Learning Inverse Kinematics with Structured Prediction

Botond Bócsi Duy Nguyen-Tuong Lehel Csató Bernhard Schölkopf Jan Peters

***Abstract*— Learning inverse kinematics of robots with redundant degrees of freedom (DoF) is a difficult problem in robot learning. The difficulty lies in the non-uniqueness of the inverse kinematics function. Existing methods tackle non-uniqueness by segmenting the configuration space and building a global solution from local experts. The usage of local experts implies the definition of an *oracle*, which governs the global consistency of the local models; the definition of this *oracle* is difficult. We propose an algorithm suitable to learn the inverse kinematics function in a single global model despite its multivalued nature. Inverse kinematics is approximated from examples using structured output learning methods. Unlike most of the existing methods, which estimate inverse kinematics on velocity level, we address the learning of the *direct* function on position level. This problem is a significantly harder. To support the proposed method, we conducted real world experiments on a tracking control task and tested our algorithms on these models.**

I. Introduction

Kinematic relationships map configuration space to operational space. They are essential components in many robot applications, such as in manipulation and task-space control [1], [2]. Unlike industrial robots, less rigid and less accurate systems, as well as systems with a nonlinear perceptual transformation may not allow an equally accurate modeling. For such robot systems, data-driven model learning presents an appealing alternative to analytical models. However, while forward kinematics models (i.e., mappings from joint-space to task-space) are straightforward to obtain using standard regression techniques [3], learning of inverse kinematics models is more difficult. For redundant robots, learning such a mapping from task-space to configuration space is a highly ill-posed problem. Given a task-space position, there can be many joint-space configurations which may form a non-convex solution space [4]. Naively learning a mapping from task-space to joint-space using standard regression can lead to degenerate models that provide inconsistent predictions in joint-space.

Learning inverse kinematics has been studied in the past [4]–[9]. Most of the proposed methods, attempt to learn the inverse kinematics on the velocity level, i.e., learning differential inverse kinematics [4], [7], [8]. However, to the best of our knowledge, there are only few approaches which deal with the direct inverse kinematics on the position level [10]. The main advantage of the second approach is that it does not require the segmentation of the input space to obtain a global solution. Differential inverse kinematics provide only locally unique solutions [4]. In this paper, we focus on learning a *direct mapping* from the task-space position to the joint-space configuration using structured output prediction [11], [12]. The basic idea behind this approach is that a probabilistic model of the joint input-output space is well-defined and can be learned in the first step. Subsequently, predictions for target outputs, i.e., joint-space configurations, can be obtained by maximizing this probabilistic model for given inputs, i.e., task-space positions. Thus, the potential ambiguity in the output space, i.e., different joint configurations resulting in the same end-effector position, is resolved by finding the most probable output solutions, which explain the current input trajectory.

The remainder of the paper is organized as follows. First, we state the problem of inverse kinematics and give an overview of the related work. Section II contains an introduction to structured output learning highlighting the properties that make them adequate to model multivalued functions. A detailed presentation of joint kernel support estimation is also given in Section II-A. In Section III, we explain how joint kernel support estimation is used in conjunction with learning inverse kinematics models. Results of real world experiments on a Barrett WAM are presented in Section IV. We summarize the contributions of the paper in Section V and indicate possible future directions.

A. Problem Statement

Modeling forward kinematics is a straightforward problem. Here, we need to model the unique relationship $\boldsymbol{x} = g(\boldsymbol{\theta})$ mapping joint angles $\boldsymbol{\theta}$ into task-space positions $\boldsymbol{x}$, where $\boldsymbol{x} \in \Re^p$ and $\boldsymbol{\theta} \in \Re^n$. Modeling inverse kinematics means finding a mapping from the end-effector coordinates into joint angles. Thus,

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

Finding $g^{-1}(\cdot)$ is not straightforward. For redundant robot 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 a unique mapping. Given a task-space position $\boldsymbol{x}$, there can be many corresponding joint-space configurations $\boldsymbol{\theta}$. Thus, learning the direct inverse kinematics function $g^{-1}(\cdot)$ relates to the problem of learning a multivalued function.

Many attempts have been done in solving inverse kinematics on the velocity level, i.e., the differential inverse kinematics [1], [2]. Here, the derivative of the forward kinematics

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

Duy Nguyen-Tuong, Bernhard Schölkopf, and Jan Peters are with Department of Empirical Inference, Max Planck Institute for Intelligent Systems, Spemannstraße 38, 72076 Tübingen, Germany, {duy, bernhard.schoelkopf, jan.peters}@tuebingen.mpg.de

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

model is employed, i.e., $\dot{\boldsymbol{x}} = \boldsymbol{J}(\boldsymbol{\theta})\dot{\boldsymbol{\theta}}$, where $\boldsymbol{J}(\boldsymbol{\theta}) = \partial g/\partial \boldsymbol{\theta}$ is the Jacobian. Differential inverse kinematics approaches determine the desired joint velocity $\dot{\boldsymbol{\theta}}$, and use this online. This joint velocity can be obtained by the Jacobian transpose method, i.e., $\dot{\boldsymbol{\theta}} = \boldsymbol{J}(\boldsymbol{\theta})^\top \dot{\boldsymbol{x}}$, or by the resolved velocity method, i.e., $\dot{\boldsymbol{\theta}} = \boldsymbol{J}(\boldsymbol{\theta})^\dagger \dot{\boldsymbol{x}}$, where $\boldsymbol{J}(\boldsymbol{\theta})^\dagger$ is the pseudo-inverse of the Jacobian [1]. Note that resolved velocity method will not work when $\boldsymbol{J}(\boldsymbol{\theta})$ is a singular matrix, thus, the inversion of the Jacobian is an ill-posed problem and $\boldsymbol{J}(\boldsymbol{\theta})^\dagger$ does not exist.

Numerically solving for $\boldsymbol{\theta} = g^{-1}(\boldsymbol{x})$ can be done by iterating either of the previous differential inverse kinematics methods, e.g., $\boldsymbol{\theta}' = \boldsymbol{\theta} + \boldsymbol{J}(\boldsymbol{\theta})^\dagger \dot{\boldsymbol{x}}$, until convergence, i.e., $\boldsymbol{x} = g(\boldsymbol{\theta})$ and hence $\boldsymbol{\theta} = g^{-1}(\boldsymbol{x})$ is fulfilled.

For redundant robots there exist multiple solutions for $\boldsymbol{\theta}$ and one has to favor certain solutions. To resolve this redundancy, gradient projection methods put additional constrains on $\boldsymbol{\theta}$ by optimizing a cost function $h(\boldsymbol{\theta})$ in the null-space of the mapping $\boldsymbol{J}(\boldsymbol{\theta})$ [2], [8], i.e., $\dot{\boldsymbol{\theta}} = \boldsymbol{J}(\boldsymbol{\theta})^\dagger \dot{\boldsymbol{x}} + \left[\boldsymbol{I} - \boldsymbol{J}(\boldsymbol{\theta})^\dagger \boldsymbol{J}(\boldsymbol{\theta})\right] \partial h/\partial \boldsymbol{\theta}$.

Beyond traditional numerical methods, learning inverse kinematics (approximating $g^{-1}(\cdot)$ from Equation (1) using sampled data) can be an appealing alternative for several reasons: (1) traditional numerical methods require a precise kinematic model of the robot that might not be available, e.g., for complex robots or 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 have been used to learn inverse kinematics so far.

B. Related Work

Most of the proposed learning approaches to the inverse kinematics problem attempt to learn the inverse kinematics on the velocity level [4], [8]. Locally weighted projection regression has been used to learn the differential inverse kinematics [8]. Here, the local linearity of the mapping $(\dot{\boldsymbol{x}}, \boldsymbol{\theta}) \rightarrow \dot{\boldsymbol{\theta}}$ was proven. This insight allows for a locally consistent algorithm, however, this does not induce global consistency. Global consistency is achieved by selectively generating data [8]. It trains several linear models and choose from them partitioning the input space.

A similar idea to the above of giving a modular constructing of $g^{-1}(\cdot)$ is employed by Susumu and Tachi [9] using neural networks as local models and a gating neural network that chooses one of them. The later approach also suffers from the need of an *oracle* which determines which local model will be used. Finding such an oracle becomes hard in high dimensional spaces.

The multivalued nature of inverse kinematics is addressed by Jordan and Rumelhart [7] who introduced an algorithm for learning multivalued functions and applied it for inverse kinematics. They used a neural network 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 minima, instability, and problems in selecting the network structure.

A more specific application of inverse kinematics is examined in Neumann et al. [6] where focus has been put on learning the inverse kinematics for bi-manual use of tools for a humanoid ASIMO robot. A special neural networks structure, known as reservoir computing, was used to model the function. They investigated how the restriction of the motion improves inverse kinematics modeling. Neural networks also have been used by de Angulo and Torras [5]. The focus of their research was to reduce the number of samples needed by the learning algorithm, achieved by decomposing the robotic arm into virtual robots.

In this paper, we focus on learning the direct inverse kinematics function on the position level, i.e., directly approximating the multivalued function given in Equation (1). We use structured output learning to model the multivalued inverse kinematic relationship. In particular, we learn a probabilistic model in the joint input-output space and obtain prediction for target outputs by maximizing this model for a given input point. Within the same approach, we address the problem of non-uniqueness of $g^{-1}(\cdot)$ in Equation (1). In contrast to the local approaches [4], [8], [9], a *global* inverse kinematics relationship is being modeled here, without partitioning it into individual models. For the structured output learning, we employ the *joint kernel support estimation* [12].

II. Structured Output Learning

In structured output learning we aim to find a function $f : \mathcal{X} \rightarrow \mathcal{Y}$ but unlike the usual setup, here $\mathcal{Y}$ has a *structure*, e.g., natural language parsing where $\mathcal{Y}$ is a set of parse trees [11]. Structured output learning methods [13] have been used with success in a variety of topics: classification with taxonomies [11], label sequence learning [11] [12], sequence alignment [11], natural language parsing [14] [11], handwritten character recognition [15], collective hypertext classification [15], object localization in images [12]. In this paper, we present an application of structured output learning in learning robot's inverse kinematics.

The distinctive feature of the structured learning methods is the consideration of the structure present in the output space that is not taken into account in standard learning methods. Taking into account the structure of this space may improve the learning process. This improvement is achieved by modeling a function $F(x, y)$ that measures the quality of a given (x, y) pair with $x \in \mathcal{X}$ and $y \in \mathcal{Y}$, and finding the most fit y for a given x as prediction. The measure of the fitness allows distinguishing between discriminative and generative methods. In the discriminative approach, the conditional probability of the output given the input is modeled using a function $F(x, y) = p(y|x)$. In contrast, the generative case considers the joint probability distribution $F(x, y) = p(x, y)$. Given $F(x, y)$, the prediction function $f(x)$ is defined as

$$f(x) \quad = \quad \underset{y \in \mathcal{Y}}{\arg\max}\, F(x, y). \qquad (2)$$

A key difference between the two approaches is that given an (x, y) pair discriminative methods attempt to increase $p(y|x)$ and for every $\hat{y} \in \mathcal{Y} \setminus \{y\}$ decrease $p(\hat{y}|x)$ [1] . In contrast, generative methods only increase $p(x, y)$ which has almost no effect on the other regions of the state-space. Generative models, such as joint kernel support estimation [12], are less used as it appears easier to model the conditional than the joint probability distribution [16]. Nevertheless, the benefit of simpler distribution shape does not come for free. The use of discriminative methods has a higher computational complexity and does not handle well noisy data sets [12].

Discriminative methods do not appear to be a natural choice for modeling multivalued functions since all correct labels *compete* with each other during the training procedure. On the other hand, generative methods have the disadvantage of a limited extrapolation capability as $p(x, y) \approx 0$ in the regions of the state-space where was no training data.

It should be noted that there are no restrictions on the input x and the output y. The inputs and outputs can have arbitrary relationships, including a multivalued relation. Thus, learning the function $F(x, y)$ will incorporate the potentially multi-valued input-output mapping. Prediction for a query point x can be obtain by finding the output values maximizing the probabilistic model $F(x, y)$ as shown in Equation (2). In this paper, we employ the generative method (in particular, the joint kernel support estimation [13]) to learn the multivalued inverse kinematic relationship. In the following sections, we describe the employed structured output learning in detail. In Section III, we show how structured output learning can be applied to learn inverse kinematics models.

A. Joint Kernel Support Estimation

Joint kernel support estimation (JKSE) models the joint probability distribution of inputs and outputs as a log-linear model of a joint feature function [12] by

$$p(x, y) \quad = \quad \frac{1}{Z} \exp\left(\boldsymbol{w}^\top \phi(x, y)\right), \tag{3}$$

where $\boldsymbol{w}$ is a vector of weights, $\phi(\cdot, \cdot)$ is a joint feature function used to express task specific knowledge. An implicit definition of $\phi(\cdot, \cdot)$ is also available using kernels, discussed later in this section. $Z = \int_{\mathcal{X}} \int_{\mathcal{Y}} dxdy\ \exp\left(\boldsymbol{w}^\top \phi(x, y)\right)$ is a normalization constant. Note that Z does not depend on x or y, therefore it does not have to be computed during the training or the testing phase. Another beneficial consequence of this independence is that the maximization of Equation (3) with respect to y is equivalent to the following simpler expression

$$f(x) \quad = \quad \arg\max_{y \in \mathcal{Y}}\ \boldsymbol{w}^\top \phi(x, y). \tag{4}$$

We need to find the value of $\boldsymbol{w}$ which generates a $p(x, y)$ that explains the given training dataset $\{(x_i, y_i)\}_{i=1}^m$ with m training points best. The high dimensionality prevents us from representing the entire joint distribution. The number of training data points m is very small compared to the cardinality of $\mathcal{X} \times \mathcal{Y}$, and, thus, we consider that it is sufficient to determine the support of the distribution $p(x, y)$. Suitable algorithms to find this support are one-class support vector machines (OC-SVM), briefly presented in Section II-B. Using OC-SVMs leads to the following form of the prediction function

$$f(x) \quad = \quad \arg\max_{y \in \mathcal{Y}} \sum_{i=1}^{m} \alpha_i k((x, y), (x_i, y_i)), \tag{5}$$

where $k : (\mathcal{X} \times \mathcal{Y}) \times (\mathcal{X} \times \mathcal{Y}) \rightarrow \Re$ is a kernel function defined on joint input-output space. Given $k((x_1, y_1), (x_2, y_2)) = \langle \phi(x_1, y_1), \phi(x_2, y_2) \rangle$ it can be entirely defined by $\phi(\cdot, \cdot)$ or it can be an arbitrary Mercer kernel [17]. α_i are weights determined by the OC-SVM learning procedure (see Section II-B for details). Only some of the α_i are not 0 which leads to a sparse representation of $f(x)$. The training points (x_i, y_i) for which $\alpha_i \neq 0$ are called support vectors. The support vectors form the base of the inference since the rest of the training set does not influence the prediction function. Note that one can influence the number of the support vectors since it depends on the parametrization of the OC-SVM.

B. One-Class SVM training

OC-SVMs are non-parametric machine learning methods which aim to determine the underlying probability distribution from which the points of a training set were drawn. Given a training set $\{(x_i, y_i)\}_{i=1}^m$, they search for a hyperplane parameterized by $\boldsymbol{w}$ and ρ, which separates the points from the origin with the largest possible margin. It is equivalent to the following optimization problem

$$\min_{\boldsymbol{w} \in \mathcal{H}, \xi_i \in \Re, \rho \in \Re} \frac{1}{2} ||\boldsymbol{w}||^2 + \frac{1}{vm} \sum_{i=1}^{m} \xi_i - \rho \tag{6}$$

subject to

$$\begin{aligned} \langle \boldsymbol{w}, \phi(x_i, y_i) \rangle_{\mathcal{H}} &\geq \rho - \xi_i, \\ \xi_i &\geq 0, \qquad \forall i = \overline{1, m} \end{aligned}$$

where $\mathcal{H}$ is a latent Hilbert space induced by the joint kernel k – for details about OC-SVMs consult [17], [18]. The parameter $v \in (0, 1]$ plays an important role in the learning process, since it provides a lower bound on the fraction of support vectors, influencing the time complexity of both the training and the testing phases [17], [18]. This property becomes important when OC-SVMs are used in a real world online application.

After solving the optimization problem, the prediction of a new (x, y) data point has the form of Equation (5) without taking the maximum over all possible y-s [17], [18].

Since Equation (6) is a quadratic optimization problem with linear constrains, the training algorithm has time complexity $\mathcal{O}(m^3)$, the prediction $\mathcal{O}(m)$, see [17], [18].

Note that the training process of OC-SVMs needs to evaluate the kernel function for m points but it does not depend on the output space $\mathcal{Y}$. Therefore, the complexity of JKSE training is a function of the size of the training data and do not dependent of the output space.

[1]Note that $\mathcal{Y}$ can be prohibitively large leading to very slow learning.

III. Structured Learning of Direct Inverse Kinematics

In this section, we apply JKSE to learn direct inverse kinematics. Neither the training nor the prediction phase of the JKSE, when applied to inverse kinematics, is equivocal, therefore we propose a potential model training setup and present an efficient testing algorithm as well. We also discuss how to make this method usable in online setting and on real systems.

A. Training data collection

In order to perform the parameter learning of the model, we require a set of training points. Constructing such a set for inverse kinematics is not straightforward. A training set $D = \{(\boldsymbol{x}_i, \boldsymbol{\theta}_i)\}_{i=1}^m$ must contain pairs of end-effector positions and the corresponding joint configurations. Providing samples from the whole configuration space is unfeasible due to the curse of dimensionality. To avoid oversampling, we only sample trajectories with end-effector positions in the area where the actual task will take place. For example, consider the task of drawing a figure eight as shown in Figure 3(a). For this task, the desired end-effector trajectory lies in a plane. Hence, sampling in a volume around this plane fully suffices. Generating task-appropriate training data results not only in faster training and faster prediction but also in higher precision. Task-dependent training sets are also motivated by the weak extrapolation capabilities of the JKSE, discussed in Section II.

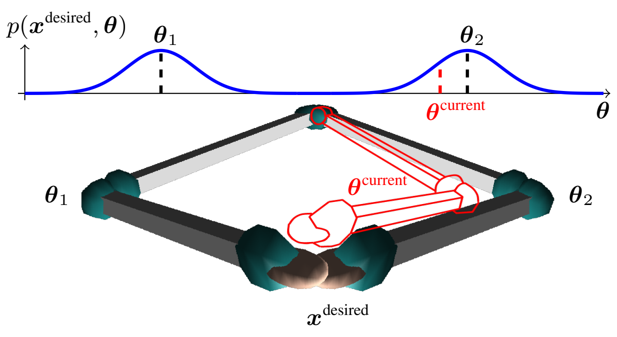
We highlight that JKSE handles different input points with the same label well. Hence, the training set can contain ambiguous data, such as $(\boldsymbol{x}_i, \boldsymbol{\theta}_i) \in D$ and $(\boldsymbol{x}_j, \boldsymbol{\theta}_j) \in D$ where $\boldsymbol{x}_i = \boldsymbol{x}_j$ but $\boldsymbol{\theta}_i \neq \boldsymbol{\theta}_j$. Such training data frequently appears in inverse kinematics when the end-effector position was reached with different joint configurations.

B. Online Application in Direct Inverse Kinematics

Next, we present how a trained JKSE model can be used for converting an end-effector trajectory into joint-space during its execution. At every time step, the prediction of the joint-state $\boldsymbol{\theta}^{\text{desired}}$ matching a desired end-effector position $\boldsymbol{x}^{\text{desired}}$ is expressed by Equation (5) where $x = \boldsymbol{x}^{\text{desired}}$ and $\mathcal{Y} = \Re^n$. The main difficulty in using Equation (5) is that it involves a maximization over $\Re^n$. This step is intractable when n is big. To ease this maximization, we assume that the prediction function is smooth and we find a local maxima close to the current joint configuration of the robot – $\boldsymbol{\theta}^{\text{current}}$. This assumption must hold in order to avoid sudden changes in the joint-space configuration.

The previous assumption simplifies the optimization problem in Equation (5) since we can apply gradient descent based optimization starting from the current joint position of the robot. The gradient can either be determined analytically or using a finite difference approximation [19]. In both cases significant performance improvements are obtained.

Suppose we are looking for a joint configuration for $\boldsymbol{x}^{\text{desired}}$ and during the training $\boldsymbol{\theta}_1$ and $\boldsymbol{\theta}_2$ had the same $\boldsymbol{x}^{\text{desired}}$ end-effector position. Due to the gradient search, the algorithm will choose the joint position that is closer to the current joint configuration $\boldsymbol{\theta}^{\text{current}}$, see the illustration on Figure 1. However, note that Figure 1 is misleading as the calculation of the exact joint probability is unfeasible given that it requires the evaluation of the normalization constant from Equation (3). Instead, we use the expression from Equation (4) to obtain the desired joint configuration since it avoids the determination of that constant. Note that if $\boldsymbol{\theta}_1$ and $\boldsymbol{\theta}_2$ were close to each other, both of them would be acceptable solutions of the maximization.



$p(\boldsymbol{x}^{\text{desired}}, \boldsymbol{\theta})$
$\boldsymbol{\theta}_1$
$\boldsymbol{\theta}_2$
$\boldsymbol{\theta}^{\text{current}}$
$\boldsymbol{\theta}$
$\boldsymbol{\theta}_1$
$\boldsymbol{\theta}^{\text{current}}$
$\boldsymbol{\theta}_2$
$\boldsymbol{x}^{\text{desired}}$

Fig. 1. Illustration of the structured output inverse kinematics algorithm prediction scheme. During the training process $\boldsymbol{x}^{\text{desired}}$ has been reached by two different joint configurations $\boldsymbol{\theta}_1$ and $\boldsymbol{\theta}_2$ (solid arms), therefore, $p(\boldsymbol{x}^{\text{desired}}, \boldsymbol{\theta}_1) = p(\boldsymbol{x}^{\text{desired}}, \boldsymbol{\theta}_2)$. However, as the current joint configuration $\boldsymbol{\theta}^{\text{current}}$ is closer to $\boldsymbol{\theta}_2$, the algorithm chooses a prediction that is closer to $\boldsymbol{\theta}_2$.

The pseudo-code of the proposed method is presented in Algorithm 1. The training phase is straightforward: given a data set $D = \{(\boldsymbol{x}_i, \boldsymbol{\theta}_i)\}_{i=1}^m$, we train a model $\mathcal{M}$ on the joint data using OC-SVM (OC-SVM-training). The trajectory tracking looks as follows. We generate the desired end-effector position $\boldsymbol{x}^{\text{desired}}$, and predict the corresponding joint configuration $\boldsymbol{\theta}^{\text{desired}}$. This joint configuration is the one that maximizes the prediction function of the OC-SVM (OC-SVM-prediction), since OC-SVM-prediction expresses the joint probability $p(\boldsymbol{x}^{\text{desired}}, \boldsymbol{\theta}^{\text{desired}})$. To perform the maximization, we use gradient search starting from the current joint position $\boldsymbol{\theta}^{\text{current}}$. The algorithm ends when the tracking task is finished.

As we aim at real-world applications, a low algorithm complexity is important. The training phase is equivalent to the OC-SVM training and thus has a complexity of $\mathcal{O}(m^3)$. This step can often be done offline, therefore it is not always crucial and the development of online versions with

Algorithm 1 Structured output inverse kinematics learning

```
INPUT: $D = \{(\boldsymbol{x}_i, \boldsymbol{\theta}_i)\}_{i=1}^m$, $v$, $k$

$\mathcal{M}$ ← OC-SVM-training ($D$, $v$, $k$)
while task is not over do
    $\boldsymbol{x}^{\text{desired}}$ ← next-position ()
    $\boldsymbol{\theta}^{\text{desired}}$ ← gradient-maximization ($\boldsymbol{\theta}^{\text{current}}$,
                      OC-SVM-prediction ($\mathcal{M}$, ($\boldsymbol{x}^{\text{desired}}$, ·)))
end while
```

lower complexity appears possible. On the other hand, the complexity of the prediction is $\mathcal{O}(ml)$ where l is the number of function evaluations required by the gradient search. By choosing an efficient gradient search, e.g., second order gradient descent, we can keep l small and achieve better performance.

The free parameter v regulates a lower bound on the fraction of support vectors, see Section II-B. It allows us controlling the trade-off between faster prediction and higher precision. The choice of the kernel function k is also an important aspect that affects the speed of the algorithm, e.g., linear kernels can be evaluated faster than Gaussian kernels but have less expressive power. Often, hyper-parameters of the kernel play an important role as well, e.g., for of a Gaussian kernel, one can control the trade-off between better generalization and higher precision with the parameters.

IV. Evaluations

In this section, we present the evaluation of the proposed method for task-space tracking. The algorithm is applied to learn the inverse kinematics of a robotic arm, and track a figure eight and a star-like figure, respectively. Both tasks are defined in Cartesian space. The first experiment is conducted on a simulated 3 degrees of freedom (DoF) robot where one DoF is redundant. Here, we show how our algorithm is capable of learning inverse kinematics of a redundant robot system from an ambiguous data-set. We also test the method in real world settings, on a 7-DoF Barrett whole arm manipulator (WAM).

As discussed in Section III, a well chosen parametrization of the learning method has high impact on the tracking performance. Hence, the kernel function of the OC-SVM has to be chosen, as to incorporate as many prior knowledge about inverse kinematics as possible. This is a joint kernel of the end-effector position $\boldsymbol{x}$ and joint coordinates $\boldsymbol{\theta}$. A natural choice for the variable of the joint kernel would be to simple concatenate $\boldsymbol{x}$ and $\boldsymbol{\theta}$ into one vector. However, adding the sines and cosines of the joint angles, i.e., $\phi(\boldsymbol{x},\boldsymbol{\theta}) = [\boldsymbol{x} \;\; \boldsymbol{\theta} \;\; \sin(\boldsymbol{\theta}) \;\; \cos(\boldsymbol{\theta})]^\top$, improves the prediction, since forward kinematics highly depends on these values. In this way, we include prior knowledge of inverse kinematics into the kernel function. During the experiments we employ a Gaussian kernel with different parameters for each task.

The gradient search of Algorithm 1 has an important role in the tracking performance as well. In the 3-DoF simulation we use a BFGS method [19] with finite difference approximation of the gradient, whereas, Nelder-Mead method [19] is employed in the Barrett experiment. Once the desired joint coordinates are known, joint velocities and joint accelerations are obtained by numerical differentiation.

A. *Evaluation on a Simulated 3-DoF Robot*

In this experiment, we present how the proposed method handles ambiguous data-sets. The training data-set is constructed as follows: using a joint-space controller, we sample data uniformly from the volume around the plane where the figure eight task will take place. The sampling process

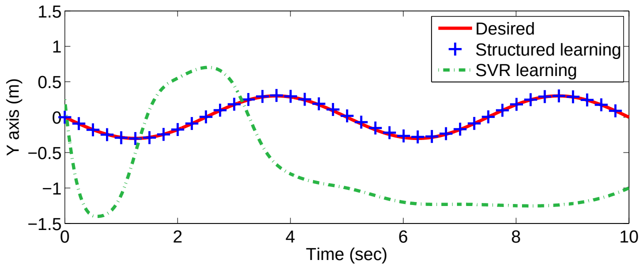


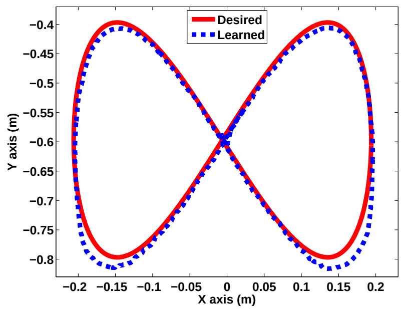
Fig. 2. Tracking precision of the figure eight along the y axis. Simple support vector regression (SVR) completely fails to learn the inverse kinematics function when the training data contains ambiguous points, whereas structured output inverse kinematics achieves high accuracy.

is repeated with two different initial joint configurations, see Figure 1. Then, the two data-sets are merged. The final training set contains points with the same end-effector position but with different join configuration. The training set contains 34996 samples, 17498 from each collection process. We use libSVM [20] to train the OC-SVM with parameters $v = 0.1$ and $g = 500$. Here, g is the parameter of the Gaussian kernel.

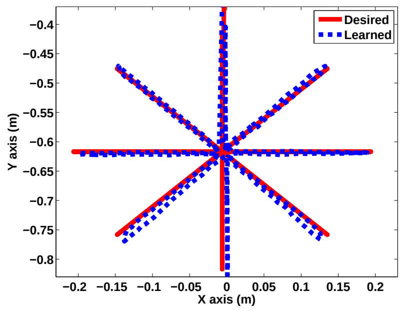
To substantiate that learning inverse kinematics is not trivial for ambiguous training data, we train a simple support vector regression (SVR) model using the ambiguous data-set. SVR completely fails to learn the inverse kinematics function. The same failure happens when the take out the 95% of the data, originated from the experiment with the second initial posture, as making the task significantly easier. The comparison of the two methods is presented on Figure 2, revealing that ordinary regression methods do not have the expressive power to model multivalued functions, like inverse kinematics.

B. *Evaluation on a 7-DoF Barrett WAM Robot*

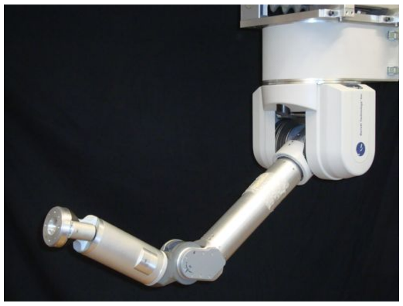
We apply our method on a 7-DoF Barrett whole arm manipulator for figure eight and star-like figure tracking experiments. We highlight that the tracking experiment is conducted in *real time*, proving that the computational requirements of the algorithm are feasible in real time settings. Training data collection is similar to the previous experiment, we sample random data points from the plane where the figure eight and the star-like tracking will take place. We use the same training data for both experiments. The training set contains 57142 points. To train the OC-SVM algorithm and preform the prediction, we use a modified version of libSVM [20] with parameters $v = 0.1$ and $g = 1000$. The tracking performance of the structured output inverse kinematics method is shown on Figure 3(a) for the figure eight task, and on Figure 3(b) for the star-like task with an average tracking error of 3 and 2 centimeters, respectively. Figures were made of 500 control points and for the rest of the points we used linear interpolation in the joint space. Thus, the computations for each desired joint angle took 20 milliseconds. Both figures show the results for the first 10 seconds of tracking.



(a) Figure eight tracking of the 7-DoF Barrett WAM.



(b) Star-like figure tracking of the 7-DoF Barrett WAM.



(c) Barrett Whole-Arm-Manipulator (WAM)

Fig. 3. Task-space tracking control for the figure eight task. The real world application results in good tracking accuracy for the (a) figure eight task and the (b) star-like figure. The tracking error is due to the imperfectness of our approach and the inaccurate dynamical model of the robot, unmodeled friction, and the noisy measurements. (c) 7-DoF Barrett whole arm manipulator used for the task-space tracking.

V. Conclusions and Future Work

In this paper, we proposed an algorithm to learn the inverse kinematics function of robots with redundant manipulators using structured output machine learning methods. The method learns the *direct* inverse kinematics function on position level. We also addressed the problem of non-uniqueness of inverse kinematics, highlighting that regular regression algorithms are not capable of modeling it. The key idea is that instead of finding a direct approximation of inverse kinematics, we modeled the joint probability distribution of the end-effector positions and joint angles. Joint kernel support estimation has been employed to model the joint probability distribution. The proposed method is supported by real world experiments on a 7-DoF Barrett WAM. Results show that high performance of task-space tracking can be achieved.

We consider that the approximation of inverse kinematics can be improved in several ways. Applying more efficient gradient search algorithms might speed-up the prediction function. We want to analyze how discriminative structured output methods can be used to model the dependencies between end-effector positions and joint angles. To succeed, we have to overcome the disadvantages of discriminative approaches presented in Section II.

VI. Acknowledgments

The authors wish 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 – ”Innovative doctoral studies in a knowledge based society”.

References

[1] B. Siciliano, “Kinematic control of redundant robot manipulators: A tutorial,” *Journal of Intelligent and Robotic Systems*, vol. 3, no. 3, pp. 201–212, 1990.

[2] L. Sciavicco and B. Siciliano, *Modelling and Control of Robot Manipulators (Advanced Textbooks in Control and Signal Processing)*, 2nd ed., ser. Advanced textbooks in control and signal processing. Springer, January 2005.

[3] C. Salaün, V. Padois, and O. Sigaud, “Learning forward models for the operational space control of redundant robots,” in *From Motor Learning to Interaction Learning in Robots*, 2010, pp. 169–192.

[4] A. D’Souza, S. Vijayakumar, and S. Schaal, “Learning inverse kinematics,” in *IEEE International Conference on Intelligent Robots and Systems (IROS 2001)*. Piscataway, NJ: IEEE, 2001.

[5] V. R. de Angulo and C. Torras, “Learning inverse kinematics: Reduced sampling through decomposition into virtual robots,” *IEEE Transactions on Systems, Man, and Cybernetics, Part B*, vol. 38, no. 6, pp. 1571–1577, 2008.

[6] K. Neumann, M. Rolf, J. J. Steil, and M. Gienger, “Learning inverse kinematics for pose-constraint bi-manual movements,” in *SAB*, 2010, pp. 478–488.

[7] M. I. Jordan and D. E. Rumelhart, “Forward models: Supervised learning with a distal teacher,” *Cognitive Science*, vol. 16, pp. 307–354, 1992.

[8] G. Tevatia and S. Schaal, “Inverse kinematics for humanoid robots,” in *International Conference on Robotics and Automation (IRCA 2000)*, 2000, pp. 294–299.

[9] E. O. Susumu and S. Tachi, “Inverse kinematics learning by modular architecture neural networks,” in *in Proc. IEEE International Conference on Robotics and Automation, 2001*, 2001, pp. 1006–1012.

[10] D. Demers and K. Kreutz-Delgado, “Learning global direct inverse kinematics,” in *Advances in Neural Information Processing Systems*. Morgan Kaufmann, 1992, pp. 589–595.

[11] I. Tsochantaridis, T. Joachims, T. Hofmann, and Y. Altun, “Large margin methods for structured and interdependent output variables.” *Journal of Machine Learning Research*, vol. 6, pp. 1453–1484, 2005.

[12] C. H. Lampert and M. B. Blaschko, “Structured prediction by joint kernel support estimation,” *Machine Learning*, vol. 77, pp. 249–269, December 2009.

[13] G. H. Bakir, T. Hofmann, B. Schölkopf, A. J. Smola, B. Taskar, and S. V. N. Vishwanathan, Eds., *Predicting Structured Data*, ser. Neural Information Processing. The MIT Press, September 2007.

[14] A. McCallum and C. Sutton, “An introduction to conditional random fields for relational learning,” in *Introduction to Statistical Relational Learning*, L. Getoor and B. Taskar, Eds. MIT Press, 2006.

[15] B. Taskar, C. Guestrin, and D. Koller, “Max-margin Markov networks,” in *Advances in Neural Information Processing Systems 16*, S. Thrun, L. Saul, and B. Schölkopf, Eds. Cambridge, MA: MIT Press, 2004.

[16] C. M. Bishop, *Pattern Recognition and Machine Learning*, 1st ed. Springer, October 2007.

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

[18] B. Schölkopf, J. C. Platt, J. C. Shawe-Taylor, A. J. Smola, and R. C. Williamson, “Estimating the support of a high-dimensional distribution,” *Neural Computation*, vol. 13, pp. 1443–1471, July 2001.

[19] M. Avriel, *Nonlinear Programming: Analysis and Methods*. Dover Publishing, 2003.

[20] C.-C. Chang and C.-J. Lin, *LIBSVM: a library for support vector machines*, 2001, software http://www.csie.ntu.edu.tw/~cjlin/libsvm.