# Learning Tracking Control with Forward Models

Botond Bócsi  Philipp Hennig  Lehel Csató  Jan Peters

*Abstract*— **Performing task-space tracking control on redundant robot manipulators is a difficult problem. When the physical model of the robot is too complex or not available, standard methods fail and machine learning algorithms can have advantages. We propose an adaptive learning algorithm for tracking control of underactuated or non-rigid robots where the physical model of the robot is unavailable. The control method is based on the fact that forward models are relatively straightforward to learn and local inversions can be obtained via local optimization. We use sparse online Gaussian process inference to obtain a flexible probabilistic forward model and second order optimization to find the inverse mapping. Physical experiments indicate that this approach can outperform state-of-the-art tracking control algorithms in this context.**

## I. Introduction

Efficient tracking control algorithms are essential in robot control [7], [14]. Tracking control tasks can be formulated on joint-space level when a desired joint-space trajectory has to be followed or can be defined in the task-space. In both cases, the control algorithm has to provide the necessary torques to each joint that will result in the desired motion. Independent of the task formulation, most of the methods are based on the inverse model of the robot (inverse dynamics for joint-space tracking and, additionally, inverse kinematics for task-space tracking). Inverse dynamics – the mapping from the desired joint accelerations to the applied torques – can be directly modeled since there is a one-to-one relation between these quantities resulting in very efficient joint-space tracking algorithms [2], [9], [19]. On the other hand, inverse kinematics – mappings from task-space to joint-space – cannot be modeled straightforwardly. For redundant robots, such a mapping from task-space to configuration space is always non-unique: given a task-space position, there will frequently be several possible joint-space configurations forming a non-convex solution space [4] and one has to choose from these solutions in a *clever* way.

Several solutions have been proposed based on physical models of the robot [16]. However, these methods break down when applied to non-rigid robots systems with insufficient sensing, as well as for systems with nonlinear perceptual transformation. The reason is that the physical models of the robot become inaccurate. Robot with flexible or not directly actuated elements are hard to model accurately with physics and even harder to control. In our experiments, a ball has been attached to the end-effector of a Barrett WAM [9] robot arm using a string, and the ball had to follow a desired trajectory instead of the end-effector of the robot. For this system even the forward kinematics model is undefined since the same joint configuration can result in different ball positions (due to the swinging of the ball). This underactuated robot is not straightforward to control since the swinging produces non-linear effects that lead to undefinable forward or inverse models. We were able to control such a robot and show that our method is capable of capturing non-linear effects, like the centrifugal force, into the control algorithm.

The method presented in this paper is based on the insight that approximate forward models can often be obtained with machine learning methods – even for such ambiguous systems – and that local inversions can be obtained via local optimization. The presented tracking control algorithm has three steps: (1) we model the forward kinematics of the robot using machine learning techniques; (2) apply local optimization to obtain the inverse mapping; (3) based on the inverse mapping, we use a joint-space controller on the controllable degrees of freedom (DoF) of the robot to obtain the desired torques. The complexity of the modeling task (induced by the swinging of the ball in our example) is not shared equally among the three steps. The control method is constructed such that all the non-linear effect are captured by the forward kinematics model and the inversion algorithm.

Hence, for robot systems with uncertain physical parameters, data-driven kinematics model learning presents an appealing alternative to analytical models. Forward kinematics is a standard regression problem, and is thus straightforward to learn from data. We use Gaussian process (GP) regression [3], [13] to approximate this model and get the inverse mapping by searching for the joint configuration that minimizes the distance in the task-space between the predicted task-position and the desired one. GPs provide a flexible tool to model complex, non-linear mappings efficiently. Thus, they are suitable for non-rigid kinematic models. Furthermore, GPs allow a natural way of incorporating prior knowledge about the robot – if available – into the prediction model [8]. To overcome the computational problems of GPs, sparse GP approximation algorithms have been used, allowing real-time modeling.

Our tracking control method uses an adaptive inverse kinematics learning approach that solves the problem of non-uniqueness of inverse kinematics and can also be employed on robots with complex or unknown physical model. The

Botond Bócsi and Lehel Csató are with Faculty of Mathematics and Informatics, Babeş-Bolyai University, Kogalniceanu 1, 400084 Cluj-Napoca, Romania, {bboti, lehel.csato}@cs.ubbcluj.ro

Philipp Hennig and Jan Peters are with Department of Empirical Inference, Max Planck Institute for Intelligent Systems, Spemannstraße 38, 72076 Tübingen, Germany, {phennig@tuebingen.mpg.de, jan.peters}@tuebingen.mpg.de

Jan Peters is with Technische Universitaet Darmstadt, Intelligent Autonomous Systems Group, Hochschulstr. 10, 64289 Darmstadt, Germany

technique can be applied online, i.e., the robot model can be trained while performing the control task.

We start the paper with a definition of the inverse kinematics problem since it plays a central role in our tracking control algorithm and give an overview over existing solutions (Section I-A). Section II-A shows how local inversions can be obtained from forward kinematic maps. Section II-B presents the forward learning algorithm and Section II-C gives a brief introduction to available sparse GP approximations. Section III sums these results up into a tracking control algorithm with practical considerations.

### A. Analytical & Numerical Solutions for Inverse Kinematics

The forward kinematics are given by the correspondence

$$x = g(\theta),$$

and maps $n$-dimensional joint angles $\theta \in \Re^n$ into $p$-dimensional task-space positions $x \in \Re^p$. The inverse kinematics transforms the end-effector coordinates into joint angles:

$$\theta = g^{-1}(x). \tag{1}$$

Finding $g^{-1}(\cdot)$ is a significantly harder problem than modeling $g(\cdot)$. For redundant systems, i.e., when the dimension of the task-space is smaller than the dimension of the joint-space ($p < n$), $g^{-1}(\cdot)$ is not well-defined. For a task-space position $x$, there can be several corresponding joint-space configurations $\theta$.

Classical approaches solve inverse kinematics on the velocity level, i.e., the differential inverse kinematics [16], where the derivative of the forward kinematics model is employed: $\dot{x} = \mathbf{J}(\theta)\dot{\theta}$, where $\mathbf{J}(\theta)$ is the Jacobian of $g(\theta)$. Differential inverse kinematics determines the desired joint velocity $\dot{\theta}$, and uses this online. This joint velocity (for example) can be obtained by the Jacobian transpose method $\dot{\theta} = \mathbf{J}(\theta)^{\top}\dot{x}$, or by the resolved velocity method $\dot{\theta} = \mathbf{J}(\theta)^{\dagger}\dot{x}$, where $\mathbf{J}(\theta)^{\dagger}$ is the pseudo-inverse [16]). It is important that resolved velocity methods are numerically unstable if $\mathbf{J}(\theta)$ is a singular matrix and $\mathbf{J}(\theta)^{\dagger}$ does not exist.

Numerical solutions to $\theta = g^{-1}(x)$ can be found by iterating any of the above inverse kinematics methods until convergence (i.e., $\theta' = \theta + \mathbf{J}(\theta)^{\dagger}\dot{x}$ as $x = g(\theta)$ for the fixpoints of this iteration).

To solve redundancy of solutions, gradient projection methods put additional constrains on $\theta$ by optimizing a cost function $h(\theta)$ in the null-space of the mapping $\mathbf{J}(\theta)$ , i.e., $\dot{\theta} = \mathbf{J}(\theta)^{\dagger}\dot{x} + \left[\mathbf{I} - \mathbf{J}(\theta)^{\dagger}\mathbf{J}(\theta)\right] \partial h/\partial\theta$ [16].

Instead of traditional numerical methods used with numerical models, we can sometimes learn inverse kinematics using sampled data. This approach can be advantageous for several reasons [1], [4]: (1) traditional numerical methods require a precise kinematics model of the robot that might not be available, e.g., for complex robots, non-rigid manipulators, flexible joint robots, or when uncalibrated cameras provide noisy Cartesian coordinates; (2) iterative solutions are often too slow for real-time applications, (3) if a system can change over time, we need to adapt the inverse kinematics model as well. In the next section, we give a brief overview of how machine learning methods have been used to learn inverse kinematics.

### B. Learning Inverse Kinematics

Among the most well-known online learning approaches for inverse kinematics is D'Souza et. al. [4], based on "locally weighted projection regression" (LWPR) [21]. LWPR learns the inverse kinematics mapping on the velocity level. Its central idea is to partition the input space into regions and learn local piecewise linear models for each region. The input space, i.e., desired end-effector velocity, is augmented with the current joint configuration of the robot and, subsequently, the mapping $(\dot{x}, \theta) \to \dot{\theta}$ is learned. By adding the current joint configuration to the input space, the localization of the linear models becomes valid. This augmentation leads to a locally consistent algorithm, however, global consistency is not guaranteed. Global consistency can be achieved by selectively generating data. D'Souza et. al. [4] argue that the main disadvantage of LWPR is that the partitioning of the augmented input space becomes difficult for robots with many DoFs.

A similar modular construction of $g^{-1}(\cdot)$ is employed by Oyama et al. [11] using neural networks as local models and a gating neural network that chooses among them. This approach, also, requires an oracle determining model responsibilities, which is difficult to obtain in high dimensions.

Our method is based on the observation that modeling the forward kinematics function is significantly easier than the inverse mapping. This insight has been investigated in the literature using different approaches to obtain the inverse mapping [14]. Radial basis function networks have been used for forward modeling by Sun and Brian [20] on a 4-DoF robot. The method was trained on data collected offline and has not been tested on trajectory following problems, but on reaching tasks. The inverse kinematics mapping has been achieved on velocity level by differentiating the neural network. The authors argue that the structure of the network and the parameter settings have a major effect on the generalization capacity of the algorithm: a long and task dependent parameter tuning is required.

Neural networks have been used by Jordan and Rumelhart [5] in a different context: they used them to learn the forward kinematics model of a robot and trained another neural network for the inverse kinematics, as the composition of the two networks to yield the identity. However, training the inverse model in this indirect way is difficult due to local minimum, instability, and problems in selecting the network structure.

## II. Tracking and Learning with Forward Models

Next, we show how efficient forward kinematics models can be constructed from data using machine learning methods. We also present how local inverses can be obtained from the learned forward models.

## A. *Inverse Kinematics with Forward Kinematics Modeling*

Once the forward kinematics model is known, the inverse mapping is obtained by choosing the joint configuration that minimizes the distance between the desired task-space position and the one predicted by the forward model. By definition, the inverse function looks as follows:

$$\begin{aligned} g^{-1}(x) &\overset{\circ}{=} \arg\min_{\theta} \|x - g(\theta)\|^2 \\ &= \arg\min_{\theta} F(x, \theta), \end{aligned} \quad (2)$$

where $F(x, \theta)$ denotes the Euclidean distance between the desired task-space position and the one predicted by the forward model. Note that $F(x, \theta)$ can be considered an energy function that we want to minimize. Thus, it may contain additional $x$ or $\theta$ dependent terms as well. For instance, it is useful to add a regularization term $\lambda h(\theta)$ that keeps the robot close to a natural joint configuration, i.e., $h(\theta)$ has smaller values for desired configurations and bigger values for undesired configurations. We omit such terms here for brevity, their inclusion is straightforward (see Section II-D for the importance of the regularization).

Equation (2) poses two questions: (1) how should we perform the minimization in a possibly high dimensional continuous joint-space and (2) how should we model $g(\cdot)$? In practice, these two problems are not independent. If we choose a continuous and differentiable function $g(\cdot)$, gradient descent search methods are applicable. The gradient of the energy function has the form

$$\nabla_\theta F(x, \theta) \;=\; 2(x - g(\theta))^\top \nabla g(\theta), \quad (3)$$

that can be derived from Equation (2).

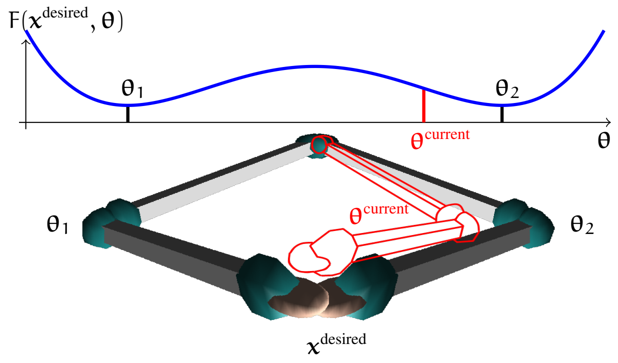
A second question relates to the non-uniqueness of the inverse kinematics function: how does the minimization ensure that a convenient solution will be chosen when a desired end-effector $x^{\text{desired}}$ position can be reached by two different joint configurations $\theta_1$ and $\theta_2$, see Figure 1? In this case, the algorithms should predict $\theta_2$ to avoid jerky movements. This behavior is desirable to achieve smooth trajectories in joint-space. We propose to start a gradient search from the current joint positions $\theta^{\text{current}}$.

Next, we present an adaptive way of modeling $g(\cdot)$ using non-parametric machine learning methods, i.e., Gaussian processes.

## B. *Forward Model Learning and Inversion with Gaussian Processes*

We obtain a non-parametric model of $g(\cdot)$ using Gaussian processes (GPs) [13]. Despite that they are non-linear functions of the input, the prediction is linear in the number of the training data points. Furthermore, the gradient of the predictive function has an analytical form.

From a machine learning point of view, GPs are non-parametric methods for regression and classification [13]. Given a training set $\mathbf{D} = \{(\theta_i, x_i)\}_{i=1}^m$ with inputs $\theta_i$ and labels $x_i$, the prediction for a new $\theta$ is a Gaussian–distributed



$F(x^{\text{desired}}, \theta)$
$\theta_1$
$\theta_2$
$\theta^{\text{current}}$
$\theta$
$\theta_1$
$\theta^{\text{current}}$
$\theta_2$
$x^{\text{desired}}$

Fig. 1. Illustration of the inverse kinematics prediction scheme [1]. In the training set $x^{\text{desired}}$ has been reached by two different joint configurations $\theta_1$ and $\theta_2$, therefore, $F(x^{\text{desired}}, \theta_1) = F(x^{\text{desired}}, \theta_2)$. As prediction our algorithm will chose $\theta_2$ since the current joint configuration $\theta^{\text{current}}$ is in the attraction range of $\theta_2$.

random variable with mean $\mu(\theta)$ and variance $\sigma^2(\theta)$ where

$$\begin{aligned} \mu(\theta) &= \mathbf{k}_\theta^\top \boldsymbol{\alpha} = \sum_{i=1}^{m} \alpha^i k(\theta, \theta_i) \\ \sigma^2(\theta) &= k(\theta, \theta) - \mathbf{k}_\theta^\top \mathbf{K}^{-1} \mathbf{k}_\theta, \end{aligned} \quad (4)$$

where $\mathbf{k}_\theta \in \Re^{m \times 1}$ is a vector with elements $\mathbf{k}_\theta^i = k(\theta, \theta_i)$ and $\mathbf{K} \in \Re^{m \times m}$ is a matrix with elements $\mathbf{K}^{ij} = k(\theta_i, \theta_j)$. The function $k : \Re^n \times \Re^n \to \Re$ is a positive definite kernel (a generalization of a positive definite matrix) [13], [15] and $\boldsymbol{\alpha} \in \Re^{m \times 1}$, where $\boldsymbol{\alpha} = \mathbf{K}^{-1} x$ are the parameters of the GP.

Observe that $\boldsymbol{\alpha}$ does not depend on the test data, thus, once $\mathbf{K}^{-1}$ is evaluated, the evaluation of $\mu(\theta)$ is linear in the size of the training set. Inverting $\mathbf{K}$ is cubic in the size of the training set, however, this inversion can be done during the training. This step will be approached with consideration.

The predictive mean of the GPs is used as the prediction of the forward model from Equation (4)[1], $g(\theta) = \mu(\theta)$, the gradient of the energy function we want to minimize from Equation (3) is given by

$$\nabla_\theta F(x, \theta) \;=\; 2(x - \mathbf{k}_\theta^\top \boldsymbol{\alpha})^\top \boldsymbol{\alpha}^\top \frac{\partial \mathbf{k}_\theta}{\partial \theta},$$

where $\partial \mathbf{k}_\theta / \partial \theta \in \Re^{m \times n}$ is a matrix with the partial derivatives of $\mathbf{k}_\theta$ with respect to every element of $\theta$.

A wide range of positive definite kernels is available [15]. In our experiments, we found that the popular squared exponential kernels provides good performance for free motion kinematic tasks. It has the form $k(\theta_1, \theta_2) = C \exp\left\{\|\theta_1 - \theta_2\|^2 / 2w\right\}$ and its gradient is

$$\frac{\partial k(\theta_1, \theta_2)}{\partial \theta_1} \;=\; \frac{1}{w} k(\theta_1, \theta_2)(\theta_1 - \theta_2)^\top,$$

where $C$ and $w$ are hyper-parameters of the kernel function[2].

[1]We used a different GP for each output dimension, i.e., $g_j(\theta) = m_j(\theta)$, $j = \overline{1,3}$ where $g_j(\theta)$ is the j-th component of $g(\theta)$.

[2]Note that $w$ can be vector valued. Thus, we can define different lengths scale in each input dimension, i.e., $k(\theta_1, \theta_2) = C \exp\left\{\sum_i (\theta_1 - \theta_2)_i^2 / 2w_i\right\}$, called an anisotropic squared exponential kernel. This useful property in forward kinematics modeling allows for different weights for each DoF.

As we have already discussed, once $\mathbf{K}^{-1}$ is known, the prediction of the GP is linear in the number of training examples. We consider $\mathbf{K}^{-1}$ known since the presented online GP method provides an efficient update of $\mathbf{K}^{-1}$ that does not require the cubic complexity inversion of $\mathbf{K}$. The complexity of $F(x, \theta)$ and $\nabla_\theta F(x, \theta)$ is $\mathcal{O}(m)$. If the gradient search from Equation (2) requires $l$ function or gradient evaluations, then $g^{-1}(x)$ has complexity $\mathcal{O}(lm)$. For smooth and fast predictions we need the inverse kinematics function to be evaluated fast. One option is to keep $l$ low by applying an efficient gradient search method and, another is to keep $m$ low by reducing the sample points of the GP. We consider the former by using a conjugate gradient optimizer [18]. The second problem, in our particular online setting, requires an *online sparsification* algorithm for Gaussian processes. Several algorithms exist in the literature [3], [6], [12], [17]. The following section gives a brief overview of the one we used in our experiments.

### *C. Dealing with Large Amounts of Data*

The main disadvantage is that the time and space complexity of GP inference grows cubically with the number of the data points. To overcome this problem, several sparsification methods have been proposed [3], [6], [12], [17]. We adopt a method proposed by Csató and Opper [3], which can be applied online. The method is the online approximation to the posterior distribution using a sequential algorithm [10] where we combine the likelihood of a single data with the GP prior from the result of the previous approximation step. Given a dataset $\mathbf{D} = \{(\theta_i, x_i)\}_{i=1}^m$, a $\mathcal{GP}_m$ model is defined based on $m$ data points. When a new point arrives, $\mathcal{GP}_{m+1}$ is obtained by using the Bayes theorem where $\mathcal{GP}_m$ plays the role of the prior model and $(\theta_{m+1}, x_{m+1})$ is the observation, i.e.,

$$\mathcal{GP}_{m+1} \quad \propto \quad p((\theta_{m+1}, x_{m+1})|\mathcal{GP}_m)\, \mathcal{GP}_m.$$

Assuming additive Gaussian noise, this assumption leads to an analytically tractable model [13].

The sparsification algorithm defines an approximation to the exact Gaussian process posterior, chosen such that the discrepancy between the exact and the approximate posterior is minimized. An important feature of the method is that the parameters of the GP are updated even when the new data point is not included into the base set. Thus, the accuracy of the approximation improves during the learning process while the base points might not change.

### *D. Practical Considerations and Implementation*

We adopt the term "forward Gaussian process modeling" (FWGP) for the method as presented here. Algorithm 1 contains an overview of the task-space tracking control method in pseudo-code. First, a GP is initialized with no training points as the forward kinematics model. Subsequently, this forward model is updated, while performing the desired task based on the respective task and joint positions. At each step, the desired joint configuration is obtained by minimizing the distance between the desired and predicted task-space

**Algorithm 1** Task-Space Tracking Control with FWGP

```
$\mathcal{FW} \leftarrow$ init-GP() {forward kinematics model}

while task is not over do
    $\mathcal{FW} \leftarrow$ update-GP($x^{\text{current}}$, $\theta^{\text{current}}$)
    $x^{\text{desired}} \leftarrow$ next-position ()
    $\theta^{\text{desired}} \leftarrow$ gradient-minimization ($\theta^{\text{current}}$,
                        $\|\mathcal{FW}(\cdot) - x^{\text{desired}}\|^2$)
    inverse-dynamics ($\theta^{\text{current}}$, $\theta^{\text{desired}}$)
end while
```

position. Finally, the inverse dynamics model is used for computing the torques for the actuated DoFs of the robot.

The assumption that there is no prior information on the kinematics model of the robot is frequently too strong. When available, such prior knowledge can be incorporated into the prediction model in different ways: (1) the prior mean of the GP can be a parametric function of the known kinematics [8] or (2) a suitable kernel function can be constructed [8].

An advantage of online adaptive sparsification is that it automatically adapts the number of representer data points to the complexity of the task: simple trajectories on low-dimensional manifolds can be represented with fewer data points than complex trajectories in high-dimensional spaces. This complexity adaptation is an advantage over models based on neural networks, where complexity and cost of the model have to be fixed beforehand.

A major concern in task-space control is keeping joint-space stability [4], [16]. As we discussed in Section II-B, augmenting the energy function from Equation (2) with a regularization term may help keeping this stability. We propose a simple regularization term, i.e.,

$$h(\theta) \quad = \quad \frac{1}{2}(\theta - \theta_{\text{rest}})^\top(\theta - \theta_{\text{rest}}), \qquad (5)$$

where $\theta_{\text{rest}}$ is a joint configuration far from the joint limits of the robot. By taking into account this extra term, we obtain joint-space stability by preventing the addition of undesired joint configurations into the training set. We emphasize that even though the regularization was not significant in our experiments (the $\lambda$ was $10^{-5}$), we did not encounter joint-space instability. This observation suggests that direct inverse kinematics modeling is less sensitive to joint-space instability than differential inverse kinematics where the joint-space regularization is essential.

## III. Experiments

In this section, we present two experiments that suggest the feasibility of the proposed method for real-time learning. First, we show that the inverse kinematics model used by FWGP outperforms LWPR for standard task-space tracking tasks and achieves accuracy that is close to the analytical solution. Subsequently, we show that FWGP can adapt to the nonlinear dynamics of non-rigid robots.

### A. *Online Task-Space Tracking for the End-effector*

Real world task-space tracking experiments has been conducted to compare the performance (tracking accuracy, learning speed) of FWGP, LWPR, and the analytical model with pseudo-inverse method. To facilitate comparison to a ground truth, we performed the physical experiments with a 7-DoF Barrett WAM arm [9], whose physical parameters are known. For stability, we used $\theta_{\text{rest}} = [0\ 0.5\ 0\ 1.9\ 0\ 0\ 0]^\top$ as rest posture from Equation (5).

The task was to perform task-space tracking of a figure eight, see Figure 2(b), while learning the kinematic model of the robot at the same time. For FWGP and LWPR, we started from an untrained system. We used a 500Hz sample rate meaning that after each minute of learning 30000 new training points had to be processed. For LWPR the initial learning rate was set to $\alpha = 250$ and the initial forgetting rate to $\lambda = 0.99$ [21].

Figure 2(a) shows that after 20 seconds of online learning an acceptable tracking accuracy was achieved by FWGP. After four more minutes of learning and tracking, high accuracy was achieved, see Figure 2(b). We present a comparison of tracking accuracy on this task in Figure 2(c). FWGP outperforms LWPR after 20 seconds of interaction time, and approaches the performance of the analytical model.

All experiments were carried out in *real time*, demonstrating that the complexity of the forward model and the proposed inversion is manageable. After four minutes of learning, as result of the GP sparsification, the final GP model was built from 15–20 base points instead of 120000. This reduction means that only a matrix of size $20 \times 20$ had to inverted in Equation (2). The same number of base points were also used for other tasks, such as the star-like figure [7]. When the task was changed, during runtime, from a figure eight to a star-like path , the online model adapted by adding between one and four new points to the base set.

Based on 15–20 data points, the conjugate gradient minimization from Equation (2) needed 10–20 function or gradient evaluations. The number of evaluations is not fixed as it depends on the distance between the current joint position and the predicted one. On average, the computation of one desired joint coordinate took 2–10 milliseconds on an Intel Core Duo 1.66Ghz processor.

### B. *Online Task-Space Tracking for a Swinging Ball*

In this experiment, we made the simulated model more complex by attaching a ball to the end-effector of the arm with a 20 cm string. The mass of the ball was negligible compared to the mass of the robot arm and air friction was neglected. The swinging motion of the ball produced a non-linear system. In this new setting, we performed task-space tracking where the position of the ball was considered as the desired position. The task was to track a circle figure with 20 cm radius, see Figure 3(a), on the horizontal plane. The task was performed in two different settings: (1) when the desired point moved slowly along the circle (one turn took 24 seconds) and (2) when the desired point moved fast along the circle (one turn took 0.62 seconds).

In the first case, FWGP learned to move the end-effector of the arm right above the desired circle while the ball was moving along the desired trajectory since the swinging of the ball was damped. It took four minutes of learning to achieve the learning accuracy presented on Figure 3(a) and Figure 3(b). The GP model was built from 20-25 data points.

In the second case, the trajectory of the end-effector of the arm was moving fast inside the desired circle and used the centrifugal force to swing the ball around, see Figure 3(c). After 20 minutes of learning, the GP model was built from 13-15 data points. Following the fast circle movement with the arm itself would be impossible, since it would reach the physical limits of the robot.

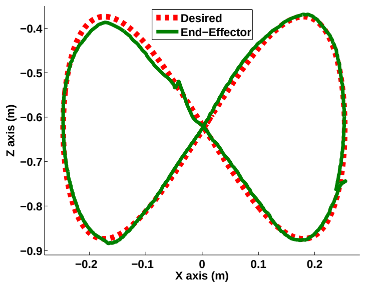
We emphasize that the same parameter settings were used for both experiments (with GP hyper-parameters $C = 1$ and $w = 0.7$) Thus, the adaptive behavior depends weakly on the hyper-parameters of the GP. In the first case, FWGP considered the swinging motion of the ball as noise and the trajectory of the ball followed a similar trajectory as the end-effector. On the other hand, in the second experiment, the GP model incorporated into the control model the centrifugal force as well. No comparison with LWPR has been made as it could not learn the previous tasks sufficiently well. One possible explanation for this failure is that differential inverse kinematics based control is more sensible to noise measurements, e.g., the velocity of the ball followed a less smooth trajectory than its position.

## IV. Discussion

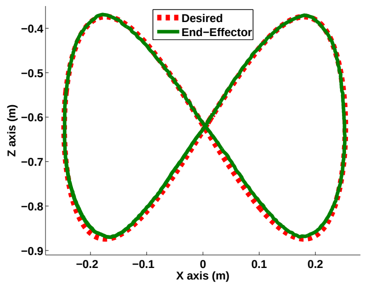
We presented an adaptive task-space tracking control algorithm for robot manipulators that is capable of controlling non-linear robot models with noisy observations, such as a ball attached at the end of a robot arm. FWGP exploits the fact that forward robot models are relatively straightforward to obtain and the inverse mapping is calculated using local optimization. Sparse online GPs were used to represent the forward kinematics model and the inverse is calculated using efficient gradient search in the joint-space. We showed that our approach outperforms the standard task-space control methods and converges to the analytical solution provided by the physical model of the robot. Furthermore, for complex robots the standard methods fail while FWGP achieves good performance.

## Acknowledgments

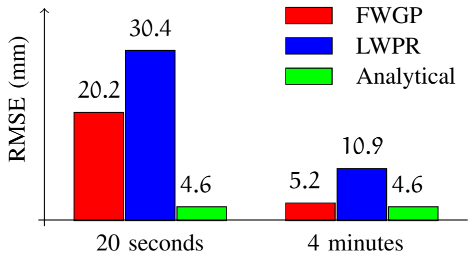
B. Bócsi wishes to thank for the financial support provided from program: Investing in people! PhD scholarship, project co-financed by the European Social Fund, sectoral operational program, human resources development 2007 - 2013. Contract POSDRU 88/1.5/S/60185. B. Bócsi and L. Csató acknowledge the support of the Romanian Ministry of Education, grant PN-II-RU-TE-2011-3-0278. The project receives funding from the European Community's Seventh Framework Programme under grant agreement no. ICT-248273 GeRT. The project receives funding from the European Community's Seventh Framework Programme under grant agreement no. ICT-270327 CompLACS.



(a) Tasks-space tracking of a figure eight after 20 seconds of online learning.

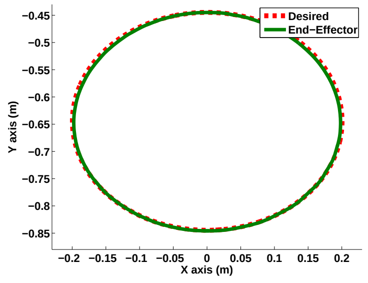


(b) Tasks-space tracking of a figure eight after four minutes of online learning.

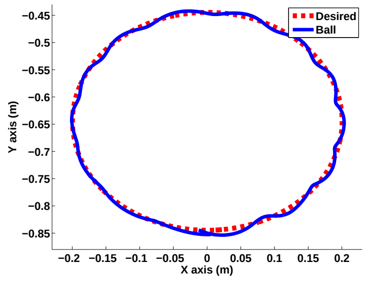


(c) Tracking accuracy comparison.

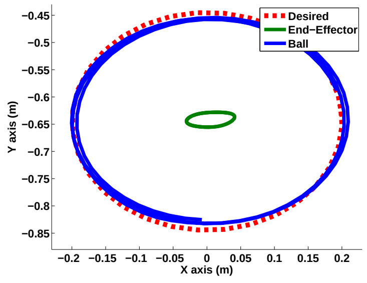
Fig. 2. Online task-space tracking learning of the figure eight task by a 7-DoF Barrett WAM robot arm. (a) The tracking performance of FWGP after 20 seconds of online learning is acceptable. (b) After four minutes of learning, very good tracking accuracy is obtained. (c) FWGP outperforms LWPR both after 20 seconds and four minutes of learning and its accuracy converges to the analytical solutions.



(a) End-effector trajectory with slow circle learning.



(b) Ball trajectory with slow circle learning.



(c) End-effector and ball trajectory with fast circle learning.

Fig. 3. Online task-space tracking learning of a circle following task by a simulated 7-DoF Barrett WAM robot arm with a ball attached on it. (a) When the movement of the desired point on the circle is slow the end-effector is placed above the desired trajectory while (b) the ball is hanging down and its swinging is damped – the swinging is considered noise by FWGP. (c) When the desired point moves fast on the circle the end-effector moves inside the desired circle while the ball swings around along the desired trajectory – the centrifugal force is incorporated into the control model.

## REFERENCES

[1] Botond Bócsi, Duy Nguyen-Tuong, Lehel Csató, Bernhard Schoelkopf, and Jan Peters. Learning inverse kinematics with structured prediction. In *Proceedings of the IEEE International Conference on Intelligent Robots and Systems*, pages 698–703, San Francisco, USA, 2011.

[2] John J. Craig. *Introduction to Robotics: Mechanics and Control*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 2nd edition, 1989.

[3] Lehel Csató and Manfred Opper. Sparse on-line Gaussian processes. *Neural Computation*, 14(3):641–668, 2002.

[4] A. D'Souza, S. Vijayakumar, and S. Schaal. Learning inverse kinematics. In *Proceedings of the IEEE International Conference on Intelligent Robots and Systems*. Piscataway, NJ: IEEE, 2001.

[5] Michael I. Jordan and David E. Rumelhart. Forward models: Supervised learning with a distal teacher. *Cognitive Science*, 16:307–354, 1992.

[6] Neil D. Lawrence, Matthias Seeger, and Ralf Herbrich. Fast sparse Gaussian process methods: The informative vector machine. In *Neural Information Processing Systems*, pages 609–616. MIT Press, 2002.

[7] Jun Nakanishi, Rick Cory, Michael Mistry, Jan Peters, and Stefan Schaal. Operational space control: a theoretical and emprical comparison. (6):737–757, 2008.

[8] Duy Nguyen-Tuong and Jan Peters. Using model knowledge for learning inverse dynamics. In *Proceedings of the IEEE International Conference on Robotics and Automation*, pages 2677–2682, 2010.

[9] Duy Nguyen-Tuong, Matthias W. Seeger, and Jan Peters. Model learning with local Gaussian process regression. *Advanced Robotics*, 23(15):2015–2034, 2009.

[10] Manfred Opper. *A Bayesian approach to on-line learning*, pages 363–378. Cambridge University Press, 1998.

[11] Eimei Oyama, Nak Young Chong, Arvin Agah, Taro Maeda, and Susumu Tachi. Inverse kinematics learning by modular architecture neural networks with performance prediction networks. In *Proceedings of the IEEE International Conference on Robotics and Automation*, pages 1006–1012, 2001.

[12] Joaquin Quiñonero Candela and Carl E. Rasmussen. A unifying view of sparse approximate Gaussian process regression. *Journal of Machine Learning Research*, 6:1939–1959, 2005.

[13] Carl E. Rasmussen and Christopher K. I. Williams. *Gaussian Processes for Machine Learning*. The MIT Press, 2005.

[14] Camille Salaün, Vincent Padois, and Olivier Sigaud. Learning forward models for the operational space control of redundant robots. In Olivier Sigaud and Jan Peters, editors, *From Motor Learning to Interaction Learning in Robots*, volume 264 of *Studies in Computational Intelligence*, pages 169–192. Springer, 2010.

[15] Bernhard Schölkopf and Alex Smola. *Learning with Kernels: Support Vector Machines, Regularization, Optimization and Beyond*. MIT-Press, Cambridge, MA, 2002.

[16] Lorenzo Sciavicco and Bruno Siciliano. *Modelling and Control of Robot Manipulators (Advanced Textbooks in Control and Signal Processing)*. Advanced textbooks in control and signal processing. Springer, 2nd edition, January 2005.

[17] Edward Snelson and Zoubin Ghahramani. Sparse Gaussian processes using pseudo-inputs. In *Advances in Neural Information Processing Systems*, pages 1257–1264. MIT press, 2006.

[18] Jan A. Snyman. *Practical Mathematical Optimization: An Introduction to Basic Optimization Theory and Classical and New Gradient-Based Algorithms*. Applied Optimization. Springer-Verlag New York, 2005.

[19] M.W. Spong and M. Vidyasagar. *Robot Dynamics And Control*. Wiley India Pvt. Ltd., 2008.

[20] Ganghua Sun and Brian Scassellati. Reaching through learned forward model. In *Proceedings of the IEEE-RAS/RSJ International Conference on Humanoid Robots*, Santa Monica, 2004.

[21] Sethu Vijayakumar, Aaron D'Souza, and Stefan Schaal. Incremental online learning in high dimensions. *Neural Computation*, 17:2602–2634, 2005.