Reverse Control for Humanoid Robot Task Recognition
Sovannara Hak, Nicolas Mansard, Olivier Stasse, Jean-Paul Laumond

To cite this version:
Sovannara Hak, Nicolas Mansard, Olivier Stasse, Jean-Paul Laumond. Reverse Control for Humanoid Robot Task Recognition. IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics, Institute of Electrical and Electronics Engineers, 2012, 42 (6), pp.1524-1537. 10.1109/TSMCB.2012.2193614. hal-00697272

HAL Id: hal-00697272
https://hal.archives-ouvertes.fr/hal-00697272
Submitted on 15 May 2012

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L’archive ouverte pluridisciplinaire HAL, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d’enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.
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.

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.
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 \( \dot{q} \), where \( q \) is the robot configuration vector. A task is defined by a vector space \( e \) and by the reference behavior \( \dot{e}^* \) 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 \( J = \frac{\partial e}{\partial q} \). Various typical behaviour can be chosen for \( \dot{e}^* \). Typically, we will use in the following an exponential decrease, set by

\[
\dot{e}^* = -\lambda e
\]

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

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

\[
\dot{q} = J^+ \dot{e}^* + P z \tag{2}
\]

where \( J^+ \) is the least-square inverse of \( J \), \( P = I - J^+ J \) is the projection operator onto the null space of \( J \) and \( z \) is any secondary criterion. \( P \) ensures a decoupling of the task with respect to \( z \). Using \( z \) 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 \( i \) if \( i > j \). The recursive formulation of the control law is proposed by [28]:

\[
q_i = q_{i-1} + (J_i P_{i-1}^A)^+ (\dot{e}^*_i - J_i q_{i-1}), \quad i = 1 \ldots n \tag{3}
\]

with \( q_0 = 0 \) and \( P_{i-1}^A \) is the projector onto the null space of the augmented Jacobian \( J_i^A = (J_1, \ldots, J_i) \). Joint velocities realizing all the tasks is \( \dot{q}^*_i = \dot{q}_i \). 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 \( \lambda \) 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 \( P \) defined in (2) (no algorithmic singularities [32]).

We call the set of possible tasks the \( \text{task pool} \) and we assume that the behavior model \( \dot{e}^* \) of each task 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.

The algorithm is showed in Alg. 1. The joint velocity trajectory of the observed motion is denoted \( \dot{\tilde{q}}(t) \). \( \tilde{P}\dot{q}(t) \) denotes the successive projections of the reference motion. Before the first iteration, \( \tilde{P}\dot{q}(t) \) 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. \( \text{activePool} \) denotes the set of tasks selected during the algorithm.

If the observed motion is exactly generated by a SoT, the resulting trajectory \( \tilde{P}\dot{q} \) 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. \( \epsilon \) 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 \( \text{projection}(i, \tilde{q}(t)) \) computes the projection of velocity onto the null space of the task \( i \) and apply the projection to the motion. The procedure \( \text{taskFitting}(i, \text{activePool}) \) handles the curve fitting of the observed motion and the theoretical motion. The process is detailed in section IV-D.

<table>
<thead>
<tr>
<th>Algorithm 1 Task selection algorithm</th>
</tr>
</thead>
<tbody>
<tr>
<td>1: Input: ( \tilde{q}(t) )</td>
</tr>
<tr>
<td>2: Output: ( \text{activePool} )</td>
</tr>
<tr>
<td>3: ( \tilde{P}\dot{q}(t) \leftarrow \dot{\tilde{q}}(t) )</td>
</tr>
<tr>
<td>4: while ( \int</td>
</tr>
<tr>
<td>5: for task ( i = 1...n ) do</td>
</tr>
<tr>
<td>6: ( r_i \leftarrow \text{taskFitting}(i, \text{activePool}) )</td>
</tr>
<tr>
<td>7: end for</td>
</tr>
<tr>
<td>8: ( i_{select} \leftarrow \arg\min(r_i) )</td>
</tr>
<tr>
<td>9: ( \text{activePool}.\text{push}(i_{select}) )</td>
</tr>
<tr>
<td>10: ( \tilde{P}\dot{q}(t) \leftarrow \text{projection}(i_{select}, \tilde{P}\dot{q}(t)) )</td>
</tr>
<tr>
<td>11: end while</td>
</tr>
</tbody>
</table>

\(^1\)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 \( \cdot \) while references and desired values are denoted \( \cdot^* \).
tasks to cancel. For every time $t$ of the motion time interval:

$$\mathbf{P}\hat{\mathbf{q}}(t) \leftarrow \mathbf{P}_{\text{select}}(t)\mathbf{P}\hat{\mathbf{q}}(t)$$

(4)

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

$$\mathbf{P}(t) = \mathbf{P}(t) - (J_{\text{select}}(t)\mathbf{P}(t))^+ (J_{\text{select}}(t)\mathbf{P}(t))$$

The projector $\mathbf{P}(t)$ is initialized to $\mathbf{P}(t) = \mathbf{I}$. The remaining motion after projection $\mathbf{P}\hat{\mathbf{q}}(t)$ 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-detected 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 $a$ and $b$. The control law to execute both tasks is given by:

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

(5)

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

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

(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 $\mathbf{J}_b$:

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

(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 $\hat{\mathbf{e}}(t)$ the trajectory due to the current $\mathbf{P}\hat{\mathbf{q}}$ in the observed-task space. The quantification of the relevance of the given task $i$, with respect to the current motion $\mathbf{P}\hat{\mathbf{q}}$, 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:

$$\mathbf{x} = \arg \min \int \frac{\|\hat{\mathbf{e}}(t) - \hat{\mathbf{e}}_x(t)\|^2 dt}{\int \|\hat{\mathbf{e}}(t)\|^2 dt}$$

(8)

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 $\hat{\mathbf{e}}$ 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. $\hat{\mathbf{e}}$ is then obtained by the kinematic model and joint trajectories of the robot: $\mathbf{J}\hat{\mathbf{q}}$. However, proceeding so would lead to (7), where the coupling with previously selected tasks appears. The projected motion is then augmented with coupling compensation:

$$\hat{\mathbf{e}}(t) = \mathbf{J}_b \mathbf{P}(t) \hat{\mathbf{q}}(t) + \mathbf{J}_b \mathbf{P}(t) + \hat{\mathbf{e}}_a(t)$$

(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 $\hat{\mathbf{e}}_i$. The motion induced in Task $i$ space by the original motion $\hat{\mathbf{q}}$ is denoted $\hat{\mathbf{e}}_i$, while the motion induced in Task $j$ after removing the motion from Task $i$ $\mathbf{P}_j \hat{\mathbf{q}}$ is denoted $\hat{\mathbf{e}}_{ij}$. 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 $\mathbf{e}_i = \mathbf{e}_{ij} = \mathbf{e}_j$, 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 $\hat{\mathbf{e}}_a = \mathbf{J}_a \hat{\mathbf{q}} = \hat{\mathbf{e}}_a$. After the removal of Task $a$, the observation in Task $b$ space is obtained from (9):

$$\hat{\mathbf{e}}_{b|a} = \mathbf{J}_b \hat{\mathbf{q}} + \mathbf{J}_b \mathbf{J}_a^+ \hat{\mathbf{e}}_a = \mathbf{J}_b (\mathbf{J}_b \mathbf{P}_a)^+ (\hat{\mathbf{e}}_b - \mathbf{J}_b \mathbf{J}_a^+ \hat{\mathbf{e}}_a) + \mathbf{J}_b \mathbf{J}_a^+ \hat{\mathbf{e}}_a$$

implicating directly $\hat{\mathbf{e}}_{b|a} = \hat{\mathbf{e}}_b$, 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 $\hat{\mathbf{e}}_a = \hat{\mathbf{e}}_a$.
The observation obtained in Task \( b \) space is independent from the projection in the null space of Task \( a \).

2) \textit{Detecting Task }\( b \), \textit{then Task }\( a \): At the first iteration, the observation in Task \( b \) space is directly

\[
\hat{\mathbf{e}}_b = J_b \hat{\mathbf{J}}_a^b \mathbf{e}_a^b + J_b (J_b \hat{\mathbf{P}}_a)^+ (\hat{\mathbf{e}}_b^* - J_b \hat{\mathbf{J}}_a^b \mathbf{e}_a^b)
\]

which implies \( \hat{\mathbf{e}}_b = \hat{\mathbf{e}}_b^* \) since \( J_b (J_b \hat{\mathbf{P}}_a)^+ = \mathbf{I} \).

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:

\[
\hat{\mathbf{e}}_{a|b} = J_a \hat{\mathbf{P}}_b J_a^b \mathbf{e}_a^b + J_a \hat{\mathbf{P}}_b (J_b \hat{\mathbf{P}}_a)^+ (\hat{\mathbf{e}}_b^* - J_b \hat{\mathbf{J}}_a^b \mathbf{e}_a^b) + J_a \hat{\mathbf{J}}_a^b \hat{\mathbf{e}}_b^*
\]

\[
= (J_a \hat{\mathbf{P}}_b J_a^b - J_a \hat{\mathbf{P}}_b (J_b \hat{\mathbf{P}}_a)^+ J_b \hat{\mathbf{J}}_a^b) \hat{\mathbf{e}}_a^b
\]

\[
+ (J_a \hat{\mathbf{P}}_b (J_b \hat{\mathbf{P}}_a)^+ + J_a \hat{\mathbf{J}}_a^b) \hat{\mathbf{e}}_b^*
\]

We show that this complex sum is in fact simply \( \hat{\mathbf{e}}_a^* \) (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.

\section*{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.\textcolor{red}{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.}

\subsection*{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 free flyer. Every motions start from the half-sitting pose. As classically done in inverse kinematics, the under-actuation of the free flyer is resolved by constraining the left foot to be on the ground.

The set of tasks considered in those experiments are:

- \textit{CoM} : the center of mass of the robot is constraint to maintain the static balance (3 DOF)
- \textit{Gaze} : the robot looks at one point in the Cartesian space (2 DOF)
- \textit{Twofeet}: both feet stay flat with regard to each other (6 DOF)
- \textit{Left/right grab} : either the left or right hand of the robot reaches a point defined in the Cartesian space (3 DOF)
- \textit{Screw} : similar to the \textit{grab} task, but the desired position has to be reached with a defined orientation (6 DOF)
- \textit{Head} : the head of the robot is constrained in position and orientation (6 DOF)
- \textit{Chest} : the chest of the robot is constrained in orientation (3 DOF)

\begin{table}[h]
\centering
\begin{tabular}{|c|c|}
\hline
Motion 1 & \begin{tabular}{c}
Right grab \\
Left grab \\
Gaze \\
CoM \\
Twofeet
\end{tabular} \\
\hline
Motion 2 & \begin{tabular}{c|c}
Right grab & \begin{tabular}{c}
Left grab \\
Right grab \\
CoM \\
Twofeet
\end{tabular} \\
\hline
Motion 3 & \begin{tabular}{c}
Gaze \\
Right grab \\
CoM \\
Twofeet
\end{tabular} \\
\hline
Motion 4 & \begin{tabular}{c|c}
Gaze & \begin{tabular}{c}
Left Screw \\
CoM \\
Twofeet
\end{tabular} \\
\hline
Motion 5 & \begin{tabular}{c}
Right grab & \begin{tabular}{c}
Right grab \\
Chest \\
CoM \\
Twofeet
\end{tabular} \\
\hline
\end{tabular}
\end{tabular}
\caption{The table of considered motions and tasks.}
\end{table}
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. 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 $\int \|Pq(t)\|^2 dt > \epsilon$ 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-
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, i.e. 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.

Fig. 8. Experiment 2: Evolution of the trajectory $\dot{q}$ 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.
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, due to numerical noise. The following section will consider the reverse-engineering task recognition.

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, i.e. 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.

| Reference | Detected | $\int ||q(t)||^2 dt$ | $\int ||Pq(t)||^2 dt$ |
|-----------|-----------|-------------------|-------------------|
| CoM Gaze  | CoM Gaze  | 0.619266          | 0.00245631        |
| Grab Twofeet | Grab Twofeet | 0.717041          | 0.00344557        |

| Reference | Detected | $\int ||q(t)||^2 dt$ | $\int ||Pq(t)||^2 dt$ |
|-----------|-----------|-------------------|-------------------|
| CoM Gaze  | CoM Gaze  | 0.534478          | 0.0545944         |
| Screw Twofeet | Screw Twofeet | 0.558084          | 0.0023297         |

TABLE III

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 classicaly done by optimizing the distance between the transformation matrix $W_R_{q_i}(q)$ from the robot origin to each joint of the robot and the measured transformation matrix $\hat{W}_R_{q_i}(t)$ using the robot joint limits as constraints.

$$\hat{q}(t) = \arg \min_q \sum_{i=1}^{n} ||\hat{W}_R_{q_i}(t) \odot W_R_{q_i}(q)||^2$$

subject to $q_{\text{min}} \leq q_i \leq q_{\text{max}}, i = 1..n$ (11)

where $q$ is the robot joint configuration vector, $\odot$ is the distance operator in $\text{SO}(3)$, $W_R_{q_i}(q)$ is computed using the kinematic model of the robot, and $\hat{W}_R_{q_i}$ is obtained by:

$$\hat{W}_R_{q_i}(t) = W_{R_{WC}} \times W_{C_{R_M}}(t) \times M_{R_{q_i}}$$

where $W_{R_{WC}}$ and $M_{R_{q_i}}$ 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. $W_{C_{R_M}}(t)$ 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 $\hat{q}$ trajectories given by (12).
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.

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.

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.
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.

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.

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.
and motion 2.a shows the result of the identification algorithm for algorithm. Despite noise, the active tasks are detected. The robot for those motions.

Fig. 16 shows the final posture of the robot. Those motions were built to be especially ambiguous to discriminate similar-looking motions on a real robot. Those motions were built to be especially ambiguous to discriminate similar-looking motions on a real robot. 

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.

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 \( \hat{e}_{a|b} = \hat{e}_b \). The proof is in two part, proving that the first left-coefficient (denoted \( C_1 \)) of the sum is the identity, and that the second left-coefficient (denoted \( C_2 \)) is null.

A. Proof that \( C_1 = J_a P_b J_a^+ - J_a P_b (J_a P_a)^+ J_a J_a^+ = I \)

The projection operator \( P_b \) can be expanded using the definition of the projector \( P = I - J^+ J \):

\[
C_1 = J_a J_a^+ - J_a J_a^+ J_a J_a^+ - J_a (J_a P_a)^+ J_a J_a^+ = J_a J_a^+ - J_a J_a^+ J_a J_a^+ - J_a (J_a P_a)^+ J_a J_a^+ = 0
\]

The first term is the identity by hypothesis on Task \( a \). Since \( (J_a P_a)^+ = P_b (J_a P_a)^+ \), the third term is null. And by hypothesis on Tasks \( a \) and \( b \), \( (J_a P_a)^+ \) is a generalized inverse of \( J_a \), and thus \( J_a (J_a P_a)^+ J_a = J_a \); the fourth term is equal to the second and simplifies.

B. Proof that \( C_2 = J_a P_b J_a P_a^+ + J_a J_a^+ = 0 \)

Similarly, the projector \( P_b \) is expanded:

\[
C_2 = J_a J_a^+ - J_a J_a^+ J_a J_a^+ - J_a (J_a P_a)^+ J_a J_a^+ = 0
\]

As previously, the first term is null, this \( (J_a P_a)^+ \) is a generalized inverse of \( J_a \), the second term is equal to the third and simplifies.

REFERENCES