Rapid Learning of Humanoid Body Schemas with Kinematic Bézier Maps

Stefan Ulbrich*, Vicente Ruiz de Angulo†, Tamim Asfour*, Carme Torras† and Rüdiger Dillmann*
* Institute for Anthropomatics,
Karlsruhe Institute of Technology, Germany
Email: [ulbrich,asfour,dillmann]@ira.uka.de
† Institut de Robòtica i Informàtica Industrial,
CSIC-UPC, Barcelona
Email: [ruiz,torras]@iri.upc.edu

Abstract— **This paper addresses the problem of hand-eye coordination and, more specifically, tool-eye recalibration of humanoid robots. Inspired by results from neuroscience, a novel method to learn the forward kinematics model as part of the body schema of humanoid robots is presented. By making extensive use of techniques borrowed from the field of computer-aided geometry, the proposed *Kinematic Bézier Maps (KB-Maps)* permit reducing this complex problem to a linearly-solvable, although high-dimensional, one. Therefore, in the absence of noise, an *exact* kinematic model is obtained. This leads to rapid learning which, unlike in other approaches, is combined with good extrapolation capabilities. These promising theoretical advantages have been validated through simulation, and the applicability of the method to real hardware has been demonstrated through experiments on the humanoid robot ARMAR-IIIa.**

I. INTRODUCTION

With increasingly complex robots –especially humanoids– the calibration process of the arms and other kinematic chains, and hence the prediction of the effects of joint movements, becomes a difficult, time-consuming and often expensive task. This process has to be repeated every time the tool center point (TCP) of the robot changes, e.g. if the robot accidently suffers deformation or –even more important– if the robot intends to interact with its environment with a tool. The hand-eye calibration by traditional means then becomes nearly impossible. Humans solve the problem successfully by pure self-observation, which has led to the adaptation of biologically-inspired mechanisms to the field of robotics.

In neuroscience, it is common knowledge that there exists a body schema that correlates proprioceptive sensor information, e.g. joint configurations, with the visible shape of the body [10]. It also represents an unconscious awareness of the current body state [11]. Experiments with both macaque monkeys and humans showed that the body schema is neither congenital nor rigid but rather learnable and adaptable, as shown by Maraviata et al [9]. For instance, an experiment examines the proximal visual receptive field (the area in cartesian space where stimuli activate the neurons associated to grasping) of the macaque monkeys. It was shown that this field was enlarged by the length of a tool that the monkeys used once they had been trained to do so. This leads to the conclusion that the tool itself became incorporated into the monkey's own body schema. Similar conclusions were drawn from experiments with human patients who suffered from brain damage or phantom pain after having lost a limb. This leads to the assumption that in the human brain similar processes exist as in the monkey's. Further observations by Stamenov [14] showed that the body schema is not a well-formed pattern but rather a set of several connected groups of neurons that represent opportunistically learned manifolds and that are distributed over regions in the brain.

As a consequence of these results, there is a great interest among robotics researchers to emulate this adaptability with techniques from machine learning. In most robotics works, the term 'learning of the body schema' is restricted to the sub-symbolic learning of the relation between the proprioceptive sensors for the joint configuration $\boldsymbol{\theta}$ and the visual position $\boldsymbol{x}$ of the end-effector. Therefore, it is basically limited to the approximation of the forward kinematics (FK), the inverse and local inverse kinematics (IK) from pairs of joint angles and cartesian coordinates:

$$f(\boldsymbol{\theta}) = \boldsymbol{x}, \quad f^{-1}(\boldsymbol{x}) = \boldsymbol{\theta} \text{ and } f^{-1}(\dot{\boldsymbol{x}}) = \dot{\boldsymbol{\theta}}. \qquad (1)$$

In general, the approximation of the latter two functions (with a high number of DoF) is an ill-posed problem as the same position can be generated by different joint configurations. However, the approximation of the FK can be used to solve the IK problem in a flexible way via techniques such as *resolved motion rate control (RMRC)* [16]. Thus, the current paper focuses on learning the FK mapping from tuples $(\boldsymbol{\theta}, \boldsymbol{x})$, which will be referred to as *training experiences, samples* or *training data.*

The main difficulty of the approximation of the FK lies in the fact that it is a highly non-linear function with non-redundant input variables, each of them significantly

The work described in this paper was partially conducted within the EU Cognitive Systems projects GRASP (FP7- 215821) and PACO-PLUS (FP6-027657) funded by the European Commission.

The authors acknowledge support from the Generalitat de Catalunya under the consolidated Robotics group, and from the Spanish Ministry of Science and Education, under the project DPI2007-60858.

influencing the result. Hence, it requires a large amount of training experiences that grows exponentially with the number of DoF of the kinematic chain. This complexity can be reduced by decomposing the robot into kinematic subchains as proposed by Ruiz et al. [2][3], but at the expense of increasing the demands on the robot's perceptive abilities or limiting the applicability to a family of robots.

Parametrized Self-Organizing Maps (PSOMs) [15] have been often used to learn kinematics problems because of its versatility and interpolation abilities. However, they require that the training samples are distributed in a regular grid (although this is mitigated in [8]) and, specially, they are not well suited to on-line learning. High-dimensional kinematic chains have been handled by using *locally weighted projection regression (LWPR)* [4]. This algorithm creates linear models locally valid for the training data, which are combined into a weighted sum that eventually approximates the FK or local IK. PSOM has in common with the LWPR approach that they quickly produce locally valid approximations but again require a large amount of training data for a complete model, as they lack good extrapolation capabilities. An exact encoding of the FK of robots with rotational joints is not possible as both approaches use approximations that are not capable of describing the product space of rotations with a finite number of samples. However, both can be used to learn a local IK approximation and are thus capable of solving the IK problem directly.

A different approach was recently proposed by Hersch et al [7], where the parameters of the FK in Denavit-Hartenberg convention are learned directly by an optimization algorithm. This optimization eventually leads to the creation of a body schema with good extrapolation capabilities and even converges to an exact model in simulation. However, this method suffers from a low learning speed –even in simulation.

To the best of our knowledge, there is not yet an algorithm that can learn a FK mapping *exactly* and in an efficient way. This is the aim of this work, where we use techniques from the field of Computational Geometry –namely, rational Bézier tensor-product functions. Derived from these functions the *Kinematic Bézier Maps (KB-Maps)* were created. In contrast to all other approaches, this representation permits an exact encoding of the FK, which is robust to sensor noise, and it allows the learning algorithm to keep the same complexity regardless of the number of training experiences. Moreover, it exhibits good extrapolation capabilities even when only a relatively small number of experiences can be provided that lie close to one another. The key aspect of the KB-Maps is that they transform a highly non-linear problem into a higher-dimensional, but linearly solvable, equation system.

The paper is structured as follows. In the next section, a brief introduction to the underlying geometrical techniques is provided. Section 3 describes their application in the KB-Maps to encode FK. Two algorithms suitable to perform the learning are presented in Section 4. Then, in Section 5, the proposed method is applied to the humanoid robot ARMAR-IIIa [1] in both real experiments and simulation, and the obtained results are discussed. The paper concludes with a brief account of the contributions and an outlook on future work.

II. Forward Kinematics Representation in Bézier Form

A. Mathematical Fundamentals

1) Bézier Curves: In affine space, every polynomial spatial curve $\boldsymbol{b}(s)$ of degree n has an unique Bézier form [13] [6]:

$$\boldsymbol{b}(s) = \sum_{i=0}^{n} \boldsymbol{b}_i \cdot B_i^n(s), \text{ with } B_i^n(s) := \binom{n}{i} \cdot s^i \cdot (1-s)^{n-i}, \quad (2)$$

where every point $\boldsymbol{b}(s)$ on the curve is the result of an affine combination of a set of $n+1$ *control points* $\boldsymbol{b}_i$ weighted by the well-known *Bernstein polynomials* $B_i^n(s)$ that serve as a basis for all polynomial curves of degree n. The Bézier form of the curve's derivative

$$\dot{\boldsymbol{b}}(s) = n \cdot \sum_{i=0}^{n-1} \Delta\boldsymbol{b}_i \cdot B_i^{n-1}(s) \quad (3)$$

can be obtained easily by the construction of the forward differences $\Delta\boldsymbol{b}_i$ with

$$\Delta\boldsymbol{b}_i := \boldsymbol{b}_{i+1} - \boldsymbol{b}_i.$$

2) Tensor Product Bézier Surfaces: Polynomial surfaces and higher multivariate functions can also be expressed in Bézier form. If they are polynomial of degree n in their main directions (when only one parameter is variable), the function can be expressed as a tensor product of two or more Bézier curves. For example, a polynomial surface of degree n, $\boldsymbol{b}(s_1, s_2)$, has the tensor product Bézier form

$$\boldsymbol{b}(s_1, s_2) = \sum_{i_1=0}^{n} \cdot \left(\sum_{i_2=0}^{n} \boldsymbol{b}_{i_1,i_2} \cdot B_{i_2}^n(s_2) \right) \cdot B_{i_1}^n(s_1). \quad (4)$$

The net of $(n+1)^2$ points $\boldsymbol{b}_{i_1,i_2}$ forms the *control net*. In general, a d-dimensional tensor product Bézier of degree n can be represented as

$$\boldsymbol{b}(\boldsymbol{s}) = \sum_{\boldsymbol{i}} \boldsymbol{b}_{\boldsymbol{i}} \cdot B_{\boldsymbol{i}}^n(\boldsymbol{s}) \quad (5)$$

where $\boldsymbol{i} := (i_1, i_2, \dots, i_d)$ represents a vector of indices going through the set $\mathscr{I}_n = \{(i_1, i_2, \dots, i_d) \text{ s.t. } i_k \in \{0, \dots, n\}\}$ of index vectors addressing the points of the control net , $\boldsymbol{s} := (s_1, s_2, \dots, s_d)$ is the parameter vector, and

$$B_{\boldsymbol{i}}(\boldsymbol{s}) := \prod_{k=1}^{d} B_{i_k}(s_k) \quad (6)$$

are the products of all Bernstein polynomials within each summand. In total, the control net of the tensor product Bézier representation is formed by $(n+1)^d$ control points.

3) Rational Polynomials and Rational Bézier Form: Although FK can be approximated by polynomials, an exact representation of the FK requires a more complex class of functions, e.g. *rational polynomials* [5]. Rational polynomial functions are similar to affine polynomial functions except for the fact that they are defined in the *projective space* $\mathscr{P}$. Simplifying, $\mathscr{P}$ is a space with an additional dimension and elements of the form

$$\mathbb{p} = \begin{bmatrix} \gamma \boldsymbol{p} \\ \gamma \end{bmatrix} \text{ or short } \mathbb{p} = \gamma \cdot \boldsymbol{p}, \quad \gamma \in \mathbb{R} \setminus 0,$$

where $\boldsymbol{p}$ is an affine point and γ is called *homogeneous coordinate* or *weight* of $\mathbb{p}$. Any projective point $\mathbb{p} \in \mathscr{P}$ can be understood as a ray that originates from the the *projective center* $(0, \ldots, 0)$ and intersects the affine space at $\boldsymbol{p}$ when $\gamma = 1$. The intersection point is called the *affine image* of $\mathbb{p}$ and division by γ is called *projection (onto the affine space)*.

On projection into the affine space, rational polynomials generally become more complex functions and may loose their polynomial characteristics (see Fig. 1). Still, in homogeneous space, there does exist the same previously introduced unique Bézier form for curves and surfaces

$$\mathbb{b}(\boldsymbol{s}) = \sum_{\boldsymbol{i}} \mathbb{b}_{\boldsymbol{i}} \cdot B_{\boldsymbol{i}}(\boldsymbol{s}) = \begin{bmatrix} \boldsymbol{b}(\boldsymbol{s}) \\ \gamma(\boldsymbol{s}) \end{bmatrix} = \begin{bmatrix} \sum_{\boldsymbol{i}} \gamma_i \boldsymbol{b}_{\boldsymbol{i}} \cdot B_{\boldsymbol{i}}(\boldsymbol{s}) \\ \sum_{\boldsymbol{i}} \gamma_i \cdot B_{\boldsymbol{i}}(\boldsymbol{s}) \end{bmatrix}.$$

and, after affine projection, the rational Bézier form

$$\mathbf{b}(\boldsymbol{s}) = \frac{\boldsymbol{b}(\boldsymbol{s})}{\gamma(\boldsymbol{s})} = \frac{\sum_{\boldsymbol{i}} \gamma_i \cdot \boldsymbol{b}_{\boldsymbol{i}} \cdot B_{\boldsymbol{i}}(\boldsymbol{s})}{\sum_{\boldsymbol{i}} \gamma_i \cdot B_{\boldsymbol{i}}(\boldsymbol{s})}. \tag{7}$$

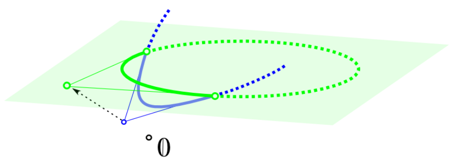


Fig. 1. The projection of a parabola in $\mathscr{P}$ onto a circle.

B. Forward Kinematics Representation: The One-dimensional Case

In this section, we show how to use the techniques presented above to come up with the Bézier representation of the forward kinematics of a robot with rotational joints.

The end-effector of a single-joint ideal robot moves along a circular trajectory when the value θ of its joint changes. In general, the FK of a robot with d degrees of freedom is simply a composition of d circles. Therefore, the basic geometric objects that we need to represent are circles and more generally their deformations. The only deformation of circles that we consider are ellipses. We expect that this flexibility contributes to a better conformation to the real function that has to be learned, that may be biased by the sensorial system or gravity.

To explain more clearly our representation of FK, we begin by showing it for a single degree of freedom. As declared before, our model is able to represent a family of ellipses including the circle.

Homogeneous polynomials of degree two become conics when projected onto the affine space and, for every conic, there exists a rational Bézier representations of degree two [5]. In particular, a rational Bézier curve

$$\mathbf{b}(s) = \frac{\sum_{i=0}^{2} \gamma_i \cdot \boldsymbol{b}_i \cdot B_i^2(s)}{\sum_{i=0}^{2} \gamma_i \cdot B_i^2(s)} \tag{8}$$

is an ellipse if

1) the weights γ_0 and γ_2 are equal, and
2) $\gamma_1 / \gamma_0 = \gamma_1 / \gamma_2 < 1$.

To be a circle, in addition it has to satisfy that a) the control points form an isosceles triangle with a common angle α, and b) $\gamma_1 / \gamma_0 = \cos \alpha$. Note that all conditions refer to proportions between weights because multiplying every weight by a constant leaves (8) unchanged.

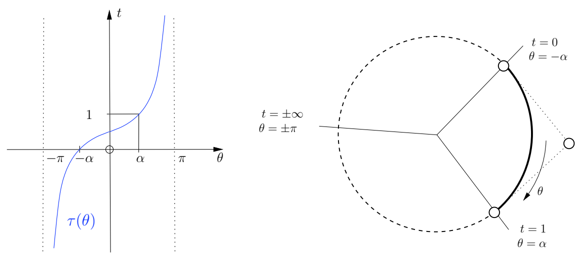
Imposing $\gamma_0 = \gamma_2 = 1$ and fixing γ_1 to an arbitrary constant smaller than one, the ellipse conditions are satisfied. At the same time, doing this, the circle is not excluded from the family of ellipses potentially represented by the Bézier form, since for any γ_1 it is possible to find a set of control points forming an isosceles triangle with a common angle whose cosine is γ_1. Thus, if learning data comes from a circle and we have enough points to constrain the model, we will obtain a circle. By imposing $\gamma_0 = 1$, the redundancy in the representation induced by proportionality in the weights is eliminated. Imposing $\gamma_0 = \gamma_2$ and fixing γ_1 to a constant has the effect of limiting the kind of ellipses that can be used to fit the FK data.

The joint effect of these constraints is that the number of sample points required to determine the Bézier form is greatly reduced (see Section III): in the one-dimensional case, it is reduced from 5 (required in general for an ellipse) to 3. Note that this is also the minimum number of sample points required if we would have assumed a model based only on circles. As a consequence, we have a more flexible model without having to pay a tribute in increased number of required data.

Our model is still incomplete. For $\mathbf{b}(s)$ to represent a complete ellipse, s must go from $-\infty$ to ∞. Instead, the data samples and the robot commands are joint encoder values θ, ranging from $-\pi$ to π. We must transform θ before being used as input to the Bézier form. We have chosen the following transformation

$$\tau : [-\pi, \pi] \mapsto \mathbb{R}, \quad \tau(\theta) = \frac{\tan(\theta/2)}{2 \cdot \tan(\alpha/2)} + \frac{1}{2}. \tag{9}$$

where $\alpha = \arccos(\gamma_1)$, see Fig. 2(a). In fact, it is more practical to fix indirectly γ_1 by choosing first an arbitrary α and setting $\gamma_1 = \cos(\alpha)$. The sense of this transformation is that, when $\mathbf{b}(s)$ becomes exactly a circle, α becomes the common angle in the isosceles triangle formed by the control points, see Fig. 2(b). In this case, it can be proven that θ becomes the angular parameterization of the circle measured in radian units in $\mathbf{b}(\tau(\theta))$, which is the final form of the one-dimensional KB-Maps.



(a) The τ transformation (b) The parameterization

Fig. 2. Transformation from a joint angle to the corresponding parameter of the Bézier form.

C. Forward Kinematics Representation: The Multidimensional Case

We like to represent a composition of d ellipses with a Bézier form, understood in the same sense that a pure FK is a composition of d circles: when all variables but one are fixed the resulting curve must be an ellipse, i.e., the isoparametric curves of the Bézier form are ellipses. To accomplish this, we set the weights $\gamma_{i_1,i_2,\ldots,i_d}$ of control points $\mathbb{b}_{i_1,\ldots,i_d}$ to $\gamma^{ones(i_1,\ldots,i_d)}$, where $ones()$ returns the number of ones in the arguments and γ is an arbitrary constant minor than one. The proof is in the Appendix. The value γ can be selected like in the one-dimensional case, via the cosine of an arbitrary angle, $\gamma = \cos\alpha$.

With arguments similar to those for the one-dimensional case, we can state that each of the ellipses defined by the isoparametric curves in the main directions can take the shape of a circle. Therefore, if we have enough data points to determine the surface (3^d, see Section III) coming from an exact FK, the Bézier form will reproduce exactly the robot kinematics. In that case, the implicit control points (named $\mathbb{q}_k$ in the Appendix) appearing in the expression of the isoparametric curves in the main directions will form an isosceles triangle. In fact, the triangles will be congruent for all main directions, having all the same common angle α. But, of course, the circles in the main directions are anyway unrelated and can be completely different.

Finally, to complete the model we must include the transformation $\tau(\boldsymbol{\theta})$ of the input encoder vector, $\boldsymbol{\theta} = (\theta_1,\ldots,\theta_d)$. The rationale is, as in the one-dimensional case, to establish a correspondence between the encoder values that are given in uniform angular units (radians) and the Bézier parameters $\boldsymbol{s}$ that yield the adequate Bézier surface points in the context of an exact FK. In sum, this is the KB-Maps model for FK:

$$\boldsymbol{f}(\boldsymbol{\theta};\boldsymbol{G}) \equiv \mathbf{b}(\tau(\boldsymbol{\theta})) = \frac{\sum_{\boldsymbol{i}} \gamma_{\boldsymbol{i}} \cdot \boldsymbol{b}_{\boldsymbol{i}} \cdot B^2_{\boldsymbol{i}}(\tau(\boldsymbol{\theta}))}{\sum_{\boldsymbol{i}} \gamma_{\boldsymbol{i}} \cdot B^2_{\boldsymbol{i}}(\tau(\boldsymbol{\theta}))} \qquad (10)$$
$$\gamma_{\boldsymbol{i}} = \gamma^{ones(\boldsymbol{i})}, \gamma < 1$$

which is the projection onto the affine space of

$$\mathbb{f}(\boldsymbol{\theta};\boldsymbol{G}) \equiv \mathbb{b}(\tau(\boldsymbol{\theta})) = \sum_{\boldsymbol{i}} \begin{bmatrix} \gamma_{\boldsymbol{i}} \boldsymbol{b}_{\boldsymbol{i}} \\ \gamma_{\boldsymbol{i}} \end{bmatrix} \cdot B^2_{\boldsymbol{i}}(\tau(\boldsymbol{\theta})), \qquad (11)$$

where $\boldsymbol{i}$ goes through $\mathscr{I}_2$ in the summands in both (10) and (11). $\boldsymbol{G}$ is the $3^d \times 3$ matrix of parameters of the model, in which each row i is $\boldsymbol{b}_{I_2^{-1}(i)}$.

In many applications, not only the position of the end-effector is of interest but also its orientation. The easiest way to also represent the orientation using the KB-Maps is to represent the kinematics of the unit vectors $\boldsymbol{e}_1$, $\boldsymbol{e}_2$ and $\boldsymbol{e}_3$ of the end-effector coordinate system separately in different KB-Maps. If $f : \mathbb{R}^d \to \mathbb{R}^{4\times 4}$ maps joint values to the transformation matrix associated to the end-effector, the complete Bézier representation is

$$f(\boldsymbol{\theta}) \equiv \mathbb{B}(\boldsymbol{\theta}) := \begin{bmatrix} \boldsymbol{e}_1(\tau(\boldsymbol{\theta})) & \boldsymbol{e}_2(\tau(\boldsymbol{\theta})) & \boldsymbol{e}_3(\tau(\boldsymbol{\theta})) & \boldsymbol{b}(\tau(\boldsymbol{\theta})) \\ 0 & 0 & 0 & 1 \end{bmatrix},$$

where $\mathbb{B} : \mathbb{R}^d \to \mathbb{R}^{4\times 4}$ is the composed KB-Map, and $\boldsymbol{e}_1(\boldsymbol{\theta})$, $\boldsymbol{e}_2(\boldsymbol{\theta})$ and $\boldsymbol{e}_3(\boldsymbol{\theta})$ denote the KB-Maps of the kinematics of unit vectors.

III. Learning

Let us define a square cost function for a training set $\{(\boldsymbol{\theta}^{j)}, \boldsymbol{p}^{j)})\}_{j=1,\cdots,m}$:

$$E(\boldsymbol{G}) = \sum_j E_j(\boldsymbol{G}) = \sum_j \|\boldsymbol{f}(\boldsymbol{\theta}^{j)};\boldsymbol{G}) - \boldsymbol{p}^{j)}\|^2. \qquad (12)$$

The minimization of $E(\cdot)$ can be used to fit $\boldsymbol{f}$ to the set of training points. We can highlight the linearity of $\boldsymbol{f}$ by rewriting (10)

$$\boldsymbol{f}(\boldsymbol{\theta}^{j)};\boldsymbol{G}) = \sum_{\boldsymbol{i}} \frac{\gamma_{\boldsymbol{i}} \cdot B^2_{\boldsymbol{i}}(\tau(\boldsymbol{\theta}^{j)}))}{\sum_{\boldsymbol{i}} \gamma_{\boldsymbol{i}} \cdot B^2_{\boldsymbol{i}}(\tau(\boldsymbol{\theta}^{j)}))} \cdot \boldsymbol{b}_{\boldsymbol{i}} = \qquad (13)$$
$$\sum_{\boldsymbol{i}} \frac{\gamma_{\boldsymbol{i}} \cdot B^2_{\boldsymbol{i}}(\tau(\boldsymbol{\theta}^{j)}))}{\gamma^{j)}} \cdot \boldsymbol{b}_{\boldsymbol{i}} = \qquad (14)$$
$$\sum_{\boldsymbol{i}} w^{j)}_{\boldsymbol{i}} \cdot \boldsymbol{b}_{\boldsymbol{i}}, \qquad (15)$$

where $\gamma^{j)} = \sum_{\boldsymbol{i}} \gamma_{\boldsymbol{i}} \cdot B^2_{\boldsymbol{i}}(\tau(\boldsymbol{\theta}^{j)}))$ and $w^{j)}_{\boldsymbol{i}} = \frac{\gamma_{\boldsymbol{i}} \cdot B^2_{\boldsymbol{i}}(\tau(\boldsymbol{\theta}^{j)}))}{\gamma^{j)}}$. The quantity $\gamma^{j)}$ is common for all summands in sample j, and can be computed only once. It corresponds to the homogeneous coordinate that must be associated to $\boldsymbol{p}^{j)}$ to belong to the surface in projective space (11), hence the notation. Clearly, the selection of the best fitting parameters $\hat{\boldsymbol{G}}$ by means of the minimization of $E(\cdot)$ is a *linear* least squares problem:

$$\hat{\boldsymbol{G}} := \underset{\boldsymbol{G}}{\operatorname{argmin}}\ E(\boldsymbol{G}) = \sum_j \| \left(\sum_{\boldsymbol{i}} w^{j)}_{\boldsymbol{i}} \cdot \boldsymbol{b}_{\boldsymbol{i}}\right) - \boldsymbol{p}^{j)}\|^2. \qquad (16)$$

We can use two kinds of methods to solve this problem: exact methods and gradient methods.

Both are able to cope with irregular distributions of data in the training set, in contrast to some models like the original PSOM's that require a grid arrangement of the data. Besides, the gradient methods are naturally suited to deal with non-stationary data, a feature that is not available to PSOM's or even to PSOM+ [8]. And since the cost function is purely quadratic, it does so without risk of failing, because there is only one global minimum.

A. Exact methods

The linear system being fitted in the least squares sense by (16) is:

$$\boldsymbol{W} \cdot \boldsymbol{G} = \boldsymbol{P}, \tag{17}$$

where $\boldsymbol{W}$ is a $m \times 3^d$ matrix composed of columns $\boldsymbol{w}^{j)} = (w^{j)}_{I_2^{-1}(1)}, \dots, w^{j)}_{I_2^{-1}(3^d)})$ and $\boldsymbol{P}$ is an $m \times 3$ matrix in which row j is $\boldsymbol{p}^{j)}$. This system has enough data to determine a solution for $\boldsymbol{G}$ if $m \geq 3^d$. In this case, the linear least squares problem has a unique solution (if the columns of $\boldsymbol{W}$ are linearly independent) obtained by solving the normal equation:

$$(\boldsymbol{W}^t \boldsymbol{W}) \cdot \hat{\boldsymbol{G}} = \boldsymbol{W}^t \cdot \boldsymbol{P}. \tag{18}$$

$\hat{\boldsymbol{G}}$ can be determined by some standard method, such as QR-decomposition. If the data $\{(\boldsymbol{\theta}^{j)}, \boldsymbol{p}^{j)})\}_{j=1,\cdots,m}$ comes from noise-free FK, because any FK of d degrees of freedom can be expressed with $\boldsymbol{f}(\boldsymbol{\theta}; \boldsymbol{G})$, equation (17) will be satisfied exactly, i.e, $E(\hat{\boldsymbol{G}}) = 0$. Since the solution is unique, $\boldsymbol{f}(\boldsymbol{\theta}; \hat{\boldsymbol{G}})$ is the only FK function satisfying the data and, thus, the one that generated them. Consequently, generalization (both interpolation and extrapolation) will be perfect.

Of course, this happens in the absence of noise, but as it will be shown in the experimental Section IV, even with noisy data, we need a low number of samples to get a good approximation of the underlying FK.

In case there is no possibility to acquire enough data, i.e. the system of linear equations is underdetermined, it is still possible to find the solution that lies closest to an *a priori* estimate of the model (e.g. as a result of simulations). This can be done using, for instance, the Moore-Penrose pseudo inverse [12]. Finally, these exact learning techniques can be used repeatedly when some new data are acquired to generate successively improved models. Optionally, old data could be discarded when new ones are acquired, leading to an adaptive model.

B. Gradient methods

The derivative of $E_j(\boldsymbol{G})$ with respect to $\boldsymbol{b_i}$ (a row of $\boldsymbol{G}$) is easily obtained:

$$\frac{\partial E_j}{\partial \boldsymbol{b_i}} = (\boldsymbol{f}(\boldsymbol{\theta}^{j)}) - \boldsymbol{p}^{j)})\; w^{j)}_{\boldsymbol{i}}. \tag{19}$$

This permits the application of an on-line implementation of linear regression, by updating each $\boldsymbol{b_i}$ after the presentation of a new sample $(\boldsymbol{\theta}^{j)}, \boldsymbol{p}^{j)})$:

$$\boldsymbol{b_i} \leftarrow \boldsymbol{b_i} - \mu(\boldsymbol{f}(\boldsymbol{\theta}^{j)}) - \boldsymbol{p}^{j)})\; w^{j)}_{\boldsymbol{i}}, \tag{20}$$

where μ is the learning rate parameter. This update rule has been called Widrow-Hoff rule [Widrow & Hoff, 1960], delta rule, or LMS (Least Mean Squares) algorithm. Its application minimizes the mean squared error of the linear fit. It is a common practice to set $\mu = \mu_0 / ||\boldsymbol{w}^{j)}||^2$, $0 < \mu_0 \leqslant 1$, variation denoted as Normalized LMS.

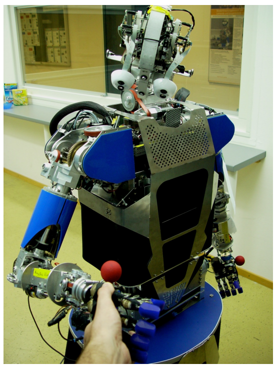
Learning by gradient methods is notoriously slower than with exact methods if a high precision is required. However, it has some advantages. The more important one is that, computationally, it is considerably lighter than exact methods. Besides, it quickly responds to dynamically changing conditions, such as easily deformable systems or the application of different tools. In general, it is naturally suited to approximate a non-stationary function.

IV. Experiments

In this section, the KB-Maps presented earlier in this work are evaluated on the humanoid platform ARMAR-IIIa [1] (see Fig. 3(a)), both in experiments on the real hardware and in simulation. The ARMAR-IIIa robot contains seven independent degrees of freedom (DoF) in each arm, one in the hip and three in the head. Each arm contains a 6 DoF force sensor in its wrist. The number of joints actively used during the experiments varied, as the complexity of the learning process grows exponentially with this number. This is the reason why a smaller number was used in the experiments on real hardware than in the simulations. As our approach aims at hand-eye coordination, all experiments include joints of both the head and one arm. This way, the camera could always point in the direction of the end-effector during the experiments. On the real robot, samples were generated by manually moving the robot arm via zero-force control (see Fig. 3(a)), while an estimated FK model obtained from the geometrical model was used to fix the head looking at the hand. Joint values were then read directly from the motor encoders in order to deal with a realistic amount of sensor noise. An optical marker (a red ball signaling the end of a tool) attached to the end-effector was tracked by the built-in stereo camera system (see Fig. 3(b)), and all training samples obtained had a distance of at least 1° and maximal 3° in parameter space to their predecessors. In simulation, joint values were generated randomly in parameter space –either normally distributed or sampled through a random walk. Artificial noise was added to the positions of the end-effector in some experiments.

A. Exact Method

In the first place, simulations that show the performance of the exact learning algorithm with six DoF are presented. For training and test, two sets with 13.000 and 6.000 training experiences, respectively, and with joint angles uniformly distributed over $\pm 80^\circ$ were used. The associated positions in euclidean space were created by the FK constructed from the CAD description of the kinematics. In addition to that, an artificial noise with standard deviation $\sigma_{noise} = 10$mm was applied only to the training data. For the evaluation of the extrapolation capabilities other 6.000 independent samples from a space more than ten times larger (with angles between $\pm 120^\circ$) were created. In the first experiment, subsets of the training data with different sizes were used


(a) Generation of training samples using zero-force control.

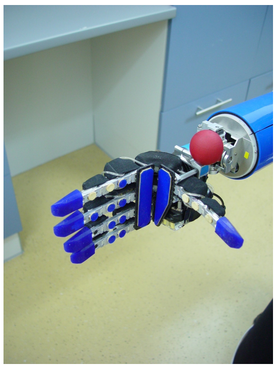

(b) Close-up of the optical marker attached to the right hand.

Fig. 3. The humanoid ARMAR-IIIa robot.

for learning. It is investigated how the error over the test, training and extrapolation data is related to the number of training experiences used for learning. In Fig. 4, the results of this experiment can be seen. The error on the training data (green) increases until it reaches the level of the artificial noise (grey), while the errors on the test data (blue) and the extrapolation (orange) decrease with a growing number of training data. After around 2.200 samples, the mean error on the unknown test data falls below the standard deviation of the artificial noise. Remember that the test set (unlike the training set) comes without noise, which explains why the test error becomes smaller than the training error. This means that the algorithm is capable of compensating for the sensor noise.

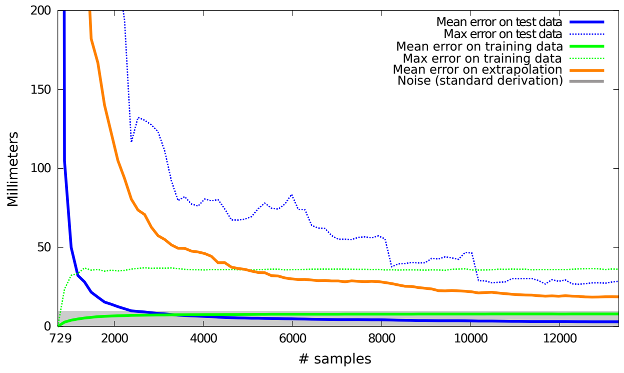


Fig. 4. Plot of a batch-learning with simulated data on the Armar-IIIa. Training samples (13.000) were generated equally distributed over $[\pm 80°]$ and with added noise of $\sigma_{err} = 10\,mm$ applied to the position $\boldsymbol{x}$ of the end-effector. A similarly generated test set with 6.000 samples and another set for extrapolation (equally distributed over $[\pm 120°]$) with 6.000 samples are included in this experiment. The figure shows the error over the learned data *(green)*, the test data *(blue)* and the extrapolation data *(orange)* in relation to a varying number of learned samples *without* online learning. Thick lines represent mean errors and dotted lines maximal errors.

Subsequently, the applicability on the real robot was examined. For this task, training experiences with five joints of the robot actively moved were produced as described in the beginning of this section. A training set with 1.000 samples and a training set with 500 samples were generated. In order to have a direct comparison, a second KB-Map was trained using exactly the same joint angles, but with the associated CAD-generated positions with an added noise of $\sigma_{noise} = 20mm$, which is approximately of the same magnitude as the one in the perception system. Fig. 5 shows the outcome of this experiment. As one can see from the similarity of both curves, the algorithm acted on real hardware as it had been predicted by the simulation.

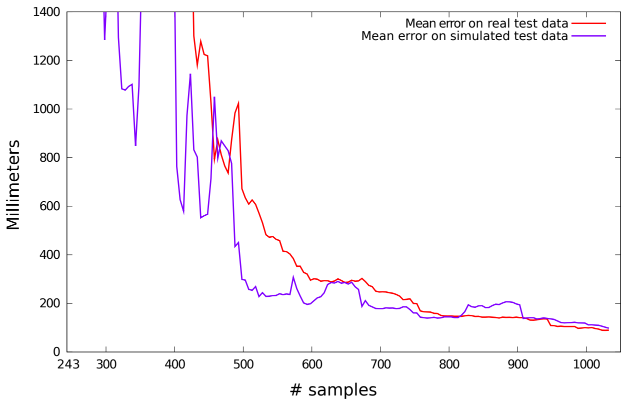


Fig. 5. Exact learning on real data recorded with 5DOF and a training set of 1000 samples and test set with 500 samples. The measured errors *(violet)* are compared to simulated values with noise $\sigma_{err} = 20\,mm$ *(red)*.

These two experiments show that the exact learning method is robust to sensor noise, and it can produce an acceptable estimation even for extrapolation points if enough training samples are provided.

B. *Gradient method*

In this section, the presented gradient method is integrated in a learning process that is divided into an online and an offline part. The order in which training samples are learnt is very important in the case of the gradient method. The best learning effect results from randomly generated data where consecutive samples have a larger distance in parameter space. In reality, however, this is not the case and samples that belong to the same trajectory will usually lie close to one another. This is why, in the first part of the learning process, points are learned online as they are generated. After a certain number of experiences have been acquired, these samples are randomly permuted and again learned by the net in the second stage. In this way, the accuracy of the net can be improved without the need to create new data.

As in the previous subsection, experiments were first performed on a simulated robot again. Joint values were created by a random walk in parameter space with a distance of at least 1° and a maximum of 3° between each angle vector. All joints values are normally distributed with $\sigma = 22°$ in order to create realistic trajectories. The robot now uses 7 active DoF and instead of learning the FK from scratch, this experiment simulates learning the application of a tool. Therefore, the initial KB-Map is an exact representation of the FK obtained from the CAD model. Then the training and test data were produced with a modified FK where the

TCP was moved by a distance of 250mm. In a variation of the experiment, artificial noise of $\sigma_{noise} = 20$ mm was added to the shifted TCP. The results are presented in Fig. 6. The light blue lines indicate the distance between the TCP taught positions in two consecutive learning iterations and increases as soon as the training data is permuted. As one can see, the mean error of the test data (red) and the data with artificial noise (orange) both drop very quickly. After 1.800 cycles the mean errors are about 50 mm. This shows the speed of this learning technique as well as the robustness to noise.

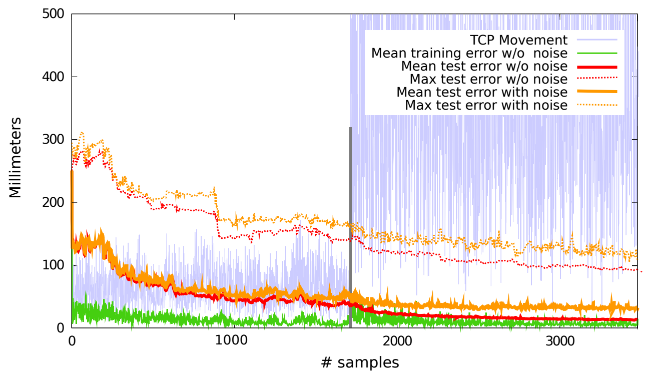


Fig. 6. Diagram of the learning progress of the incremental learning while learning a 7 DoF kinematic chain. It shows the actual error between estimation and experienced position in each iteration *(green)*, the error on the test data of learning without noise *(red)* and noise with $\sigma = 20mm$ *(orange)*. Thick lines represent mean errors and dotted lines maximal errors. The distance between the particular TCP positions is shown *(light blue)*. It increases dramatically after 1700 iterations, when the algorithm enters the second stage and the acquired experience is learnt again in random order.

In the last experiment, the learning behavior of the gradient method with simulated data and on real hardware is directly compared. The robot uses 6 active joints, and a number of 2.200 training experiences were created by moving the end-effector as described earlier in this section. The same joint values of these experiences were used in simulation to generate training samples. The outcome of the comparison of the two KB-Maps can be seen in Fig. 7. From the similarity of the two curves, it follows that the learning on real hardware succeeds as predicted by the simulation.

As a consequence from these two experiments, it can be seen that the gradient-based learning can be used to refine a crude FK model very rapidly. Thus, the obtained results proved the robustness of this learning to noise, as well as its applicability to a real humanoid robot.

V. Conclusions

In the present paper, a novel approach for learning the FK mapping as part of the body schema of humanoid robots was presented. Inspired by PSOMs, we wanted to overcome the large number of robot movements required to get a good approximation of FK.

First, since FK is a composition of circles, models based on polynomials (as PSOM) cannot exactly represent a FK. Thus, we have chosen a model based on rational Bézier polynomials –the Kinematic Bézier Maps–, which are a family of functions that includes the description of any FK. Besides, these functions have an important advantage: adjusting the model to a set of a sample points is a linear least squares problem.

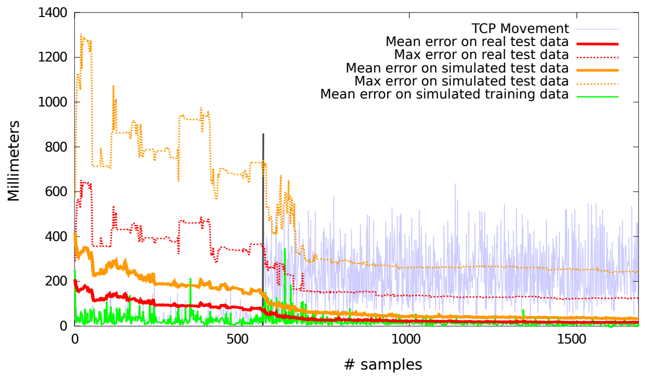


Fig. 7. Comparison of the incremental learning progress when processing real data from marker tracking (orange) and simulation w/o noise (red). Thick lines represent mean errors and dotted lines maximal errors. After 600 training experiences no new data is acquired but previously learnt data is processed anew in arbitrary order. The difference between the TCP positions is shown (light blue).

Second, we have introduced *a priori* knowledge of the function to be learnt in the model which is the key to reducing the number of samples. This has been achieved by restricting the model to represent only compositions of a certain family of ellipses which always includes the circle. The constraints implied by this restriction are easily integrated in the linear least square problem. The approach can be summarized as reformulating the problem in a larger space –the positions of the Bézier control points–, where it becomes linearly solvable.

This higher-dimensional problem can be easily solved with any standard linear least-squares method, yielding our exact learning method. Alternatively, the least squares cost has a simple derivative, encouraging alternative algorithms, the so-called gradient learning methods, which are well suited for online-learning. Using the exact method, in the absence of noise, it is possible to learn *exactly* a FK with only 3^d samples, where d is the number of DoF, which none of the previous works was able to accomplish. And so, with an arbitrary sample distribution. This means that, even if samples are grouped in a very reduced zone of the workspace, interpolation and extrapolation are perfect.

We have carried out experiments, both simulated and in real hardware, with a humanoid robot under noisy conditions, proving that our algorithms are able to quickly learn a good approximation of the kinematics of the robot from inaccurate measures.

Our learning algorithm performs very well if enough noisy samples from the whole workspace are provided. Even if the noisy samples are restricted to a local zone of the workspace, we obtain good interpolation and extrapolation, although the last one requires more samples. But, if the samples are noisy, few and local, the algorithm performs poorly, especially in extrapolation, where it can exhibit very large errors. This is due to the fact that with noise and scarce data, the isoparametric curves of the model become often strongly elliptical.

This provides an idea about how to improve our system in these conditions, although there do not exist any easy solution because the constraints to enforce complete circularity are non-linear. Finally, a less challenging future work is to deal not only with rotational joints, but to generalize the model for robots having any combination of prismatic and rotational joints.

Appendix

A. Isoparametric Curves of the Multidimensional Model

A d-dimensional tensor product Bézier form of degree 2 in which the vector $\boldsymbol{i}$ is spelled out for convenience, has the form:

$$\mathbb{b}(s_1,\ldots,s_d) = \sum_{i_1,\ldots,i_d=0}^{2} \mathbb{b}_{i_1,\ldots,i_d} \cdot B^2_{i_1,\ldots,i_d}(s_1,\ldots,s_d). \quad (21)$$

Without loss of generality, we show the isoparametric curve of this Bézier form when s_1 is the free variable. The above equation can be rewritten as:

$$\sum_{k=0}^{2} B^2_k(s_1)\Big(\sum_{i_2,\ldots,i_d=0}^{2} B^2_{i_2,\ldots,i_d}(s_2,\ldots,s_d)\cdot \mathbb{b}_{k,i_2,\ldots,i_d}\Big). \quad (22)$$

We can define a new function $\mathbb{q}_k(s_2,\ldots,s_d)$ to rename the expression in the big parenthesis; when $s_2,\ldots,s_d$ are fixed, $\mathbb{q}_k$ is a constant and (22) becomes a single-variable Bézier curve defined by the control points $\mathbb{q}_0$, $\mathbb{q}_1$ and $\mathbb{q}_2$:

$$\sum_{k=0}^{2} B^2_k(s_1)\cdot \mathbb{q}_k(s_2,\ldots,s_d). \quad (23)$$

Let the homogeneous coordinates of $\mathbb{q}_0$, $\mathbb{q}_1$ and $\mathbb{q}_2$ be ϖ_0, ϖ_1 and ϖ_2, respectively. To be an ellipse, ϖ_0=ϖ_2 and $\varpi_1/\varpi_0 < 1$ must be satisfied. Remind that we set the weights $\gamma_{i_1,i_2,\ldots,i_d}$ of control points $\mathbb{b}_{i_1,\ldots,i_d}$ to $\gamma^{ones(i_1,\ldots,i_d)}$, where $ones()$ returns the number of ones in the arguments and γ is an arbitrary constant minor than one.

The values of the ϖ's are then

$$\varpi_0 = \sum_{i_2,\ldots,i_d=0}^{2} B^2_{i_2,\ldots,i_d}(s_2,\ldots,s_d)\cdot\gamma_{0,i_2,\ldots,i_d}$$

$$\varpi_1 = \sum_{i_2,\ldots,i_d=0}^{2} B^2_{i_2,\ldots,i_d}(s_2,\ldots,s_d)\cdot\gamma_{1,i_2,\ldots,i_d}$$

$$\varpi_2 = \sum_{i_2,\ldots,i_d=0}^{2} B^2_{i_2,\ldots,i_d}(s_2,\ldots,s_d)\cdot\gamma_{2,i_2,\ldots,i_d}$$

Everything in the development of ϖ_0 is the same as that in ϖ_2, except the first index in the weights, which is 0 for ϖ_0 and 2 for ϖ_2. Since $\gamma_{0,i_2,\ldots,i_d} = \gamma^{ones(i_2,\ldots,i_d)}$ and $\gamma_{2,i_2,\ldots,i_d} = \gamma^{ones(i_2,\ldots,i_d)}$, we conclude that ϖ_0=ϖ_2. Similarly, ϖ_0 and ϖ_1 differ only in the first index of all involved weights. Those in ϖ_1 are $\gamma_{1,i_2,\ldots,i_d} = \gamma^{ones(i_2,\ldots,i_d)+1}$, which means that they correspond to those involved in ϖ_0 multiplied by γ. Therefore, the conditions $\varpi_1 = \varpi_0\gamma$ and $\varpi_1/\varpi_0 = \gamma < 1$ are met, which concludes the proof that, with the chosen weights for control points $\mathbb{b}_{i_1,\ldots,i_d}$, the isoparametric curves of (21) are ellipses.

References

[1] T. Asfour, K. Regenstein, P. Azad, J. Schroder, A. Bierbaum, N. Vahrenkamp, and R. Dillmann. Armar-iii: An integrated humanoid platform for sensory-motor control. pages 169–175, Dec. 2006.

[2] V. R. de Angulo and C. Torras. Speeding up the learning of robot kinematics through function decomposition. *IEEE Transactions on Neural Networks*, 16(6):1504–1512, 2005.

[3] 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*, 38(6):1571–1577, 2008.

[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] G. E. Farin. *NURBS: From Projective Geometry to Practical Use*. A. K. Peters, Ltd., Natick, MA, USA, 1999.

[6] G. E. Farin. *Curves and surfaces for CAGD: a practical guide*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 2002.

[7] M. Hersch, E. Sauser, and A. Billard. Online learning of the body schema. *International Journal of Humanoid Robotics*, 5(2):161–181, 2008.

[8] S. Klanke and H. J. Ritter. Psom+ : Parametrized self-organizing maps for noisy and incomplete data. In *Proceedings of the 5th Workshop on Self-Organizing Maps (WSOM 05)*, Paris, France, Sep 2005.

[9] A. Maravita and A. Iriki. Tools for the body (schema). *Trends in Cognitive Sciences*, 8(2):79 – 86, 2004.

[10] A. Maravita, C. Spence, and J. Driver. Multisensory integration and the body schema: Close to hand and within reach. *Current Biology*, 13:531–539, 2003.

[11] W. Penfield and T. Rasmussen. *The Cerebral Cortex of Man: A Clinical Study of Localization of Function*. Macmillan, 1950.

[12] R. Penrose. A generalized inverse for matrices. In *The Cambridge Philosophical Society*, 51, pages 406–413, 1955.

[13] H. Prautzsch, W. Boehm, and M. Paluszny. *Bezier and B-Spline Techniques*. Springer-Press New York, Inc., Secaucus, NJ, USA, 2002.

[14] M. I. Stamenov. Body schema, body image, and mirror neurons. In H. D. Preester and V. Knockaert, editors, *Body Image and Body Schema*, pages 21–43. De Preester, 2005.

[15] J. A. Walter. PSOM network: Learning with few examples. In *In Proc. Int. Conf. on Robotics and Automation (ICRA-98*, pages 2054–2059, 1998.

[16] D. Whitney. Resolved motion rate control of manipulators and human prostheses. *Man-Machine Systems, IEEE Transactions on*, 10(2):47–53, June 1969.