Humanoid Robot Task Recognition from Movement Analysis

Sovannara Hak, Nicolas Mansard, Olivier Stasse

Unification of Task Identification and Control for a Humanoid Robot

Sovannara Hak, Nicolas Mansard, Olivier Stasse

***Abstract*— We present a method using simple mechanisms to perform a real time imitation on a humanoid robot. Instead of focusing on the fidelity of the motion, we will rather focus on what controllers have to be activated to perform a desired motion. The originality of our approach is to work in the task spaces of the robot in order to solve simultaneously the identification and the execution of a motion. With this approach, we bypass the classical phase of the transformation from the imitated model to the imitant model. The tasks identification method consists in proposing several model of the motion, and then computing the parameters of this model with a least square optimization.**

The motivation behind the use of a simple technique is to be as much efficient as possible regarding computation time, as this work is a preliminary step for the resolution of the reactive control problem for a humanoid robot where more complex mechanism have to be used. We formulate it as the problem of deciding very quickly which sequence of controllers, has to be activated (this choice will be based on motion imitation) to realize a motion.

Although in this work we consider the problem where a robot imitate a robot, we investigate the possibility of extending our method to sequential motion imitation and a case where a robot imitate a human.

I. Introduction

One of the challenging task for humanoid robot is to achieve a reactive behavior. This involves the robot taking the right decision very quickly. We want to achieve this reactive behavior in an motion imitation application. An instructor performs a motion, and our robot replicate this movement. We will focus on the task space which offers many advantages over the joint angle space.

A. Related work

Imitation based motion are usually decomposed in two part: the recognition part which deals with the evaluation of what to imitate. And the action part which deals with the problem of determining how to imitate. Statistical and learning approach had been widely use for imitation purpose, as it can be seen in [1] which quickly reviews the statistical and mathematical tools used to tackle imitation problems.

The work presented in [2], [3] shows how a behavior based control can be achieved for a humanoid robot. That is to say that they choose beforehand what kind of motion they want to perform (jab, hook, elbow, shield and uppercut). They modeled their behaviors with joint angles trajectories examples and applied a dimensional reduction to have a significant clusterization. The recognition part is handled by a Bayesian classifier which recognize a trajectory in joint or Cartesian space. To perform the reference motion, they must interpolate the known examples : their *behavior controller* can not be directly applied to the robot. The methods presented is not adapted for reactive behavior if the robot have a lots of degree of freedom or a large vocabulary of motion.

In [4], Gaussian mixture model are used to extract from several demonstrations a representation of a general motion in a redundant space. This space is composed of articular configuration of the robot, Cartesian positions of end effector and the error between end effectors and objective. A reduction of dimension of the data acquired in the demonstration is then performed. Then using an imitation metric, an optimal motion is made up from the set of demonstrations. It is important to notice that the output of the reduction algorithm depends on the nature of the tasks being performed. That is to say that the important features of a tasks are not invariant. The set of demonstrations (also called the learning set of motion) is acquired by kinesthetic demonstrations, in other words, the demonstrator perform a motion on the robot which motors are set to a passive mode. The motors encoder will then record the data relevant to each degree of freedom of the robot.

The kinesthetic approach to acquire motion data is also used in [5]. They want to imitate objective based motion, such as reaching for grasping an object or releasing an object in a box. The point is to catch the invariant of the task, which here is to actually grasp (or release) the object, knowing that positions may vary from an instance of the task to another. A model of the final configuration given the position of the objective is built. In order to tackle the problem of perturbations or changes in initial conditions, a stereo vision input is used. The global system also allows the use of several types of controllers (velocity/acceleration).

The work done in [6] map observations of a movement from sensors to a task space that will best represent the movement (for example a task space will generalize a motion better than others for a particular task). This is done by computing some score functions inspired by neuroscience. A task space pool is created beforehand, and for each task space, the score is computed. Then the system choose the task space with the best score. In other words, the system

Sovannara Hak and Nicolas Mansard, CNRS; LAAS; 7 avenue du colonel Roche, F-31077 Toulouse, France Université de Toulouse; UPS, INSA, INP, ISAE ; sovannara.hak@laas.fr, nicolas.mansard@laas.fr

Olivier Stasse, CNRS-AIST, JRL (Joint Robotics Laboratory), UMI 3218/CRT, Intelligent Systems Research Institute AIST Central 2, Umezono 1-1-1 Tsukuba, Ibaraki 305-8568 Japan olivier.stasse@aist.go.jp

choose amongst a set of space the one which will best describe the task. The criterion taken into account for the task space selection is the saliency of the object that is manipulated, the variance of the dimension of a space during several demonstration, and some heuristics that express that uncomfortable or exhausting motion are more relevant to a task. In other words, uncomfortable motion is an important feature of the global motion, and then it must be replicated.

B. Hypothesis and outline

Our work is based on the *Stack of Tasks* structures : given a set of tasks to achieve (which are stacked by order of priority) and a model of a robot, a command law is computed and then applied to the robot. This relationship between the task space and the joint space of the robot will be exploited to perform an imitation by working in the task space instead of working in the joint space. The task space refers to the finite and discrete set of all possible tasks to achieve.

However, some assumptions have to be made for the imitation application. First, the motion to imitate has to be generated by a *stack of tasks* structure with the same task space, so that the relationship between the task space and the joint space is still valid. Then, all tasks involved in a motion belong to a known set of possible tasks. This assumption will also assure that the motion to replicate can really be executed by the imitator.

The second assumption is that the kinematic models of both imitator and teacher have to be known, because they are needed to compute a control law from a stack of tasks.

This work is not focused in the problem of estimation of a motion, and so, we assume that a system will provide us motion data in the desired space. That is to say that we know the joint angle trajectory or the velocity of the motion of the entity to imitate.

Our major contribution is a method that identify the tasks to be used and their control parameters in an efficient manner and manage to discriminate very close motion. This is realized by using an analytical description of the task exponential decay, and performing the recognition directly in the task space instead of the articular space.

The next sections describe the base of our work : the task-function formalism and how to use this formalism along with an optimization approach to detect what is performed by the imitated model; description of some experimentations; future works discussion where the application of the method in the case where the motion to imitate is performed by a human, and the extension of this method to a sequential problem are briefly investigated.

II. Stack of tasks

In order to control a robot, a sensory-motor control approach based on task component can be used instead of a trajectory planner. A task function [7] is an elegant approach to produce intuitively sensor-based robot objectives. Based on the redundancy of the system, the approach can be extended to consider a hierarchical set of tasks [8].

A. Task function formalism

Defining the motion of the robot in terms of task consists in choosing several control laws to be applied on a subspace of the robot degree of freedom. A task is defined by a vector $\mathbf{e}$ (which is typically the error between a signal $\mathbf{s}$ and its desired value $\mathbf{e} = \mathbf{s}^* - \mathbf{s}$) The Jacobian of the task is noted $\mathbf{J} = \frac{\partial \mathbf{e}}{\partial \mathbf{q}}$ where $\mathbf{q}$ is the robot configuration vector. We consider that the robot input control is the velocity $\dot{\mathbf{q}}$: $\dot{\mathbf{e}} = \mathbf{J}\dot{\mathbf{q}}$. Considering a reference behavior $\dot{\mathbf{e}}$ to be executed in the task space,

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

the control law to be applied on the robot whole body is given by the least-square solution:

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

where $\mathbf{J}^+$ is the least-square inverse of $\mathbf{J}$, $\mathbf{P} = \mathbf{I} - \mathbf{J}^+\mathbf{J}$ is the null-space of $\mathbf{J}$ and $\mathbf{z}$ is any secondary criterion. $\mathbf{P}$ ensures a decoupling of the task with respect to $\mathbf{z}$, which can be extended recursively to a set of *n* tasks, each new task being fulfilled without disturbing the previous ones:

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

where $\dot{\mathbf{q}}_0 = 0$ and $(\mathbf{J}_i \mathbf{P}^A_{i-1})^+$ is the projector onto the null-space of the augmented Jacobian $\mathbf{J}^A_i = (\mathbf{J}_1, \dots \mathbf{J}_i)$. The robot joint velocity realizing all the tasks is $\dot{\mathbf{q}} = \dot{\mathbf{q}}_n$. A complete implementation of this approach is proposed in [9] under the name *Stack of Tasks* (SoT). The structure enables to easily add or remove a task, or to swap the priority order between two tasks, during the control while preserving the continuity of the control law. Constraints (such as joints limit) can be taken into account locally.

Using this formalism, the key idea is that imitate a motion is equivalent to execute the same stack of tasks. Because of the relationship between the tasks and trajectories as illustrated in the figure 1. Then the imitation problem becomes a problem of finding all the tasks relevant to the motion.

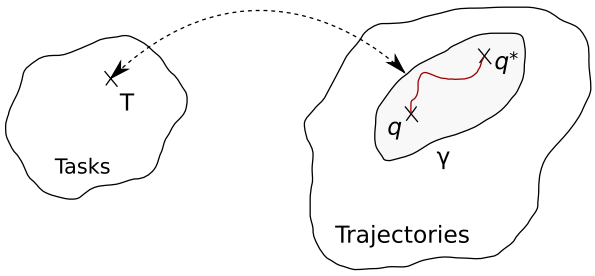

Fig. 1. Relationship between tasks and trajectories.

Detecting all tasks in a row may be intractable. That is why a method to decouple the tasks is needed. This decoupling will be provided by the projection operator.

B. The null-space projection

The null-space projectors have a central role in our work. Most of the time, with the tasks formulation, the robot does not execute only one single task (for example, one task for the gaze of the robot, and another to an end effector, some constraints...). In order to perform an imitation, we then want to be able to identify all the tasks being performed by the imitated entity. The idea to tackle this problem is to detect the most significant task in the motion, and then cancel it using the properties of the null-space projector. Doing this recursively will allow us to identify all the tasks being performed.

We use the recursive formula proposed by [10] to compute the null-space projector:

$$\begin{cases} \mathbf{P}_0 & = \mathbf{I} \\ \mathbf{P}_i & = \mathbf{P}_{i-1} - (\mathbf{J}_i \mathbf{P}_{i-1})^{+} (\mathbf{J}_i \mathbf{P}_{i-1}) \end{cases} \tag{4}$$

where $\mathbf{I}$ is the identity matrix, $\mathbf{J}_i$ is the Jacobian matrix of the task i. Once the projector has been computed, we use it to project the joint angle trajectory in the null-space of a task. The effect of the projection, is that the sub-part of the robot involved in the task will have its motion cancelled. This feature will be used in the algorithm described below.

III. Imitation

A. Task selection algorithm

We want to reconstruct the stack of tasks of the imitated model. In other word, to select relevant tasks or controllers that will perform an imitation. We iteratively select tasks until we consider all tasks detected. This stop criterion will be based on the null-space projection. Remember that projecting a motion onto the null-space of a task will cancel its relative motion. Each time a task is selected, the motion is projected onto the null-space of that task. The projected motion is then used to select another task. When the projected motion becomes non-significant, we consider that all tasks involved in the motion have been detected, and then stop the algorithm.

The task selection algorithm (where the task fitting function is described later) is shown below :

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

where P is a projector, r_i is the score of the cost function of the optimization, $activePool$ the set of tasks selected to perform the imitation.

The stop criterion will allow the discrimination of very close motions as it is illustrated in the next section.

B. Distinction between two close motions

An interesting challenge in motion detection, and therefore imitation, is to make the distinction between two motions involving different tasks but still *look* close.

In this section, two motions will be considered and we will show how two different instances of stack of tasks can be build from those two motions. The first one is a reaching task motion with the right hand. The reaching motion of the right hand has an influence on the left hand. That is because the target is out of reach of the robot arm, so the robot has to use his chest to reach it, the motion of the chest involves the motion of the left hand. In the remainder of the section, that motion will be called the *no task motion*.

The second motion is the same reaching task motion, but a second reaching task is defined on the left hand. The desired position of the left hand is the final position of the left hand in the first motion. In the remainder of the section, that motion will be called the *task motion*.

A video showing those two motions on the HRP-2 is downloadable[1].

The two motions are really close, and it is very difficult for someone to tell which motion involves a left and right hand task. In the first case, the motion of the left hand is not an important motion, as it is a side effect of the motion of the right hand. Whereas in the second case, the motion of the left hand is important, because the left hand has been controlled. Figure 2 plots the 3-D trajectories of the left hand in both case.

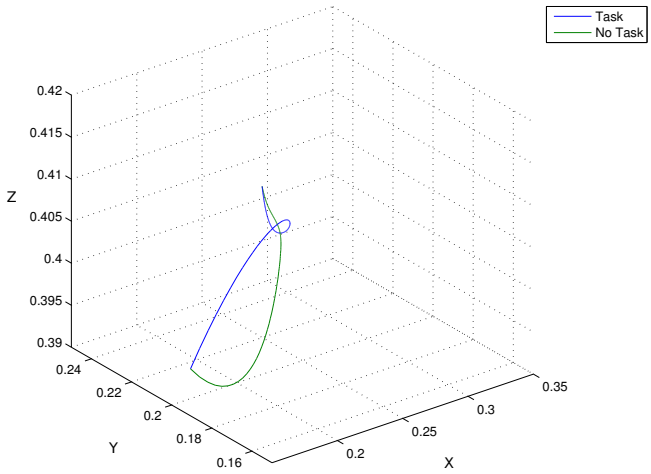

Fig. 2. 3D trajectories of the left hand.

The projection of the motion in the null-space of the right hand task will have different consequences for the *no task* and *task* motion. Projecting the *no task motion* into the null-space of the right hand will cancel the motion involved in this task. Including the non-controlled motion of the left hand. The norm of the remaining motion is therefore almost null, so the tasks selection algorithm stops.

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

On the other hand, when projecting the *task motion*, the remaining motion is not null. The remaining motion is the motion involved by the left hand task, and therefore its norm is not null. The algorithm will continue and look for other tasks. Figure 3 illustrates the norm of both projected motions.

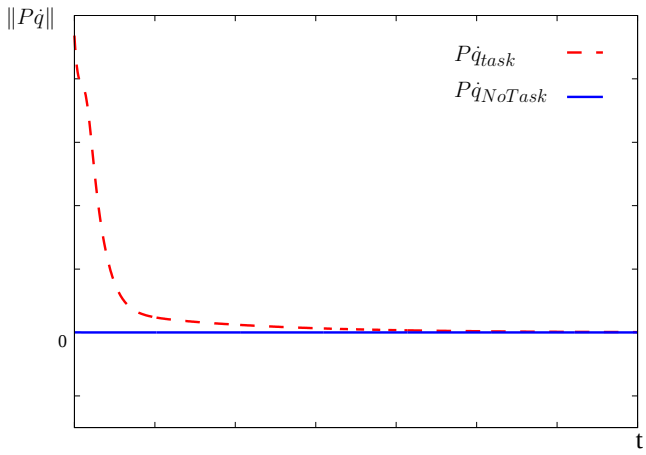

Fig. 3. Projection of the motion in the right hand null-space.

Being able to discriminate those two motions allow the use of classical techniques of sequential data processing, such as Hidden Markov Model, and particle filtering in the task space. This will be discussed in the future works section.

C. Task fitting by optimization

The point of this step is to estimate how much a task is relevant in the execution of a motion. And by the same time compute the parameters of that task.

Here, we compute the motion data that the execution of a task would produce using forward kinematics.

Our idea is to define a parametrized model of the task. It is important that the model will be a good representation of the task. We use a least-square optimization to find the parameters that will be appropriate for the model to fit the actual motion data. As the motion generated for a task vary for different parameters.

$$\mathbf{x}^* = \arg\min_{\mathbf{x}} \|\mathbf{p}^*(t) - \mathbf{p}_{\mathbf{x}}(t)\| \tag{5}$$

where $p^*(t)$ is the reference trajectory, $\mathbf{p}_{\mathbf{x}}(t)$ is the trajectory generated by the model using the parameters $\mathbf{x}$.

D. Model of a robot task

In the case where the imitant and imitated model are a humanoid robot, the task is modeled as an exponential decay with 3 input parameters.

$$\mathbf{p}_{\mathbf{x}}(t) = x_1 \mathrm{e}^{(-x_2 t)} + x_3 \tag{6}$$

This is motivated by the nature of the control of the robot.

An iteration of the optimization is time consuming as it computes the motion of the robot for each value of the iterate (parameters of the task). In order to minimize the number of iteration, and the computation time a good first guess of the task parameters have to be computed very quickly.

An analytical method is used to compute the first guess of the iterate. 3 data points $\mathbf{p}_1^*(t)$, $\mathbf{p}_2^*(t)$ and $\mathbf{p}_3^*(t)$ from the observation at times $t = 1, 2, 3$ are used to form a system of 3 equations which solutions are the first guess value.

$$\begin{aligned} 0 &= (\mathbf{p}_3 - x_3)^{k_2-k_1}(\mathbf{p}_1 - x_3)^{k_3-k_2} \\ &\quad -(\mathbf{p}_2 - x_3)^{k_3-k_1} \qquad (7) \\ x_2 &= -\frac{\ln \frac{\mathbf{p}_2 - x_3}{\mathbf{p}_1 - x_3}}{t_2 - t_1} \qquad (8) \\ x_1 &= \frac{\mathbf{p}_1 - x_3}{\mathrm{e}^{-x_2 t_1}} \end{aligned}$$

with $t_i = k_i \Delta t$ and Δt is the time step. The data points are close to each other. In the particular case of $t_i = 0$, we can obtain a simpler formulation for x_1 :

$$x_1 = \mathbf{p}_1 - x_3 \tag{9}$$

The third parameter of our model is the objective of the task. The equation 7 is a polynomial of x_3, because each exponent are strictly positives by definition as $k_1 > k_2 > k_3$. We then use a numerical solver to compute the parameters of the exponential. It is important to notice that those analytical solution must be used with caution. The order of the polynomial will depend on the choice of the data points. The consequence is that if we choose data points that are not close to each other, the order of the polynomial will increase, and we will encounter numerical noise in the resolution. On the other hand if we choose data points that are too close, the noise of the data will perturb our exponential fitting and it will not fit globally the data.

IV. Experimentations and Results

For all experimentations, we use a simulation of the humanoid robot HRP-2 having 30 degrees of freedom. We used the same robot for the imitant and imitated one.

A. Experiment 1 : robot trajectory fitting

In this simulated experiment, the imitated model performs a position reaching task with the right hand. The desired position was far enough to disturb the left hand. The disturbance is due to the fact that on a humanoid robot, the kinematic chain for the hands share the degrees of freedom coming from the chest. In other word, the motion of the right hand is controlled, whereas the motion of the left hand is not.

The trajectory of the hands were traced and artificially noised. The noised signal was given as input of the optimization, that will fit the data with a model of an exponential decay.

As it can be seen in the figures 4 and 5, the optimization find a very good set of parameters for the exponential decay of the right hand. Whereas for the left hand, the set of optimal parameters can't exactly fit the data point in range $[0 : 200]$. This is because the motion of the left hand was a motion here involved by the motion of the chest rather than a controlled motion.

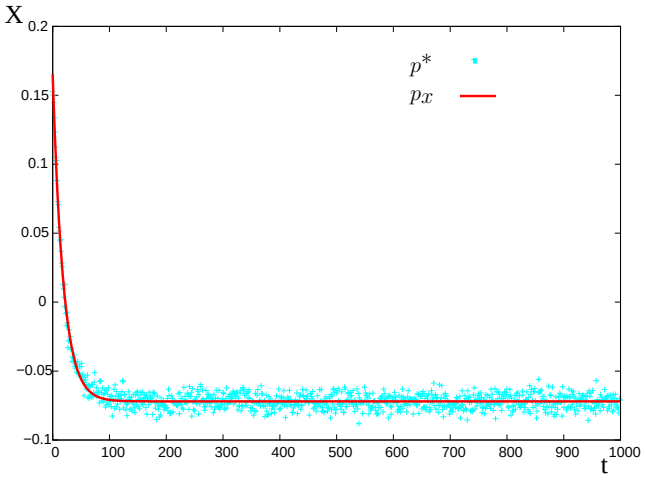

Fig. 4. Right hand : the task model fits

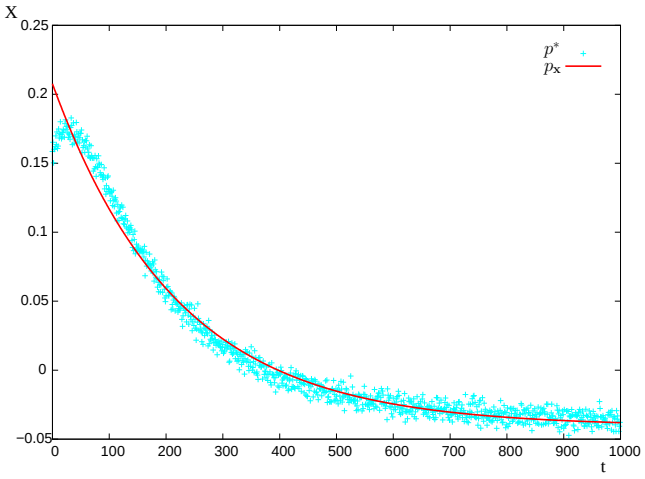

Fig. 5. Left hand : poor fit in [0:200]

B. Experiment 2 : Scale test

This experiment is intended to test our method against a larger scale problem. That is to say that the detection and fitting techniques are applied in the case where the size of the stack of tasks is close to a real situation. The imitated model here is a robot. It will perform a right and left hand tasks, a head task, a constraint of center of mass balancing and a task that will constraint the feet to keep a constant distance between them.

First a motion involving those tasks is generated in simulation. This motion is then artificially noised. The noised signal will be used for our experimentation. The motion is given to the detection program which select the task that is the best fitted by the optimization. Figure 6 shows the result of the fitting for the left hand task.

When a task is detected, the motion is projected into the null-space of that task. Figure 7 shows the norm of the joint angle velocity of the robot when the motion of the robot is successively projected in the null-spaces of the detected tasks. Each projection cancel a part of the motion, until the

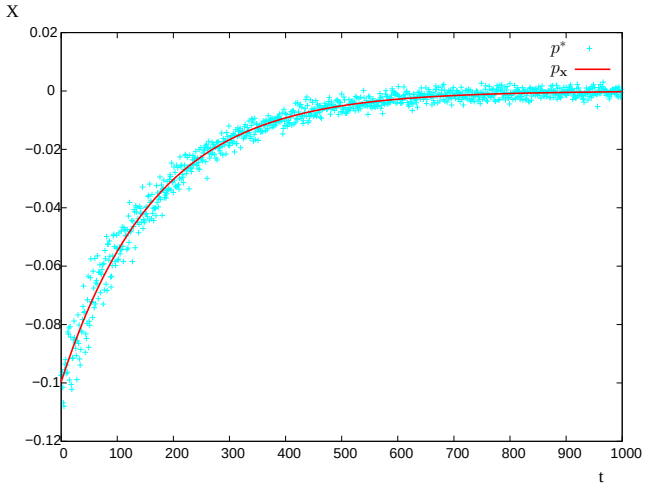

Fig. 6. Example of the left hand noised trajectory fitted with an exponential decay

robot's motion is null, which means that all relevant tasks have been detected.

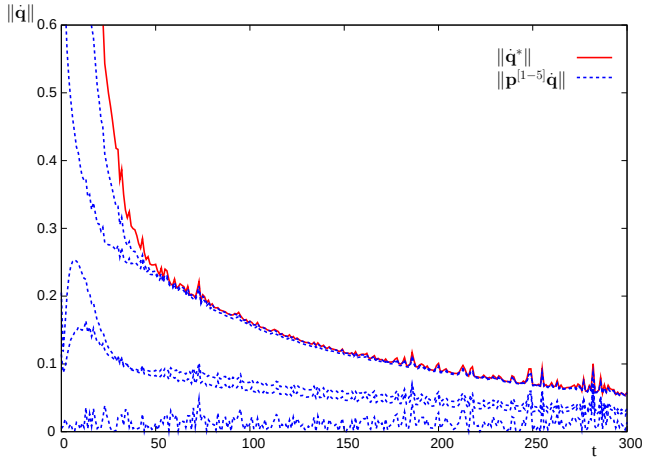

Fig. 7. Successively projecting the motion in the null-spaces of the tasks

V. Future works

A. Model of a human motion and real human trajectory fitting

We want to investigate the possibility of applying the tasks selection method in the case where the motion to imitate is provided by a human.

A motion capture system was used to record a real human motion. The data were recorded at 100Hz with 10 cameras. The subject wore 23 markers for tracking.

The model chosen for the human motion is a minimum jerk trajectory. This kind of trajectory describes a natural human motion as stated in [11]. The jerk is the derivative of the acceleration, so the trajectory is computed by integrating the jerk 3 times. The model chosen for the jerk is not a

polynomial but rather a four stages signal which correspond to a linear acceleration, deceleration, acceleration and null acceleration. That will lead to a 6 parameters model. The parameters are the relative time between each stages, and 3 values of the jerk. Then the jerk is defined as :

$$\text{jerk}(t) = \begin{cases} K_1 & if \quad 0 < t < t_1 \\ K_2 & if \quad t_1 < t < t_1 + t_2 \\ K_3 & if \quad t_2 < t < t_1 + t_2 + t_3 \\ 0 & if \quad t > t_1 + t_2 + t_3 \end{cases} \tag{10}$$

Defining the jerk that way will directly provide us a minimum jerk function. Figure 8 shows an example of a minimum jerk, and its corresponding trajectory.

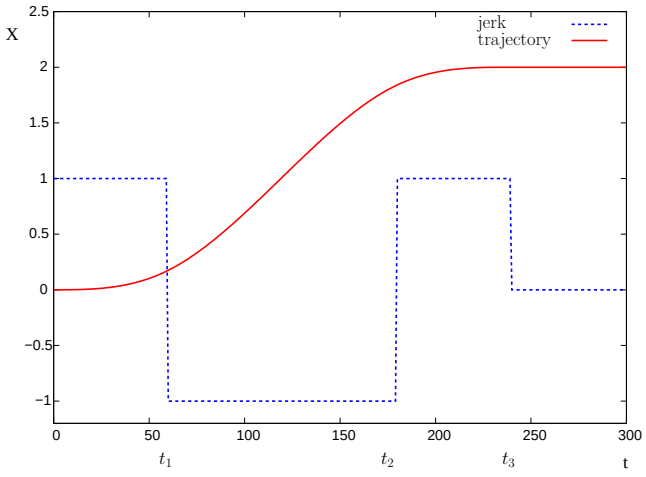

Fig. 8. An example of a minimum jerk and its corresponding trajectory

We assumed that the trajectory was a point to point one. So the trajectory must end with a constant position. The problem is to make this trajectory continuous, so that the optimization performance is not poor. This is done by constraining the final relative time, and the final jerk value : the velocity and acceleration at $t = t_1 + t_2 + t_3$ is null. It leads to a system of 2 equations which will be solved over K_3 and t_3.

The optimization problem becomes :

$$\begin{cases} \mathbf{x}^* = \arg\min \|\mathbf{p}^*(t) - \mathbf{p}_\mathbf{x}(t)\|^2 + x_3^2 + x_4^2 \\ \mathbf{p}_\mathbf{x}(t) = \iiint_\mathbf{x} \text{jerk}(t)\text{dt} \end{cases} \tag{11}$$

where $\mathbf{x} = [dt_1 \ dt_2 \ K_1 \ K_2]$, $\mathbf{p}^*$ is a human trajectory. The second part of the cost function $x_3^2 + x_4^2$ is introduced to limit the K_1 and K_2 parameters, so that the trajectory obtained by integrating the jerk is reasonable.

In order to test this model, we use the motion capture system to record a subject performing a motion. The subject was asked to touch with his finger the top of a bottle and to keep at least one foot on the ground. The bottle that was far enough from him so that the subject has to lift a foot in order to reach the bottle. The point was to focus on the trajectory of the touching hand. The final position is clearly defined, and the orientation is not constrained.

The trajectory of the touching hand is given to the optimization program in order to fit the real trajectory with our model. As it can be seen on the figure 9, the optimization performs well with our model of the trajectory.

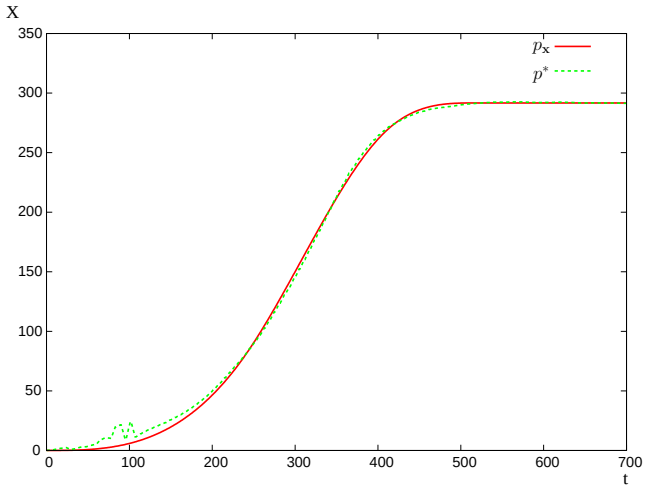

Fig. 9. Minimum Jerk Fitting

Of course, this experiment does not prove that the minimum jerk model can discriminate intended motion and unintended ones. This has to be investigated.

B. Sequential data processing

We also want to use the tasks selection method presented here in a sequential data processing context. What we can call a *full* motion can be segmented in a sequence of stack of tasks as it can be seen in figure 10

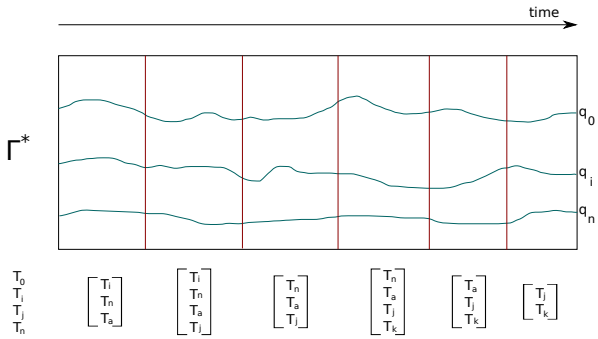

Fig. 10. Motion segmented in a sequence of stack of tasks

For example, in a boxing activity, one can first do two jabs followed by one uppercut. That is where we want to apply classical sequential data processing techniques, in order to learn probabilities of transition between instances of stack of tasks.

VI. Conclusion

Our contribution takes place in both stage of the classical imitation method:

- the evaluation of what to imitate. Our method manages to distinguish two motions which differences lie on the controlled tasks. In that situation, a human will not be able to tell whether or not a motion is a controlled or a side effect one.
- how to imitate. The use of task function formalism, and the optimization technique provide us all the informations needed to replicate a motion (that is to say the parameters of the tasks).

This work leads us to two different interesting problems which can be seen as the possibility of processing human motion data with a stack of tasks approach. The other problem is the problem of sequential motion imitation of a robot where the core of the problem is the selection of the relevant tasks to activate.

REFERENCES

[1] S. Schaal, A.J. Ijspeert, and A. Billard. Computational approaches to motor learning by imitation. *Philosophical Transaction of the Royal Society of London, series B*, 2003.

[2] E. Drumwright and M. Matarić. Generating and recognizing free-space movements in humanoid robots. In *Intelligent Robots and Systems*, Las Vegas, USA, October 2003.

[3] E. Drumwright, O. Chadwicke Jenkins, and M. Matarić. Exemplar-based primitives for humanoid movement classification and control. In *International Conference on Robotics and Automation*, New Orleans, LA, USA, April 2004.

[4] S. Calinon, F. Guenter, and A. Billard. On learning, representing and generalizing a task in a humanoid robot. *IEEE Transactions on Systems, Man and Cybernetics, Part B. Special issue on robot learning by observation, demonstration and imitation*, 2007.

[5] M. Hersch, F. Guenter, S. Calinon, and A. Billard. Dynamical System Modulation for Robot Learning via Kinesthetic Demonstrations. *Transactions on Robotics*, 2008.

[6] M. Mühlig, M. Gienger, J. Steil, and C. Goerick. Automatic selection of task spaces for imitation learning. In *Intelligent Robots and Systems*, St Louis, USA, October 2009.

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

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

[9] N. Mansard and F. Chaumette. Task sequencing for sensor-based control. *Transactions on Robotics*, February 2007.

[10] P. Baerlocher and R. Boulic. Task-priority formulations for the kinematic control of highly redundant articulated structures. In *Intelligent Robots and Systems*, Victoria, BC, Canada, October 1998.

[11] T. Flash and N. Hogan. The coordination of arm movements: An experimentally confirmed mathematical model. *Journal of Neuroscience*, 1984.