* the distance between the two curves. *Right Hand* motion is properly fitted by the task model, showing that the task is active. *Left Hand* is not fitted properly, since the task is not active. The values of the residues show that the task *Right grab* is a better explanation for the movement than the task *Left grab*.



(a) Motion 2.a (b) Motion 2.b

Fig. 8. Experiment 2: Evolution of the trajectory $\dot{\mathbf{q}}$ projected in the spaces of *Right grab* (in solid lines) and *Left grab* (in dashed lines) after successive projection of the motion in the nullspace of the task *Right grab* and in the nullspace of the task *CoM*. In *motion 2.a*, a great part of the motion of the left arm is due to *CoM* task: removing *CoM* task will cancel almost all motion in the left arm. In *motion 2.b*, the motion of the left arm is not only involved by *CoM* task but mainly by *Left grab* task. Removing *CoM* task will only cancel a small part of the motion of the left arm.

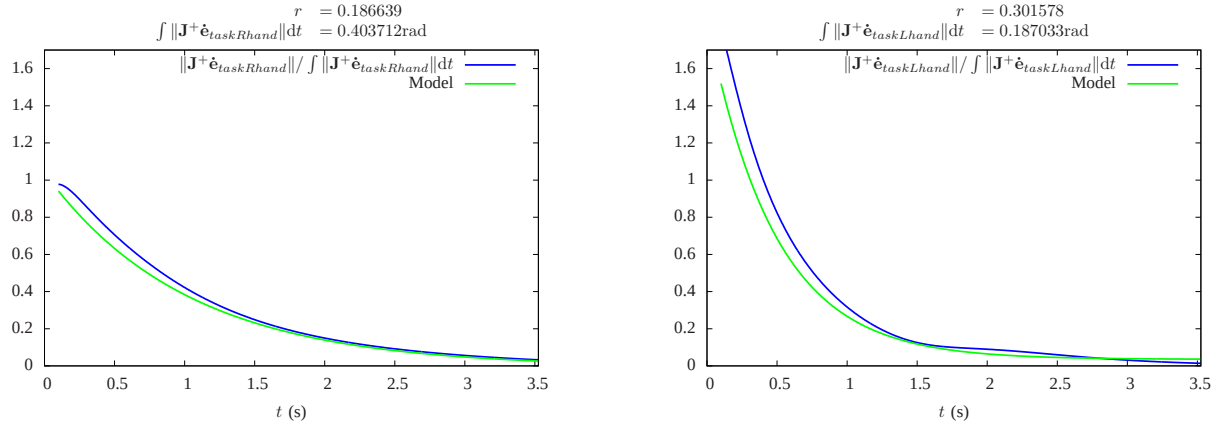


Fig. 9. Experiment 2: Task fitting on *motion 2.b* on the right and the left hand tasks. Both motion of the right and left hand are fitted by the model, with small residue: the tasks are detected.

Reference	Detected	$\int \|\dot{q}(t)\|^2 dt$	$\int \|P\dot{q}(t)\|^2 dt$
CoM Gaze Grab Twofeet	CoM Gaze Grab Twofeet	0.619266	0.00245631
CoM Gaze Screw Twofeet	CoM Gaze Grab Screw Twofeet	0.717041	0.00344557

TABLE III
RESULTS OF THE TASK SELECTION ALGORITHM FROM THE ANALYSIS OF THE *motion 3.a* AND *motion 3.b*.

Reference	Detected	$\int \|\dot{q}(t)\|^2 dt$	$\int \|P\dot{q}(t)\|^2 dt$
CoM Gaze Twofeet	CoM Gaze Twofeet	0.534478	0.0545944
CoM Screw Twofeet	CoM Screw Twofeet	0.558084	0.0023297

TABLE IV
RESULTS OF THE TASK SELECTION ALGORITHM FROM THE ANALYSIS OF *motion 4.a* AND *motion 4.b*.

explained by the fact that *Grab* is actually a sub-task of *Screw*. The residual motion after successive projections is finally very close to 0, which proved that all tasks were properly detected.

3) Screw VS Gaze: The two motions considered are *motion 4.a* and *motion 4.b*. *motion 4.a* can be described by the following scenario : an object is in front of the robot, and creates an occlusion in the vision of the robot. To get rid of this occlusion, the robot can lean on its right. While the robot leans, the left hand is dragged by the chest as an involuntary side effect. The position and orientation of that hand is recorded as a desired state of the hand for the first *motion 4.b* task. In *motion 4.b*, the gaze is uncontrolled. The motion of the head is, that time, a side effect of the left hand movement. The final states of the robot are shown in Fig. 4 and the analysis results are summarized in Table IV.

As previously, correct tasks are detected for each movement. The residue after projections on all detected tasks are very small, proving that everything was properly detected and subtracted.

In conclusion, we have shown experimentally in this section that, without noise, the detection algorithm performs perfectly, *ie* detects exactly all the active tasks, without any false detection, and issues a neglectable residue of projected motion due to numerical noise. The following section will consider realistic cases of noisy signals acquired by real sensors.

VI. EXPERIMENTATION ON THE ROBOT

In this section, we experimentally demonstrate the validity of the task recognition algorithm in realistic situation in presence of noise. This time, the reference motion is demonstrated by the real HRP-2 and observed using a motion-capture system. (Fig. 10). As previously, the task recognition algorithm is applied to two pairs of similar looking motions. First, we briefly explain how the motion-capture system is used to record the joint-angle trajectory. Then two pairs of motions are observed and provided to the recognition algorithm.

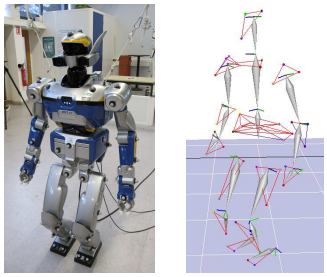


Fig. 10. Markers set and virtual skeleton of HRP-2.

A. Experimental setup

A motion is executed with the *SoT* framework on the HRP-2 robot equipped with markers on each body part. The motion-capture system used is composed of 10 digital cameras and record data at 200Hz. The motion-capture system provides the trajectory of each body of the robot. The data collected from those markers are used to build a virtual skeleton that match the kinematic hierarchy of the robot (Fig. 10). The result of this process is the 6D trajectories of all the bodies in space, without any joint constraints. The analysis of the motion is performed on the joint-angle trajectories. Therefore, a joint-angle trajectories have to be computed from the motion-capture data. The joint-angle trajectories are computed as classically done by optimizing the distance between the transformation matrix ${}^{W}\mathbf{R}_{q_i}(\mathbf{q})$ from the robot origin to each joint of the robot and the measured transformation matrix ${}^{W}\hat{\mathbf{R}}_{q_i}(t)$ using the robot joint limits as constraints.

$$\hat{\mathbf{q}}(t) = \quad \arg\min_{\mathbf{q}} \quad \sum_{i}^{n} \| {}^{W}\hat{\mathbf{R}}_{q_i}(t) \ominus {}^{W}\mathbf{R}_{q_i}(\mathbf{q}) \|^2 \quad (10)$$

$$\text{s.t.} \qquad q_{i\min} \leq q_i \leq q_{i\max}, \; i = 1..n \quad (11)$$

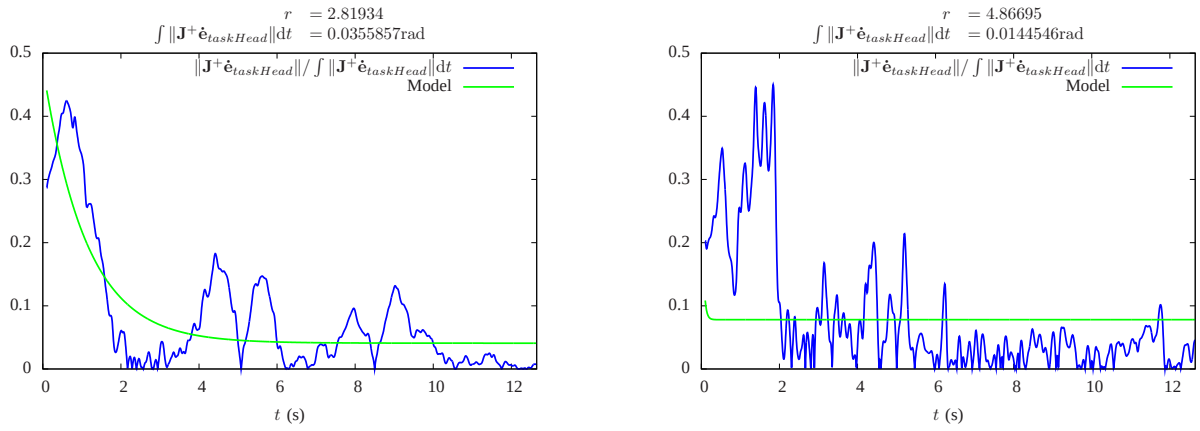
where $\mathbf{q}$ is the robot joint configuration vector, $\ominus$ is the distance operator in SO(3), ${}^{W}\mathbf{R}_{q_i}(\mathbf{q})$ is computed using the kinematic model of the robot, and ${}^{W}\hat{\mathbf{R}}_{q_i}$ is obtained by:

$${}^{W}\hat{\mathbf{R}}_{q_i}(t) = {}^{W}\mathbf{R}_{W_C} \times {}^{W_C}\hat{\mathbf{R}}_{M_i}(t) \times {}^{M_i}\mathbf{R}_{q_i} \quad (12)$$

where ${}^{W}\mathbf{R}_{W_C}$ and ${}^{M_i}\mathbf{R}_{q_i}$ are the constants displacement due to the differences between an arbitrary skeleton model of the motion-capture and our kinematic model of the robot, computed in a calibration step. ${}^{W_C}\hat{\mathbf{R}}_{M_i}$ is the measured transformation matrix from the origin of the motion-capture reference frame to the virtual body i.

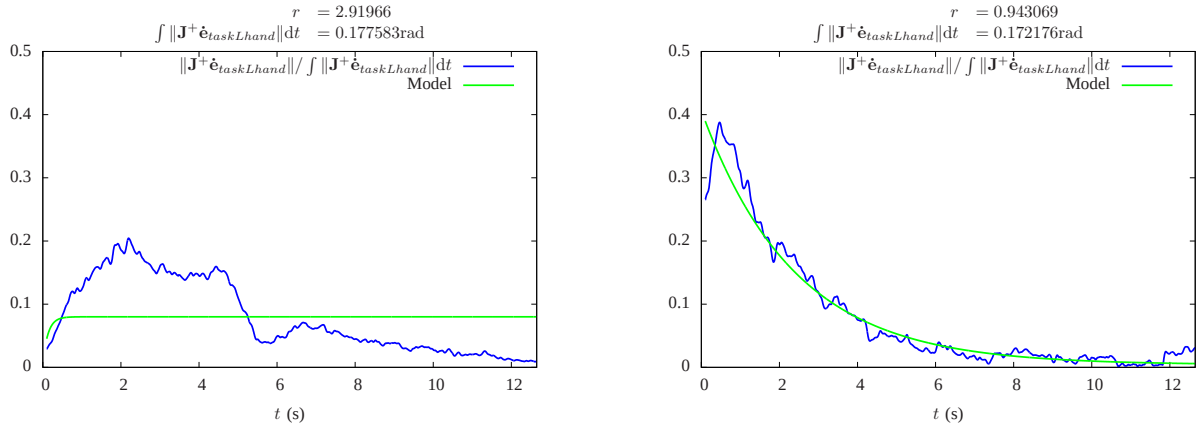
The obtained joint-angle trajectories are then used for the reverse-engineering task recognition.

On the robot, the mechanical design includes a flexibility after the ankle joint that interferes with the motion at the very beginning as the acceleration of the joint-angle increases quickly. The flexibility is not modeled neither in the control algorithm nor the detection method. In order to bypass the influence of the flexibility, the motion analysis is done after the first 100ms where the influence of the flexibility is predominant. In the following two sections, we consider that the measures are directly the $\hat{\mathbf{q}}$ trajectories given by (12).



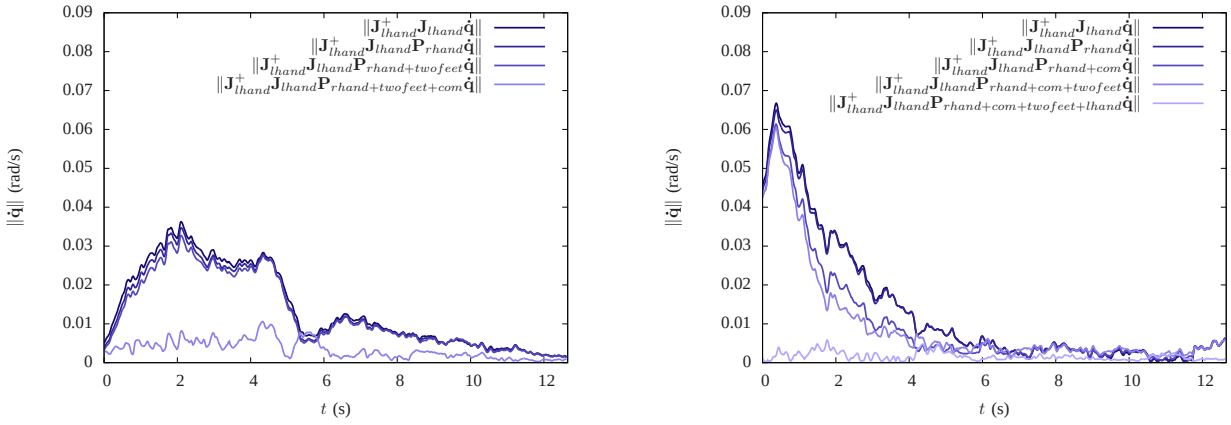
(a) Motion 2.a (b) Motion 2.b

Fig. 11. Experiment 3: Fitting at the first iteration of the task selection algorithm for *head* task for *motion 2.a* and *motion 2.b* For both motion, r is not the lowest residue amongst all the tasks fitting: *Head* task, not active, is not selected.



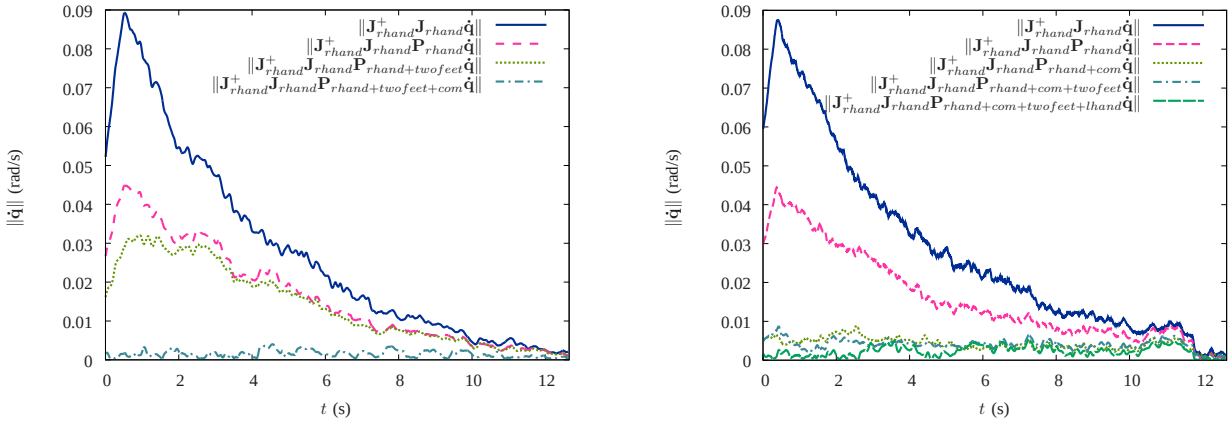
(a) Motion 2.a (b) Motion 2.b

Fig. 12. Experiment 3: Fitting at the first iteration of the task selection algorithm for *Left grab* task for *motion 2.a* and *motion 2.b Left grab* task is not pertinent for *motion 2.a*, but is selected for *motion 2.b*.



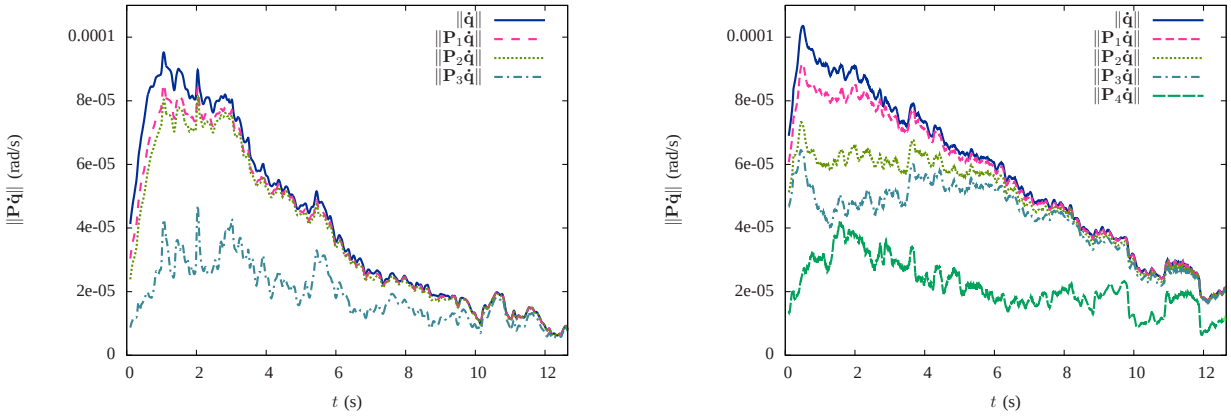
(a) Motion 2.a (b) Motion 2.b

Fig. 13. Experiment 3: Norm of motions associated to *Left grab* task (*ie* motion of the left arm). The motion is cancelled after removing the motion due to *CoM* task in *motion 2.a*. In that case, it means that the left arm motion was due to *CoM* task. Whereas in *motion 2.b*, the motion of the left arm is not cancelled after removing *CoM* task, but is cancelled after removing *Left grab* task : the motion of the left arm was due to *Left grab* task.



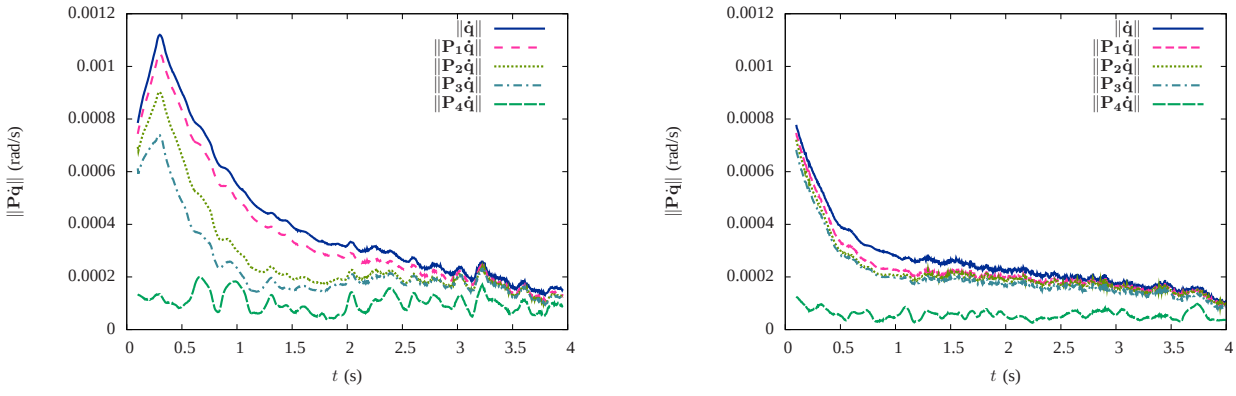
(a) Motion 2.a (b) Motion 2.b

Fig. 14. Experiment 3: Quantity of motion in the right hand space after successive projection. The quantity of motion in the right hand space is almost null after the nullification of the tasks *Right grab and CoM*. It means that the motion of the right arm is a consequence of *Right grab* and *CoM* tasks. It is consistent with the fact that, in order to maintain its balance, the right arm has to move.



(a) Motion 2.a (b) Motion 2.b

Fig. 15. Experiment 3: Evolution of the norm of the motion after successively projecting the motion in the tasks null space for *motion 2.a* and *motion 2.b*. The norm decreases iteratively during the recognition algorithm.



(a) Motion 5.a (b) Motion 5.b

Fig. 17. Experiment 4: Evolution of the norm of the motion after successively projecting the motion in the tasks null space for *motion 5.a* and *motion 5.b*. The norm decreases iteratively during the recognition algorithm.

Reference	Detected	$\int \|\dot{q}(t)\|^2 dt$	$\int \|P\dot{q}(t)\|^2 dt$
CoM Right grab Twofeet	CoM Right grab Twofeet	0.104835	0.0482885
CoM Left grab Right grab Twofeet	CoM Left grab Right grab Twofeet	0.142293	0.0541836

TABLE V
RESULTS OF THE TASK SELECTION ALGORITHM FROM THE ANALYSIS OF *motion 2.a* AND *motion 2.b* ON THE REAL ROBOT.

B. Experiment 3: Grabbing VS Maintaining balance

This experiment corresponds to the one realized in simulation in section V-C1: the first motion is a motion of the left arm induced by the coupling between a far reaching of the right hand, and the balancing *CoM* task (*motion 2.a*). The second motion is a reaching of both hands, built to be ambiguous with regard to the first motion (*motion 2.b* in Table I). Table V shows the result of the identification algorithm for *motion 2.a* and *motion 2.b* with a termination threshold set to $\epsilon = 0.09$. This threshold was chosen to handle the noise introduced in the motion capture data.

For *motion 2.a*, the order of the task extraction is : *Right grab*, *Twofeet* and *CoM* task. Whereas for *motion 2.b*, the extracted tasks are : *Right grab*, *CoM*, *Twofeet* and *Left grab*.

Fig. 11-12 show the fitting at the first iteration of the algorithm for *Head* and *Left grab* tasks. As expected, the fitting performs well only for *Left grab* task in *motion 2.b* while it is rejected for *motion 2.a*. Although no *Head* task is involved in the two motions, the head is not fixed in space due to the motion of the neck for balancing. In *motion 2.b*, the candidate *Head* task involves less motion than in *motion 2.a*. The reason is that when the opposite end effectors of the robot are constrained, the chest is less used. However, *Head* task is not kept as a candidate for any of the two motion (the associated residue r is never the lowest).

The evolution of the motion projected into *Left grab* and *Right grab* spaces after the successive projections onto the automatically-selected nullspaces are shown in Fig. 13. For *motion 2.a*, the motion of the left arm is cancelled after removing *CoM* task. For *motion 2.b* the motion of the left arm is not cancelled by *CoM* projection, but is cancelled after removing *Left grab* task. Fig. 14 shows how the motion of the right arm evolves when removing motions due to a task. It can be seen that the right arm moves because of the *Right grab* and *CoM* tasks. Fig. 15 shows the evolution of the quantity of motion after each task selection on the two motions. The quantity decreases after successive projections, until the remaining motion is mainly sensor noise.

C. Experiment 4: Gaze VS Chest

The considered motions are *motion 5.a* and *motion 5.b*. In the first case, the robot is looking at its right hand while grabbing something. In the second case, the robot is grabbing something with its right hand while maintaining its chest in its current orientation. Fig. 16 shows the final posture of the robot for those motions.

Table VI summarizes the results of the task-selection algorithm. Despite noise, the active tasks are detected. The two motions are properly disambiguated. Finally, all tasks are properly discovered, as proved by the very small residue.

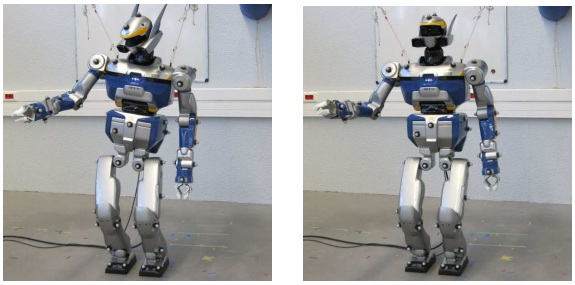


Fig. 16. Final position for *motion 5.a* and *motion 5.b*.

Reference	Detected	$\int \|\dot{q}(t)\|^2 dt$	$\int \|P\dot{q}(t)\|^2 dt$
CoM Gaze Right grab Twofeet	CoM Gaze Right grab Twofeet	0.320001	0.0785168
Chest CoM Right grab Twofeet	Chest CoM Right grab Twofeet	0.216742	0.0516134

TABLE VI
RESULTS OF THE TASK SELECTION ALGORITHM FROM THE ANALYSIS OF *motion 5.a* AND *motion 5.b* ON THE REAL ROBOT.

Fig. 17 shows the evolution of the norm of the motions after the projection into the null space of the tasks : *Right grab*, *Gaze*, *Twofeet* and *CoM* for *motion 5.a*. And *Right grab*, *Twofeet*, *CoM* and *Chest* for the *motion 5.b*. As in the previous experiments, the projection decreases strictly the quantity of movement at each algorithm iteration. The quantity of movement finally remaining at the end of the algorithm loop is clearly mainly due to noise.

VII. CONCLUSION

This article describes a complete method to identify which tasks are involved in an observed movement based uniquely on the observed trajectory. The analysis is driven by the knowledge of what a task is and how it behaves. The analyzed movement is supposed to be generated by a set of controllers belonging to a known pool of tasks. The task recognition problem is then tackled by reverse engineering the motion. The observed trajectory is analyzed in each known task space to decide which are the active tasks. The method does not rely on the nature of the behavior of a task. Therefore any control laws used to generate a motion can be similarly used to characterize an observed movement.

The method has been successfully applied in different scenarios to discriminate similar-looking motions on a real robot. Those motions were built to be especially ambiguous in order to illustrate the efficiency of the method.

The experiments were limited to the analysis of robotic movement. In particular, the experiments only consider exponential decrease as behaviors. However, it is directly possible to handle other task models, for example, minimum jerk, in order to analyze human trajectories.

APPENDIX A
PROOF OF THE REVERSE IMPLICATION

This appendix proves that $\widehat{\dot{\mathbf{e}}}_{a|b} = \dot{\mathbf{e}}_a^*$. The proof is in two part, proving that the first left-coefficient (denoted $\mathbf{C}_1$) of the sum is the identity, and that the second left-coefficient (denoted $\mathbf{C}_2$) is null.

A. Proof that $\mathbf{C}_1 = \mathbf{J}_a\mathbf{P}_b\mathbf{J}_a^+ - \mathbf{J}_a\mathbf{P}_b(\mathbf{J}_b\mathbf{P}_a)^+\mathbf{J}_b\mathbf{J}_a^+ = \mathbf{I}$

The projection operator $\mathbf{P}_b$ can be expanded using the definition of the projector $\mathbf{P} = \mathbf{I} - \mathbf{J}^+\mathbf{J}$:

$$\begin{aligned}\mathbf{C}_1 =& \mathbf{J}_a\mathbf{J}_a^+ - \mathbf{J}_a\mathbf{J}_b^+\mathbf{J}_b\mathbf{J}_a^+ \\ &- \mathbf{J}_a(\mathbf{J}_b\mathbf{P}_a)^+\mathbf{J}_b\mathbf{J}_a^+ + \mathbf{J}_a\mathbf{J}_b^+\mathbf{J}_b(\mathbf{J}_b\mathbf{P}_a)^+\mathbf{J}_b\mathbf{J}_a^+\end{aligned}$$

The first term is the identity by hypothesis on Task a. Since $(\mathbf{J}_b\mathbf{P}_a)^+ = \mathbf{P}_b(\mathbf{J}_b\mathbf{P}_a)^+$, the third term is null. And by hypothesis on Tasks a and b, $(\mathbf{J}_b\mathbf{P}_a)^+$ is a generalized inverse of $\mathbf{J}_b$, and thus $\mathbf{J}_b(\mathbf{J}_b\mathbf{P}_a)^+\mathbf{J}_b = \mathbf{J}_b$: the fourth term is equal to the second and simplifies. ■

B. Proof that $\mathbf{C}_2 = \mathbf{J}_a\mathbf{P}_b(\mathbf{J}_b\mathbf{P}_a)^+ + \mathbf{J}_a\mathbf{J}_b^+ = \mathbf{0}$

Similarly, the projector $\mathbf{P}_b$ is expanded:

$$\mathbf{C}_2 = \mathbf{J}_a(\mathbf{J}_b\mathbf{P}_a)^+ - \mathbf{J}_a\mathbf{J}_b^+\mathbf{J}_b(\mathbf{J}_b\mathbf{P}_a)^+ + \mathbf{J}_a\mathbf{J}_b^+$$

As previously, the first term is null, this $(\mathbf{J}_b\mathbf{P}_a)^+$ is a generalized inverse of $\mathbf{J}_b$, the second term is equal to the third and simplifies. ■

REFERENCES

[1] S. Hak, N. Mansard, and O. Stasse, "Humanoid robot task recognition from movement analysis," in *International Conference on Humanoid Robots (Humanoids)*, (Nashville, USA), December 2010.

[2] E. Yoshida, A. Mallet, F. Lamiraux, O. Kanoun, O. Stasse, M. Poirier, P. Dominey, J. Laumond, and K. Yokoi, ""Give me the purple ball" - he said to HRP-2 N.14," in *International Conference on Humanoid Robots (Humanoids)*, (Pittsburg, USA), November 2007.

[3] O. Kanoun, J. Laumond, and E. Yoshida, "Planning foot placements for a humanoid robot : a problem of inverse kinematics," *International Journal of Robotics Research*, vol. 30, pp. 476–485, June 2010.

[4] C. Stauffer and W. Grimson, "Learning patterns of activity using real-time tracking," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 22, pp. 747–757, August 2000.

[5] H. Eng, K. Toh, A. Kam, J. Wang, and W. Yau, "An automatic drowning detection surveillance system for challenging outdoor pool environments," in *IEEE International Conference on Computer Vision*, (Nice, France), October 2003.

[6] J. Shotton, A. Fitzgibbon, M. Cook, T. Sharp, M. Finocchio, R. Moore, A. Kipman, and A. Blake, "Real-time human pose recognition in parts from a single depth image," in *IEEE Computer Vision and Pattern Recognition (CVPR)*, (Colorado Springs, USA), June 2011.

[7] I. Laptev, "On space-time interest points," *International Journal of Computer Vision*, vol. 64, pp. 107–123, September 2005.

[8] Z. Liu and S. Sarkar, "Effect of silhouette quality on hard problems in gait recognition," *IEEE Transactions on Systems, Man and Cybernetics, Part B*, vol. 35, pp. 170–183, April 2005.

[9] V. B. Zordan and N. C. V. D. Horst, "Mapping optical motion capture data to skeletal motion using a physical model," in *Symposium on Computer Animation*, (San Diego, USA), July 2003.

[10] I. Oikonomidis, N. Kyriazis, and A. Argyros, "Efficient model-based 3d tracking of hand articulations using kinect," in *British Machine Vision Conference*, (Dundee, UK), September 2011.

[11] J. Cutting and L. Kozlowski, "Recognizing friends by their walk: Gait perception without familiarity cues," *Bulletin of the Psychonomic Society*, vol. 9, pp. 353–356, 1977.

[12] A. Kale, A. Sundaresan, A. Rajagopalan, N. Cuntoor, A. Roy-Chowdhury, V. Kruger, and R. Chellappa, "Identification of humans using gait," *IEEE Transactions on Image Processing*, vol. 13, pp. 1163–1173, September 2004.

[13] M. Karg, K. Khnlenz, and M. Buss, "Recognition of affect based on gait patterns," *IEEE Transactions on Systems, Man and Cybernetics, Part B*, vol. 40, pp. 1050–1061, August 2010.

[14] S. Schaal, A. Ijspeert, and A. Billard, "Computational approaches to motor learning by imitation," *Philosophical Transaction of the Royal Society of London, series B*, vol. 358, pp. 537–547, March 2003.

[15] E. Drumwright and M. Matarić, "Generating and recognizing free-space movements in humanoid robots," in *IEEE/RAS Intelligent Robots and Systems (IROS)*, (Las Vegas, USA), October 2003.

[16] E. Drumwright, O. C. Jenkins, and M. Matarić, "Exemplar-based primitives for humanoid movement classification and control," in *International Conference on Robotics and Automation (ICRA)*, (New Orleans, USA), April 2004.

[17] J. Pearl, *Probabilistic Reasoning in Intelligent Systems*. Morgan Kaufmann Publisher Inc, 1988.

[18] L. Kaelbling, M. Littman, and A. Cassandra, "Planning and acting in partially observable stochastic domains," *Artificial Intelligence*, vol. 101, pp. 99–134, 1998.

[19] S. S. J. Peters, "Reinforcement learning of motor skills with policy gradients," *Neural Networks*, vol. 21, pp. 682–697, May 2008.

[20] S. Calinon, F. D'halluin, E. Sauser, D. Caldwell, and A. Billard, "Learning and reproduction of gestures by imitation : An approach based on hidden markov model and gaussian mixture regression," *IEEE Robotics and Automation Magazine*, vol. 17, pp. 44–54, June 2010.

[21] T. Inamura, Y. Nakamura, and I. Toshima, "Embodied symbol emergence based on mimesis theory," *International Journal of Robotics Research*, vol. 23, pp. 363–377, April 2004.

[22] J. Gu, X. Ding, S. Wang, and Y. Wu, "Action and gait recognition from recovered 3-d human joints," *IEEE Transactions on Systems, Man and Cybernetics, Part B*, vol. 40, pp. 1021–1033, August 2010.

[23] J. Kwon and F. Park, "Natural movement generation using hidden markov models and principal components," *IEEE Transactions on Systems, Man and Cybernetics, Part B*, vol. 38, pp. 1184 – 1194, October 2008.

[24] Y. Liang, S. Shih, A. Shih, H. Liao, and C. Lin, "Learning atomic human actions using variable-length markov models," *IEEE Transactions on Systems, Man and Cybernetics, Part B*, vol. 39, pp. 268 – 280, February 2009.

[25] S. Nakaoka, A. Nakazawa, F. Kanehiro, K. Kaneko, M. Morisawa, H. Hirukawa, and K. Ikeuchi, "Learning from observation paradigm: Leg task models for enabling a biped humanoid robot to imitate human dances," *International Journal of Robotics Research*, vol. 26, pp. 829–844, August 2007.

[26] M. Mühlig, M. Gienger, J. Steil, and C. Goerick, "Automatic selection of task spaces for imitation learning," in *IEEE/RAS Intelligent Robots and Systems (IROS)*, (St Louis, USA), October 2009.

[27] C. Samson, M. Le Borgne, and B. Espiau, *Robot Control: the Task Function Approach*. Clarendon Press, Oxford, UK, 1991.

[28] B. Siciliano and J.-J. Slotine, "A general framework for managing multiple tasks in highly redundant robotic systems," in *IEEE International Conference on Advanced Robotics (ICAR)*, (Pisa, Italy), June 1991.

[29] Y. Nakamura, H. Hanafusa, and T. Yoshikawa, "Task-priority based re-dundancy control of robot manipulators," *International Journal of Robotics Research*, vol. 6, pp. 3 – 15, June 1987.

[30] A. Liégeois, "Automatic supervisory control of the configuration and behavior of multibody mechanisms," *IEEE Transactions on Systems, Man and Cybernetics*, vol. 7, pp. 868–871, December 1977.

[31] N. Mansard and F. Chaumette, "Task sequencing for high level sensor-based control," *IEEE Transactions on Robotics*, vol. 23, pp. 60–72, February 2007.

[32] S. Chiaverini, "Singularity-robust task-priority redundancy resolution for real-time kinematic control of robot manipulators," *IEEE Transactions on Robotics and Automation*, vol. 13, pp. 398 – 410, June 1997.

[33] C. Lawrence, J. L. Zhou, and A. L. Tits, "User's guide for CFSQP version 2.5: A C code for solving (large scale) constrained nonlinear (minimax) optimization problems, generating iterates satisfying all inequality constraints," Tech. Rep. TR-94-16RL, Institute for Systems Research, University of Maryland, 1997.