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: [stefan.ulbrich,tamim.asfour,dillmann]@kit.edu
† Institut de Robòtica i Informàtica Industrial,
CSIC-UPC, Barcelona
Email: [ruiz,torras]@iri.upc.edu

***Abstract*— The kinematics of a robot with many degrees of freedom is a very complex function. Learning this function for a large workspace with a good precision requires a huge number of training samples, i.e. robot movements. In this work, we introduce the Kinematic Bézier Map (KB-Map), a parametrizable model without the generality of other systems, but whose structure readily incorporates some of the geometric constraints of a kinematic function. In this way, the number of training samples required is drastically reduced. Moreover, the simplicity of the model reduces learning to solving a linear least squares problem. Systematic experiments have been carried out showing the excellent interpolation and extrapolation capabilities of KB-Maps and their low sensitivity to noise.**

***Index Terms*— Learning, robot kinematics, humanoid robots.**

I. Introduction

With increasingly complex robots—especially humanoids—the calibration process of the arms and other kinematic chains becomes a difficult, time-consuming and often expensive task. This process has to be repeated every time the robot accidentally 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. Nevertheless, the accuracy of kinematics is important in many prominent robotic problems [2]. Humans solve the problem successfully by pure self-observation, which has led to the adaptation of biologically-inspired mechanisms to the field of robotics.

Following this trend, a rather novel approach to deal with the kinematic problem is *learning*. In order to get training samples, the end-effector cartesian coordinates associated to a given joint value vector must be obtained using some kind of sensorial system. Learning can provide approximate solutions when the kinematic functions are difficult or slow to compute. It is also the only way of having accurate solutions when there are uncertainties in the kinematic parameters.

A very preliminary version of this work was presented at Humanoids-2009 [1].

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

V. Ruiz de Angulo and C. Torras 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.

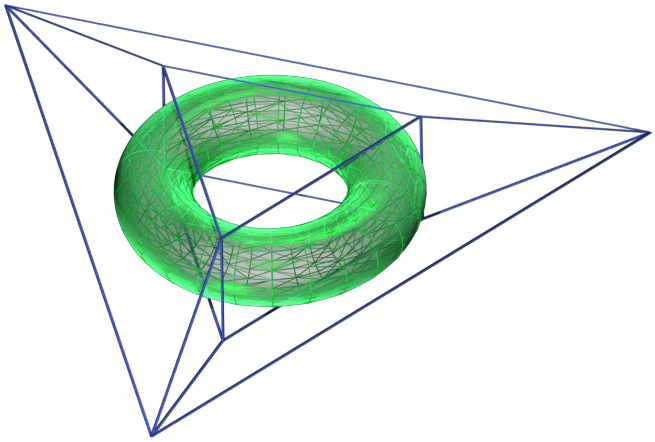


Fig. 1. The workspace of a simple 2-DoF robot with orthogonal axes is the surface of a torus. A KB-Map can learn an exact parametrization of such a workspace in the absence of noise. Here, the resulting manifold (green) and the control net (blue) that contains all learned information are shown.

Another important advantage of the learning approach is adaptability. For an industrial robot (especially high-precision ones), this means self-calibration while working. For autonomous robots it is even more interesting, since they are more prone to offsets in sensor readings, geometric changes due to wear and tear, deformations, tool usage, etc. The robot should be able to cope with these problems without human intervention.

Each set of values for the robot joints determines a unique end-effector pose (position and orientation). This is a physical realization of the Forward Kinematics (FK) mapping from joint angle values $\boldsymbol{\theta} \in \mathbb{R}^d$, d being the number of robot degrees of freedom (DoF), to coordinates $\boldsymbol{x}$ in the cartesian workspace. The Inverse Kinematics (IK) mapping, from $\boldsymbol{x}$ to $\boldsymbol{\theta}$, is difficult to handle because of two reasons. First, it is a one-to-many mapping. Second, obtaining an IK solution is computationally much more demanding than finding a FK one. Moreover, analytical or geometrical solutions are not known for manipulators with many redundant degrees of freedom.

The learning of the relation between the joint coordinates $\boldsymbol{\theta}$ and the pose $\boldsymbol{x}$ of the end-effector can be approached in three

ways:

1) $\boldsymbol{x} \rightarrow \boldsymbol{\theta}$, that is, direct learning of IK. When confronted with different outputs for a same input, most learning systems resolve the uncertainty by averaging the output. Unfortunately, the average of IK solutions is not an IK solution in general. Thus, with this strategy one can only learn the IK of non-redundant systems [3], [4]. Even for robots not commonly considered as redundant, because only a finite number of solutions exists (like the PUMA), the joint space must be restricted so that only one solution is possible.
2) $(\Delta\boldsymbol{x}, \boldsymbol{\theta}) \rightarrow \Delta\boldsymbol{\theta}$. The problem above can be avoided with a different input-output representation. Instead of mapping directly end-effector poses to joint values, one learns how to modify slightly $\boldsymbol{x}$ by means of small movements of the joints. When these movements are made in the vicinity of a given $\boldsymbol{\theta}$, the average is truly a IK solution. Therefore, incorporating $\boldsymbol{\theta}$ to the input allows valid localized solutions [5], [6]. It is also possible to bias the movements of the robot toward configurations satisfying a constraint, so that it becomes incorporated in the learned mapping. To carry out a complete movement, a number of intermediate points must be calculated, close enough to allow the system to provide good approximations for the gaps between them. Anyway, to avoid the progressive accumulation of errors, this kind of strategy may require visual feedback.
3) $\boldsymbol{\theta} \rightarrow \boldsymbol{x}$. This strategy consists in learning the well defined function FK. The learned forward model can then be processed in several ways to obtain IK information. Iterative procedures for solving a system of non-linear equations are usual, in particular Newton's methods based on successive linear interpolation of FK equations or model [7]. For redundant robots, an extra optimization term can be locally minimized at each step [8], [9]. This term can be changed during run-time *without further learning*, therefore being a much more flexible strategy than the preceding one. There are neural architectures conceived to solve this constrained minimization at a fast convergence rate [10]. Another possibility is to calculate small steps $\Delta\boldsymbol{x}$ (that can be accurately calculated in a single step by the same techniques) and generate incrementally a reaching trajectory [11]. Still another option is to minimize a cost function whose minimum is the IK solution, as made by Parametrized Self-Organizing Maps (PSOMs) [12] and its extension, PSOM+ [13]. This cost can also include other optimization criteria. All these approaches require the calculation of the Jacobian of the FK, although there exist also techniques avoiding this step [14].

The current work follows the third approach, 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 influencing the result. Hence, it requires a large amount of training experiences that grows exponentially with the number of DoF of the kinematic chain. To get a feeling of the problem, imagine one has obtained 3 samples for each DoF of a PUMA in a regular grid, thus $3^6 = 729$ samples in total. The best that a usual interpolator can do when all inputs but one are fixed is to behave like a quadratic function—what PSOMs really do. But a quadratic polynomial can only approximate the true function with good precision in very narrow ranges. To cope with the complete joint workspace of the PUMA, therefore, many more samples for dimension are needed, which makes the total number of sample points explode.

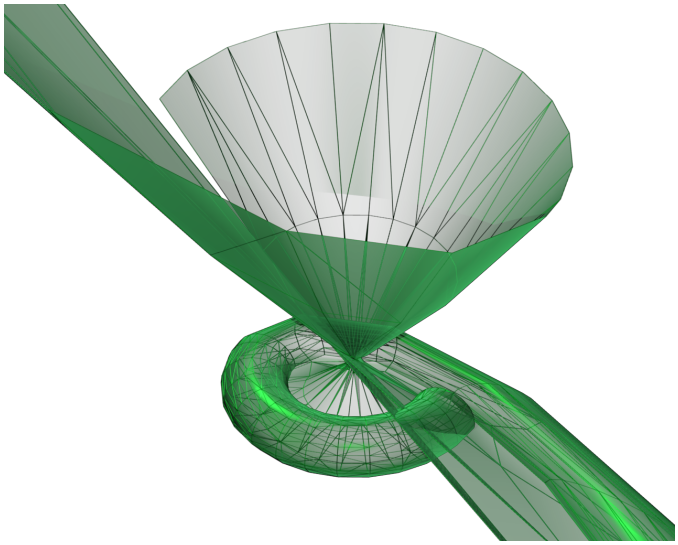


Fig. 2. Plot of workspace similar to that in Fig. 1 but learned by a PSOM. The underlying lattice has 5×5 knots (polynomial degree 5) and all angles were chosen equidistantly between 0° and 160°, thus sampling nearly a quarter of the torus' surface, 25 configurations in total. Good interpolation can be clearly recognised on the lower left.

An approach that attacks directly this "curse of dimensionality" is to decompose the kinematics function into lower dimensionality functions, requiring a number of samples orders of magnitude lower than raw kinematics. However, one previously proposed decomposition is restricted to certain types of manipulators [15] and another requires a complex learning architecture [16] difficult to manage and also a more complex sensorial set-up.

An alternative way to reduce the number of required samples without reducing the size of the workspace or the versatility of redundant systems is possible: the introduction of a priori knowledge of the function to be learned. The only work following this line was recently proposed by Hersch et al. [17]. The parameters of the FK in Denavit-Hartenberg convention are learned directly by an optimization algorithm. This optimization eventually leads to a kinematic mapping 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 does not exist yet an algorithm that can learn a FK mapping *exactly* and in an efficient way. We present a learning model for the $\boldsymbol{\theta} \rightarrow \boldsymbol{x}$ mapping (i.e., FK) that incorporates a lot of a priori knowledge

embedded into the model. This allows to interpolate and even extrapolate with zero error using only 3^d samples in the absence of noise, which none of the previous works is able to accomplish. But, at the same time, our model encompasses a family of functions wider than that of FK, which can be useful to approximate for example FK's deformed by gravity. The Jacobian of this forward model can be efficiently obtained. Our approach is based on 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. This representation permits an exact encoding of the FK, which is robust to sensor noise, and 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.

In 1 and Fig. 2, some of the advantages of the new approach are illustrated. A PSOM and a KB-Map were trained with 25 points of a 2-DoF mechanism (a torus workspace) lying in a range of $[0°, 160°]$ in each joint angle. The PSOM interpolates very well in the trained region, but extrapolates badly. The KB-Map *exactly* estimates the workspace shape *inside and outside the training region* and, as a matter of fact, $3^2 = 9$ points would suffice to obtain the same result.

This paper is structured as follows. In the next section, a brief introduction to the underlying geometrical techniques is provided, followed by a description of their application in the KB-Maps to encode FK. Two algorithms suitable to perform the learning are presented in Section III. Then, in Section IV, the proposed method is applied to two simulated robot arms and to the humanoid robot ARMAR-IIIa [18], 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 [19] [20]:

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

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. These combinations are convex so that the curve lies within the convex hull formed by the control points for $s \in [0, 1]$. At $s = 0$ and $s = 1$ the curve coincides with the first and the last control point $\boldsymbol{b}_0$ and $\boldsymbol{b}_n$, respectively. The Bézier form of the curve's derivative

$$\dot{\boldsymbol{b}}(s) = n \cdot \sum_{i=0}^{n-1} \Delta\boldsymbol{b}_i \cdot B_i^{n-1}(s) \tag{2}$$

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

$$\boldsymbol{b}(s_1, s_2) = \sum_{i_1=0}^{n} \cdot \left(\sum_{i_2=0}^{n} \boldsymbol{b}_{i_1,i_2} \cdot B_{i_2}^n(s_2) \right) \cdot B_{i_1}^n(s_1). \tag{3}$$

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

$$\boldsymbol{b}(\boldsymbol{s}) = \sum_{\boldsymbol{i}} \boldsymbol{b}_{\boldsymbol{i}} \cdot B_{\boldsymbol{i}}^n(\boldsymbol{s}), \tag{4}$$

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

$$B_{\boldsymbol{i}}^n(\boldsymbol{s}) := \prod_{k=1}^{d} B_{i_k}^n(s_k) \tag{5}$$

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 the FK can be approximated by polynomials, an exact representation of the FK requires a more complex class of functions, e.g. *rational polynomials* [21]. In this section, a brief introduction to rational polynomials, projective geometry and the rational Bézier form will be presented, while [20], [21] provide more detailed information. Rational polynomial functions are similar to affine polynomial functions except for the fact that they are defined in the *projective space* $\mathcal{P}$. Simplifying, $\mathcal{P}$ is a space with an additional dimension and elements of the form

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

where $\boldsymbol{p}$ is an affine point and γ is called *homogeneous coordinate* or *weight* of $\mathbb{p}$. Any projective point $\mathbb{p} \in \mathcal{P}$ can be understood as a ray that originates from the *projective center* $\big(0, \ldots, 0\big)^T$ 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 (into the affine space)*.

On projection into the affine space, rational polynomials generally become more complex functions and may loose their polynomial characteristics (see Fig. 3). Still, in projective

space, there does exist the same previously introduced unique Bézier form for curves and surfaces

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

and, after affine projection, the rational Bézier form

$$\boldsymbol{b}(\boldsymbol{s}) = \frac{\gamma(\boldsymbol{s})\boldsymbol{b}(\boldsymbol{s})}{\gamma(\boldsymbol{s})} = \frac{\sum_{\boldsymbol{i}} \gamma_{\boldsymbol{i}} \cdot \boldsymbol{b}_{\boldsymbol{i}} \cdot B_{\boldsymbol{i}}^{n}(\boldsymbol{s})}{\sum_{\boldsymbol{i}} \gamma_{\boldsymbol{i}} \cdot B_{\boldsymbol{i}}^{n}(\boldsymbol{s})}. \tag{6}$$

The greater their values, the closer the function approaches the corresponding control point. If one weight gets smaller than zero, then the curve does not lie in the convex hull of the control polygons anymore.

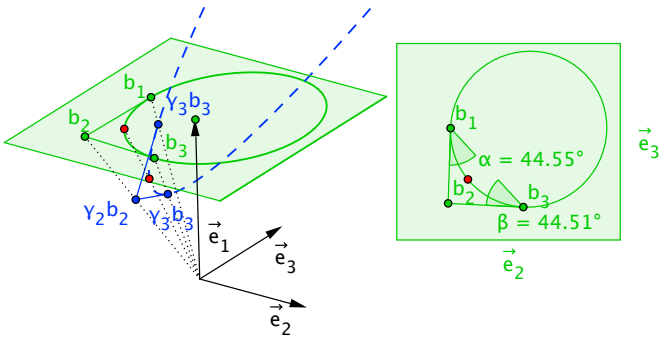


Fig. 3. The rational parameterization of a circle. On the left, the rational parabola *(blue)* with the weights γ_i and its affine image *(green)* are shown in $\mathcal{P}^2$. The projection is indicated by the dotted lines. On the right, the circle and the circle condition are shown in $\mathcal{A}^2$.

B. Forward Kinematics Representation: The One-dimensional Case

In this section, we show how to use the techniques presented above to define 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 the product space of d circles. Therefore, the basic geometric objects that we need to represent are circles and more generally their deformations. Currently, the deformation of circles that we focus on 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 [21]. In particular, a rational Bézier curve

$$\boldsymbol{b}(s) = \frac{\sum_{i=0}^{2} \gamma_i \cdot \boldsymbol{b}_i \cdot B_i^2(s)}{\sum_{i=0}^{2} \gamma_i \cdot B_i^2(s)} \tag{7}$$

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, it additionally 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 (7) 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, by doing this, the circle is not excluded from the family of ellipses potentially represented by the Bézier form. 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 come 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 $\boldsymbol{b}(s)$ 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

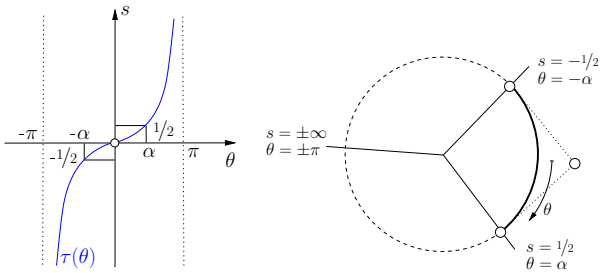
$$\tau : [-\pi, \pi] \mapsto \mathbb{R}, \quad \tau(\theta) = \frac{\tan(\theta/2)}{2 \cdot \tan(\alpha/2)}, \tag{8}$$

where $\alpha = \arccos(\gamma_1)$, see Fig. 4(a). In fact, it is more practical to fix indirectly γ_1 by first choosing an arbitrary angle α and setting $\gamma_1 = \cos(\alpha)$. The meaning of this transformation is that, when $\boldsymbol{b}(s)$ becomes exactly a circle, α becomes the common angle in the isosceles triangle formed by the control points, see Fig. 4(b). Appendix I proves that, in this case, $\boldsymbol{b}(\tau(\theta))$ becomes the angular parameterization of the circle measured in radian units, which is the final form of the one-dimensional KB-Maps.

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

$$ones() : \{0,1,2\}^d \to \mathbb{N}$$



(a) The τ transformation (b) The parameterization

Fig. 4. Transformation τ from a joint angle θ to the corresponding parameter of the Bézier form s.

returns the number of ones in the arguments and γ is an arbitrary constant minor than one. The proof is in Appendix II. The value γ can be selected like in the one-dimensional case, via the cosine of an arbitrary angle α, i.e. $\gamma = \cos\alpha$.

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 this case, the implicit control points (named $\mathfrak{q}_k$ in Appendix II) 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, $\boldsymbol{\theta} = (\theta_1, \ldots, \theta_d)^t$. The reason 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:

$$\boldsymbol{f}(\boldsymbol{\theta}; \boldsymbol{G}) \equiv \boldsymbol{b}(\tau(\boldsymbol{\theta})) = \frac{\sum_{\boldsymbol{i}} \gamma_{\boldsymbol{i}} \cdot \boldsymbol{b}_{\boldsymbol{i}} \cdot B^2_{\boldsymbol{i}}(\tau(\boldsymbol{\theta}))}{\sum_{\boldsymbol{i}} \gamma_{\boldsymbol{i}} \cdot B^2_{\boldsymbol{i}}(\tau(\boldsymbol{\theta}))}, \quad \gamma_{\boldsymbol{i}} = \gamma^{ones(\boldsymbol{i})}, \gamma < 1 \tag{9}$$

which is the projection onto the affine space of

$$\mathbb{f}(\boldsymbol{\theta}; \boldsymbol{G}) \equiv \mathbb{b}(\tau(\boldsymbol{\theta})) = \sum_{\boldsymbol{i}} \begin{bmatrix} \gamma_{\boldsymbol{i}} \boldsymbol{b}_{\boldsymbol{i}} \\ \gamma_{\boldsymbol{i}} \end{bmatrix} \cdot B^2_{\boldsymbol{i}}(\tau(\boldsymbol{\theta})), \tag{10}$$

where $\boldsymbol{i}$ goes through $\mathcal{I}_2$ in the summands in both (9) and (10). $\boldsymbol{G}$ is the $3^d \times 3$ matrix of parameters of the model, in which each row i is $\boldsymbol{b}_{I_2^{-1}(i)}$.

Computing the IK using this FK model by some kind of optimization requires the Jacobian of (9). In Appendix III we show how to calculate it very efficiently.

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 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 $f : \mathbb{R}^d \to \mathbb{R}^{4\times4}$ 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 $\mathbb{B} : \mathbb{R}^d \to \mathbb{R}^{4\times4}$ 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.

D. Forward Kinematics Representation: Bézier Splines

The presented Bézier representation still has a shortcoming. Evaluating the forward model at angles close to $\pm180°$ can lead to numerical instability (see Fig.4(b)). The convergence speed of the gradient method (see next section) was observed to be slower for angles in that region of the joint values.

One possibility to solve these problems is to use *Bézier splines*—curves that are piecewise in Bézier form—rather than a single Bézier curve. We represent the ellipses in the main directions with three Bézier curves, each of them used in a safe range. This alternative representation does not involve a larger number of parameters and it is completely compatible with the techniques shown in the next sections. Its construction is explained in detail in Appendix III. If not stated otherwise, the term KB-Map refers to the new spline representation during the rest of this work.

III. Learning Algorithms

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

$$E(\boldsymbol{G}) = \sum_j E_j(\boldsymbol{G}) = \sum_j \|\boldsymbol{f}(\boldsymbol{\theta}_j; \boldsymbol{G}) - \boldsymbol{p}_j\|^2. \tag{11}$$

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 (9)

$$\boldsymbol{f}(\boldsymbol{\theta}_j; \boldsymbol{G}) = \sum_{\boldsymbol{i}} \frac{\gamma_{\boldsymbol{i}} \cdot B^2_{\boldsymbol{i}}(\tau(\boldsymbol{\theta}_j))}{\sum_{\boldsymbol{i}} \gamma_{\boldsymbol{i}} \cdot B^2_{\boldsymbol{i}}(\tau(\boldsymbol{\theta}_j))} \cdot \boldsymbol{b}_{\boldsymbol{i}} = \tag{12}$$

$$\sum_{\boldsymbol{i}} \frac{\gamma_{\boldsymbol{i}} \cdot B^2_{\boldsymbol{i}}(\tau(\boldsymbol{\theta}_j))}{\hat{\gamma}_j} \cdot \boldsymbol{b}_{\boldsymbol{i}} = \tag{13}$$

$$\sum_{\boldsymbol{i}} w_{\boldsymbol{i},j} \cdot \boldsymbol{b}_{\boldsymbol{i}}, \tag{14}$$

where $\hat{\gamma}_j = \sum_{\boldsymbol{i}} \gamma_{\boldsymbol{i}} \cdot B^2_{\boldsymbol{i}}(\tau(\boldsymbol{\theta}_j))$ and $w_{\boldsymbol{i},j} = \frac{\gamma_{\boldsymbol{i}} \cdot B^2_{\boldsymbol{i}}(\tau(\boldsymbol{\theta}_j))}{\hat{\gamma}_j}$. The quantity $\hat{\gamma}_j$ is common for all summands in sample j, and it 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 (10), hence the notation. Clearly, the selection of the best fitting parameters $\boldsymbol{G}^*$ by means of the minimization of $E(\cdot)$ is a *linear* least squares problem:

$$\boldsymbol{G}^* := \underset{\boldsymbol{G}}{\operatorname{argmin}}\ E(\boldsymbol{G}) = \sum_j \| \left(\sum\nolimits_{\boldsymbol{i}} w_{\boldsymbol{i},j} \cdot \boldsymbol{b_i} \right) - \boldsymbol{p}_j \|^2. \quad (15)$$

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 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 or even to PSOM+ [13]. 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

In order to express the learning equations in matrix notation, we need to introduce a bijective function $I_n(\boldsymbol{i})$ that enumerates all possible index vectors $\boldsymbol{i}$ pertaining to $\mathcal{I}_2$ from 1 to $(n+1)^d = 3^d$. The linear system being fitted in the least squares sense by (15) is:

$$\boldsymbol{W} \cdot \boldsymbol{G} = \boldsymbol{P}, \quad (16)$$

where $\boldsymbol{W}$ is a $m \times 3^d$ matrix composed of columns

$$\boldsymbol{w}_j = (w_{I_2^{-1}(1),j}, \ldots, w_{I_2^{-1}(3^d),j}),$$

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:

$$(\boldsymbol{W}^T \boldsymbol{W}) \cdot \boldsymbol{G}^* = \boldsymbol{W}^T \cdot \boldsymbol{P}. \quad (17)$$

$\boldsymbol{G}^*$ 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 a noise-free FK equation (16), they will be fitted exactly, i.e, $E(\boldsymbol{G}^*) = 0$. This is because any FK of d degrees of freedom can be expressed with $\boldsymbol{f}(\boldsymbol{\theta}; \boldsymbol{G})$. Since the solution is unique, $\boldsymbol{f}(\boldsymbol{\theta}; \boldsymbol{G}^*)$ is the only FK function fitting 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 shown in the experimental Section IV, even with noisy data we only 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 [22]. 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 obtained in the following way:

$$\frac{\partial E_j(\boldsymbol{G})}{\partial \boldsymbol{b_i}} = (\boldsymbol{f}(\boldsymbol{\theta}_j; \boldsymbol{G}) - \boldsymbol{p}_j)\ w_{\boldsymbol{i},j}. \quad (18)$$

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 $(\boldsymbol{\theta}_j, \boldsymbol{p}_j)$:

$$\boldsymbol{b_i} \leftarrow \boldsymbol{b_i} - \mu(\boldsymbol{f}(\boldsymbol{\theta}_j; \boldsymbol{G}) - \boldsymbol{p}_j)\ w_{\boldsymbol{i},j}, \quad (19)$$

where μ is the learning rate parameter. This update rule has been called Widrow-Hoff rule [23], 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

$$\mu = \mu_0 / ||\boldsymbol{w}_j||^2, \quad 0 < \mu_0 \leqslant 1,$$

a variation denoted as Normalized LMS.

For linear cost functions, just as E_j, when $\mu_0 = 1$ the application of the learning rule reduces the cost function to zero (i.e. the sample is completely learned). Learning by gradient methods is notoriously slower than with exact methods if high precision is required. However, it has some advantages. The more important one is that, computationally, it is considerably lighter than exact methods, with respect to speed and memory. Besides, it responds very quickly 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

This section is divided into three parts. First, a low-dimensional case with two rotational DoF, i.e., a 2R mechanism will be considered. The learned manifold is a surface and can hence be easily visualized. The advantage of the presented algorithms and some basic observations will be discussed. Afterwards, the algorithms will be applied to higher-dimensional cases. Finally, the presented techniques will be used to create a model from the perceptions collected by a real humanoid robot.

A. 2R-Mechanism

The first experiments were performed with a very simple 2R-mechanism in simulation. In general, the constraint space (or workspace) is a biquadratic surface. It hence becomes possible to visualize this manifold in order to give an insight to its structure. In this example, the parameters of the kinematics were chosen in a way that the constraint manifold coincides with the surface of a torus ($a_1 = 100$ *mm*, $d_1 = 0$ *mm*, $\alpha_1 = 90°$, $a_2 = 50$ *mm*, $d_2 = 0$ *mm* and $\alpha_2 = 0°$).

This example begins with a PSOM being trained by regularly sampling a portion of the torus surface, see Fig. 2. The underlying lattice in the parameter space has 5×5 knots and, thus, the learned surface is a polynomial of degree 5. All angles were chosen equidistantly between $0°$ and $160°$. As a consequence, nearly a quarter of the torus' surface is sampled,

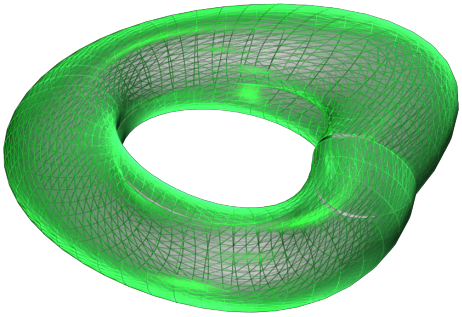


Fig. 5. PSOM surface learned from samples distributed over the whole torus-shaped workspace. As in Fig. 2, an underlying lattice with 5×5 knots is used. The training samples were degraded by artificial noise with $\sigma = 2^\circ$.

yielding 25 configurations in total. The very good interpolation of the samples can be clearly seen on the lower left of Fig. 2. In order to show the algorithm's extrapolation capabilities, the surface over the whole parameter space is shown in this picture. It diverges quickly from the torus' surface as soon as one of the angles leaves the area covered by the lattice (i.e. the area spanned by the samples).

If a KB-Map is trained under the same conditions, the learned constraint space coincides exactly with the torus. This is because there was no error in the training data and $3^2 = 9$ points suffice to define the surface. The manifold and the control net that carries all the information gained in the learning process are displayed in Fig. 1.

Next we investigate the impact of simulated noise on these results. To every angle θ_1 and θ_2 we add noise that is normally distributed around 0° with standard deviation $\sigma = 2^\circ$. Since, even without noise, extrapolation is unacceptable for the PSOM, in Fig. 5 we show the results of training it with learning samples coming from the whole workspace. The interpolation passes necessarily through all erroneous data which results in a distortion of the surface. This drawback of the PSOM algorithm, however, is solved in the algorithm's extension, the *PSOM+* [13]. The effect is much less drastic as the points the surface interpolates are determined by a metric that regulates the curvature. This also removes the restriction that bounds the training data to be on the knots of the underlying lattice. The KB-Map reacts differently to the noise (Fig. 6). On the one hand erroneous data are not interpolated exactly as long as more than the minimum amount of samples is provided (in order to find the least mean squares solution). Moreover, the curvature is limited due to the biquadratic nature of the surface much like that of the PSOM+. On the other hand, extrapolation accuracy decreases largely but, in contrast to PSOM, the curve always lies on ellipses in the main directions. As a consequence, the extrapolation will always resemble a distorted torus. In Fig. 6, the noisy samples come from the same restricted workspaces as in Fig. 1 and 2. But, in spite of this, the degradation of the topology in the whole domain is more graceful than with the PSOM. Another important observation is the fact that the extrapolation remains very good if *only one* of the angle parameters (θ_1 *or* θ_2) lies outside the interval used for training. In Fig. 8, the extrapolated regions are

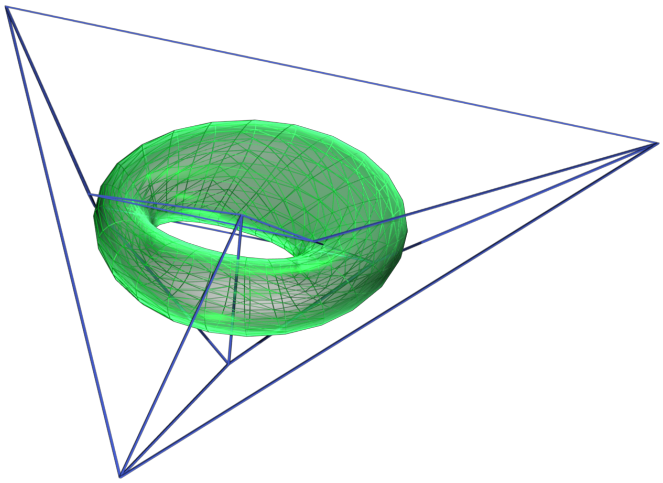


Fig. 6. Illustration of the extrapolation capabilities and noise robustness of the KB-Map (spline extension). The same setup used in previous experiments (see Fig. 2 and Fig. 5) was used for training. Again, 25 samples obtained with angles between 0° and 160° were degraded by artificial noise with $\sigma = 2^\circ$. This time, however, they were not chosen equidistantly from the interval but uniformly distributed. In the image one can see that, despite the noise, the estimate still resembles greatly a torus. This is due to the KB-Map's property that all of the estimate's main directions (optimally circles) are never transformed to anything different than ellipses. Further, the estimated surface does not necessarily pass through the erroneous training samples as the number of samples is greater than the system parameters ($3^2 = 9$) and a least squares solution can be found. These parameters are the points that form the blue colored control net surrounding the surface, which lies in their convex hull.

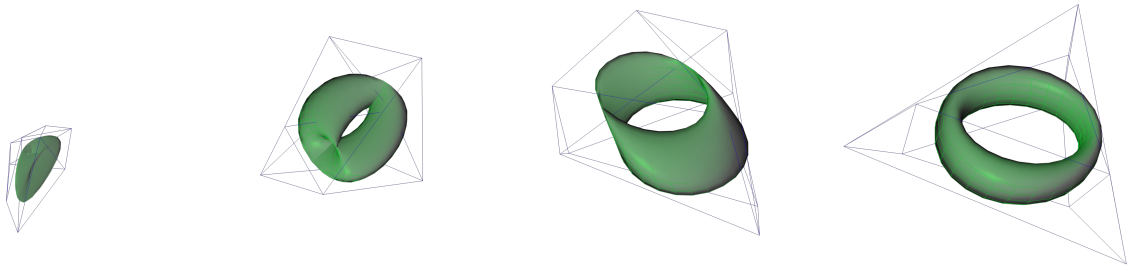
highlighted in red on the torus, whereas the original training range is tinted blue.

In the following experiments, these regions are used for measuring the quality of the extrapolation. They are subsets of the n-dimensional parameter space that

- have the same volume as the training region,
- are connected to it in $n-1$ dimensions and are completely disjoint with it.

The last two experiments on the torus deal with the incremental gradient algorithms provided with KB-Maps. The same torus as in the previous examples is learned with an initial control net where all vertices lie at the origin. In the sequence partially shown in Fig. 9, one can observe the unfolding net and the manifold as it is adapting to the torus. From the number of learned training samples one can see that incremental learning is significantly slower than batch learning. This is especially true if no approximate initial model is available and the samples are learned only once.

Finally, a perfect model of the known torus is used as the initial model for the incremental learning. Now we double the minor radius, i.e. $a_2 = 100$ *mm*. Fig. 7 displays the new model after a single learning step ($\mu_0 = 1$). The new constraint space is shown as a transparent surface in this image. One can see that the new model touches the constraint space in the learning sample whose position is indicated by the blue dot. In its main direction the model still consists of ellipses. Hence, just one learning step creates a model that is valid within a small region around the training sample.



(a) 5 samples (b) 10 samples (c) 20 samples (d) 40 samples

Fig. 9. Sequence showing the online learning process of a KB-Map at different numbers of learned samples. Unlike in the previous experiments, these samples are uniformly distributed over the whole parameter space.

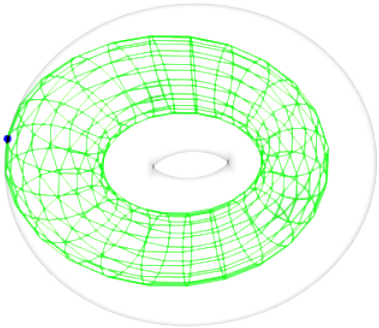


Fig. 7. Result of incremental learning after showing the fist training sample (*blue*) when the kinematics has changed. The *transparent* surface depicts the new constraint space after the minor radius was doubled.

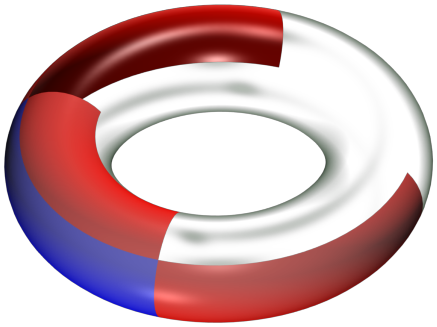


Fig. 8. Example of a parametrized constraint space of a 2R-mechanism. Highlighted are the regions from where training and test samples originate *(blue)* and those used for testing the accuracy of the extrapolation *(red)*. The latter are portions of the parameter space that have the same volume as the training region, that are connected to it but are completely disjoint with it.

B. Generic 6R-Mechanism

Now we explore if the conclusions above hold for a higher number of active DoF. For the following experiments, a simulated generic 6R-mechanism with six active rotational DoF has been used. It is defined by the following Denavit-Hartenberg parameters:

$$a_i = 200\ mm, \quad d_i = 0\ mm, \text{ and } \alpha_i = 90^\circ \qquad \forall i \in 1 \dots 6$$

An interpolation workspace, $\Theta^6_{in}(\delta)$, and an extrapolation workspace, $\Theta^6_{out}(\delta)$, dependent on a wideness parameter δ,

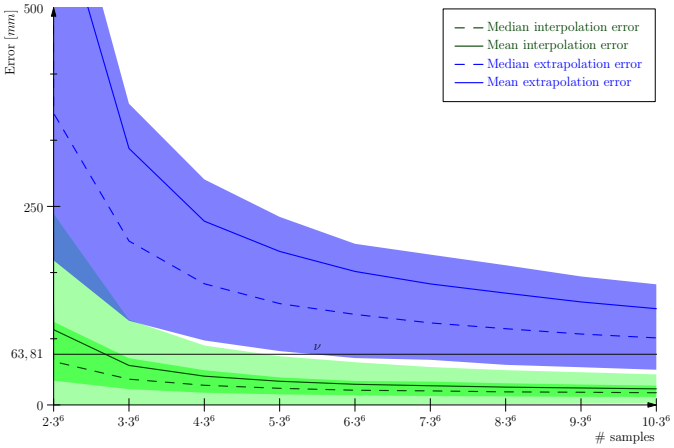


Fig. 10. This diagram shows convergence of the KB-Map learning algorithm when handling noisy training data. The *green* curves and regions show the accuracy of the general test data from $\Theta^6_{in}(45^\circ)$ and the *blue* indicates the extrapolation test data from $\Theta^6_{out}(45^\circ)$. For the general accuracy the *bright* and *dark green* areas depict the standard deviation and the interquartile range. Only the interquartile range is shown for the extrapolation test, i.e. 75% of all errors lie within this interval. The average error in the localization of the end-effector due to the artificial noise in the angle encoders is $\nu = 63,81\ mm$.

are defined in analogy to Fig. 8 as:

$$\Theta^6_{in}(\delta) = \{(\theta_1, \dots, \theta_6) : |\theta_i| \leq \delta, \forall i\} \tag{20}$$

and

$$\Theta^6_{out}(\delta) = \bigcup_{i=1}^{6} \{(\theta_1, \dots, \theta_6) : |\theta_j - 2\delta| \leq \delta \ \vee \ |\theta_j + 2\delta| \leq \delta, \forall j \neq i \wedge |\theta_i| \leq \delta\}. \tag{21}$$

Note that $\Theta^6_{out}(\delta)$ is twelve times larger than $\Theta^6_{in}(\delta)$. The training set and the interpolation test set in our experiments are built by sampling uniformly $\Theta^6_{in}(\delta)$. The extrapolation test set is constructed in the same way from samples out of $\Theta^6_{out}(\delta)$. Again, we will first investigate the KB-Maps and PSOM exact learning and then later the KB-Maps gradient learning.

1) Exact Learning: The first two experiments use the KB-Maps exact learning algorithm. They examine the relation between accuracy, and noise intensity and number of training samples, respectively. The goal is to assess the noise tolerance of the KB-Maps. A normally distributed noise with a standard deviation $\sigma = 2^\circ$ was added to each angle in the training sets of different KB-Maps. The KB-Maps differ only in the

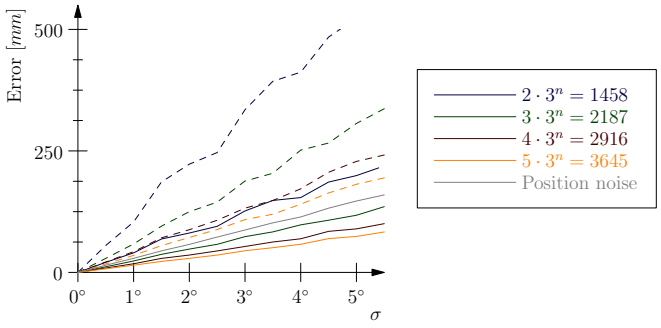


Fig. 11. Relation between accuracy and noise intensity when applying KB-Maps exact learning. Models with differently sized training sets are compared. The mean errors over the test data from $\Theta^6_{in}(90°)$ and extrapolation data from $\Theta^6_{out}(90°)$ are indicated as continuous and dashed lines, respectively.

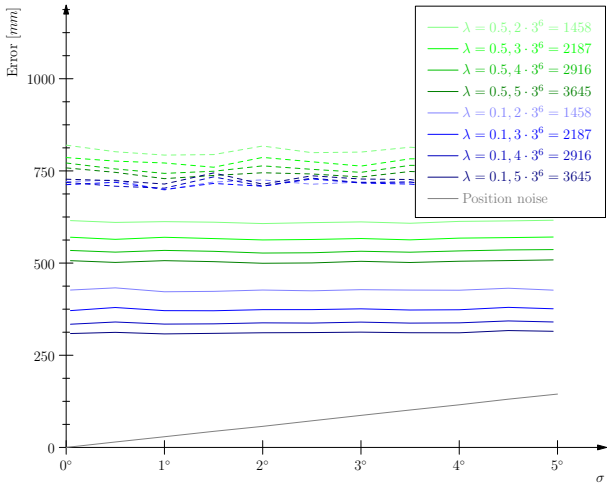


Fig. 12. Results of the PSOM+ algorithm learning capabilities in the conditions used in Fig. 11. Mean errors for interpolation and extrapolation (dashed lines) are shown in relation to variable noise.

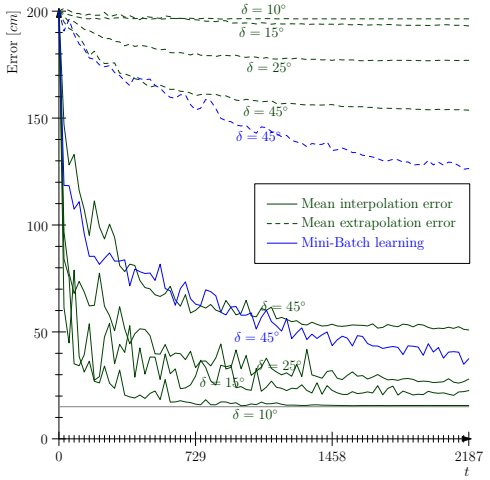


Fig. 13. Results of using the KB-Maps incremental learning algorithm to adapt to the usage of a tool. Workspace of different size δ (see (20) and (21)) were used. The horizontal grey line represents the mean data noise.

cardinality of their training sets, with elements drawn from $\Theta^6_{in}(45°)$. After learning, the models were validated using a test set of 3000 samples drawn from the same workspace to test interpolation accuracy and using another set of 3000 samples drawn from $\Theta^6_{out}(45°)$ to evaluate the extrapolation accuracy. Unlike the training sets, there was no noise in these two validation sets. The results are shown in Fig. 10. One can see that with the acquisition of about 3^6 samples, the mean error drops below the mean error of the training data. This means that it is even possible—given enough data—to compensate an erroneous perception up to a certain degree. Thanks to the information on the kinematics function encoded in the model, the amount of data required is small.

Furthermore, it is possible to see that the error is not normally distributed (as the mean and median errors do not coincide) and that few outliers with high errors occur.

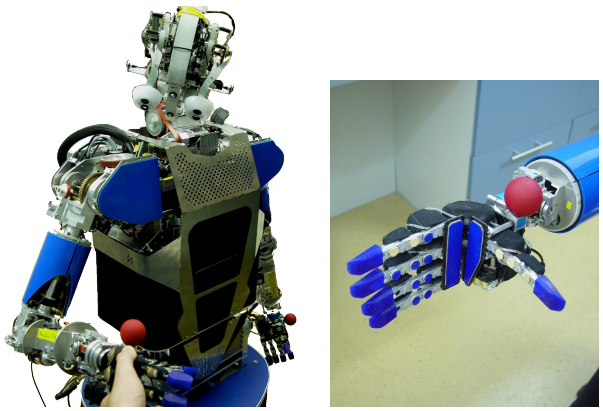
During the second experiment, the noise intensity is variable. The training set and the interpolation test set were drawn from $\Theta^6_{in}(90°)$, while the extrapolation test set was drawn from $\Theta^6_{out}(90°)$. Except for that, the conditions were the same as in the experiment before. The outcome is shown in Fig. 11. As has been visualized in Fig. 6 for the 2R-mechanism, the extrapolation error increases rapidly as soon as training data is noisy. Interestingly, the position errors and the noise intensity are proportional in this diagram. The estimates produced by the models, again, can be more accurate than the noisy observations. For interpolation this happens with about $3 \cdot 3^6$ samples. As a consequence, this means that this number suffices to deal with any (reasonable) intensity of noise.

The last experiment in this section compares these results with those from the PSOM+ algorithm. A regular lattice of 3^6 was created in $\Theta^6_{in}(90°)$ and used by PSOM+ nets that differed in the cardinality of their training sets and the smoothness parameter denoted as λ that influences the curvature metric. Results are depicted in Fig. 12. It is easy to see that using PSOM+ it is not possible to create an exact or even an accurate model. One can observe that the influence of noise is smaller than for the KB-Map, but even in the absence of noise ($\sigma = 0°$) the mean error in interpolation never falls below 300 *mm*. For a lattice of this size, the error cannot be improved by increasing the number of training samples. To achieve a higher interpolation accuracy, the number of knots has to be increased, resulting in a higher computational demand w.r.t. time and memory and possibly number of samples. In extrapolation, the difference in error is more blatant, especially when the intensity of noise is low.

2) Incremental / Gradient Learning: This experiment demonstrates the capability to learn the robot usage of a tool with an incremental learning scheme using the gradient algorithm of KB-Maps. Instead of learning the kinematics model 'from scratch', we use an initial model, i.e. the exact representation of the robot kinematics without the tool. After having created this initial model, the length of the last element of the kinematic chain was increased from the initial $a_6 = 200$ *mm* to $a'_6 = 400$ *mm*. As in previous experiments, a noise with standard deviation $\sigma = 2°$ was applied to only the training data angles. The experiment was performed with KB-Maps using sample sets from $\Theta^6_{in}(\delta_i)$ and $\Theta^6_{out}(\delta_i)$ with different angles δ_i. The goal is to evaluate the hypothesis that the gradient algorithm improves local estimates quickly.

The results are shown in Fig. 13. The error drops very fast for small values of δ and soon approaches the error in the perception caused by noise. It is important to see that this happens at a number of training samples smaller than the minimal number required for learning, which is $3^6 = 729$. However, the more locally the training samples are distributed, the less the learning affects the extrapolation accuracy. This means that incremental learning very quickly improves the accuracy in a region of the workspace. If the robot has to perform a single action repeatedly then this means that the required kinematics knowledge for this action can be acquired really fast. In Fig. 13, one KB-Map applies a variation of the learning scheme known as *mini-batch*. The learning samples are divided into blocks of constant size (in this experiment the size is 27) that are learned one after another. The data contained in the blocks are learned, however, more intensively by being fed to the algorithm repeatedly. Using this method, convergence proved to be quicker the bigger the block size chosen. Note that this evaluation can be performed at any arbitrary configuration of the robot when using the Bézier-Spline variant, whereas the classical KB-Map is more bound to their origin in the parameter space.

C. Humanoid Robot



(a) Manual movements via zero-force control.

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

Fig. 14. The humanoid robot ARMAR-IIIa that was used for the experiments.

Here we evaluate KB-Maps on the humanoid platform ARMAR-IIIa [18] (see Fig. 14(a)). The ARMAR-IIIa robot contains seven independent degrees of freedom (DoF) in each arm, one in the hip and three in the head. As our approach aims at hand-eye coordination, all experiments include joints of both the head and one arm. Training samples were generated by manually moving the robot arm via zero-force control (see Fig. 14(a)). Joint values were then read directly from the motor encoders, which provided very noisy values in this robot. An optical marker (a red ball signaling the end of a tool) attached to the wrist was tracked by the built-in stereo camera system (see Fig. 14(b)) and was considered as the end-effector. The two experiments presented below use original KB-Maps without Bézier splines.

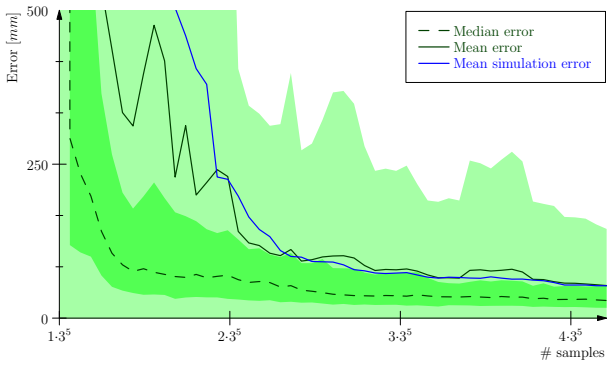


Fig. 15. Performance of the exact learning method for different numbers of training samples. The mean and median errors *(dark green)*, the interquartile range *(green area)* and the standard deviation *(light green area)* are shown for the experiment on real data. For comparison the mean errors from simulation *(blue)* are also included in this image.

In the first experiment, only five joints of the robot were effectively sampled as described to produce 1500 kinematic samples, of which 1000 were used as training samples and the remaining 500 ones as test set. Fig. 15 displays the result of using the exact algorithm with several sets of training samples of different cardinality. A second KB-Map was trained using exactly the same joint angles, but with the associated CAD-generated positions (simulated kinematics) and with an added noise of $\sigma_{noise} = 20$ *mm*, which is approximately of the same magnitude as the one in the perception system. As one can see from the similarity of both curves, the algorithm performs on real hardware as *predicted by the simulation*.

In the second experiment, the initial KB-Map implements an exact representation of the FK obtained from the CAD model of ARMAR-IIIa. Training and test data were produced in the same way as above after shifting the optical marker 250 *mm* to simulate tool usage. In this case, six joints were used to obtain 2200 samples, distributed in two sets of 700 and 1500 for training and testing, respectively. The gradient algorithm using a mini-batch size of 10 samples was used for learning. The results, displayed in Fig. 16), show that even with the high amount of noise in the encoders of ARMAR-IIIa plus the intrinsic noise in the tracking system, KB-Maps are able to quickly reduce the error to less than a quarter of the initial one using only 10 training samples. Note that this happens in the context of a rather high-dimensional kinematics (6 effective DoFs).

V. Conclusions

A novel approach for learning the FK mapping based on a special-purpose model was presented. Inspired by PSOMs, we aimed to overcome the large number of robot movements required to get a good approximation of FK.

First, since FK of angular robots is a composition of circles, models based on polynomials (as PSOM) cannot exactly represent 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 angular FK. Besides, these functions have an important

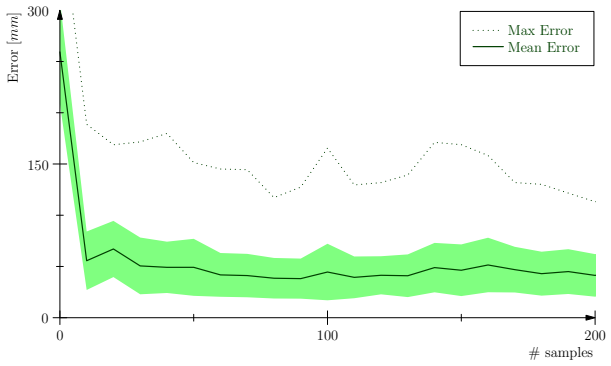


Fig. 16. In this image, the results of the gradient learning algorithm on the humanoid robot ARMAR-IIIa are presented. Maximal and median errors and the standard deviation (indicated by the *green area*) are shown in relation to the number of training samples.

advantage: adjusting the model to a set of sample points is a linear least squares problem.

Second, we have introduced *a priori* knowledge of the function to be learned 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 ellipses of a certain family 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 in projective space—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 robot DoFs, 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.

Another advantage of the model is that the Jacobian can be calculated very efficiently. This means that KB-Maps are very appropriate for approaches computing IK using a FK model through some kind of optimization. KB-Maps may potentially suffer from numerical problems when joint values are very close to $\pm\pi$, but these can be easily avoided with a variation using Bézier-Splines, i.e., a combination of three Béziers with common parameters always in a safe domain range.

We have carried out a lot of simulated experiments studying the relation between interpolation (and extrapolation) accuracy and the number of training samples and level of noise. The result is that, with a low level of noise, KB-Maps can be extremely accurate—even in extrapolation—with relative few training samples. Even under moderate noise, KB-Maps can be as accurate as desired if enough data are provided. And this accuracy is obtained with few parameters. This is in contrast to approaches using general-purpose models that do not only require progressively larger number of samples to reach arbitrary levels of accuracy, but also an indefinite increase in their complexity (hidden units in feedforward networks, grid points in PSOMs, stored points in Locally Weighted Projected Regression).

Another conclusion of our experiments is that there seems to exist a threshold number of training samples that suffices to get an accuracy better than that in the training data, no matter the level of noise. In general, 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. In comparison to other approaches, KB-Maps are more advantageous when the level of noise is not very high.

Finally, we have carried out experiments on real hardware, 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.

The behavior of KB-Maps is thus satisfactory in a wide range of conditions. 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 of how to improve our system under these conditions, although there does not exist any easy solution because the constraints to enforce complete circularity are non-linear. Another 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.

Finally, we have to point out that KB-Maps—in spite of their improvements—cannot escape from the exponential growth of required training samples as the number of robot DoFs increases. Because of this, for robots of seven or more DoFs, it is advisable to complement KB-Maps with a decomposition approach.

APPENDIX I
PROOF 1

In this section, it will be shown that the *tangent of the half angle substitution* applied to a Bézier satisfying the circle conditions presented in Section II-B exactly coincides with the angular parameterization of a circle. This will be shown for angles on the two-dimensional unit circle without loss of generality[1]. The following is a set of control points satisfying the conditions in 2D space:

$$\mathbb{b}_0 = \big(\cos\alpha, \sin\alpha, 1\big)^T,$$
$$\mathbb{b}_1 = \cos\alpha \cdot \big(0, \tfrac{1}{\cos\alpha}, 1\big)^T,$$
$$\mathbb{b}_2 = \big(-\cos\alpha, \sin\alpha, 1\big)^T.$$

Since the affine image of the two-dimensional Bézier spanned by these control points $\boldsymbol{b} = \begin{pmatrix} b_1 \\ b_2 \end{pmatrix}$ is a rational

[1]A general circle in 3D space can be obtained by adding a zero-valued affine coordinate and then translating, scaling, and rotating the control points of the unit circle. Affine transformation of the control points results in affine transformation of the Bézier curve.

parameterization of the circle, for each $|\theta| < 180°$ there exist a unique s such that

$$\begin{pmatrix}\cos(\theta)\\ \sin(\theta)\end{pmatrix} \stackrel{!}{=} \begin{pmatrix}b_1(s)\\ b_2(s)\end{pmatrix} = \begin{pmatrix}\frac{4s^2\cdot\cos\alpha - 4s^2 + \cos\alpha + 1}{-4s^2\cdot\cos\alpha + 4s^2 + \cos\alpha + 1}\\ \frac{4s\sin\alpha}{-4s^2\cdot\cos\alpha + 4s^2 + \cos\alpha + 1}\end{pmatrix} \quad (22)$$

The relation between θ and s can be determined by the trigonometric identity

$$\tan(\frac{\theta}{2}) = \frac{\sin\theta}{1+\cos\theta} = \frac{b_2(s)}{1+b_1(s)} = 2s\cdot\frac{\sin\alpha}{\cos\alpha+1} = 2s\cdot\tan(\frac{\alpha}{2}),$$

This leads to

$$s = \frac{\tan(\theta/2)}{2\cdot\tan(\alpha/2)}$$

Note that for $\theta = \pm 180°$ the Bézier parameter maps to $\pm\infty$.

Appendix II
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:

$$\mathbb{b}(s_1,\ldots,s_d) = \sum_{i_1,\ldots,i_d=0}^{2} \mathbb{b}_{i_1,\ldots,i_d}\cdot B^2_{i_1,\ldots,i_d}(s_1,\ldots,s_d). \quad (23)$$

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:

$$\sum_{k=0}^{2} B^2_k(s_1)\big(\sum_{i_2,\ldots,i_d=0}^{2} B^2_{i_2,\ldots,i_d}(s_2,\ldots,s_d)\cdot\mathbb{b}_{k,i_2,\ldots,i_d}\big). \quad (24)$$

We can define a new function $\mathfrak{q}_k(s_2,\ldots,s_d)$ to rename the expression in the big parenthesis; when $s_2,\ldots,s_d$ are fixed, $\mathfrak{q}_k$ is a constant and (24) becomes a single-variable Bézier curve defined by the control points $\mathfrak{q}_0$, $\mathfrak{q}_1$ and $\mathfrak{q}_2$:

$$\sum_{k=0}^{2} B^2_k(s_1)\cdot\mathfrak{q}_k(s_2,\ldots,s_d). \quad (25)$$

Let the homogeneous coordinates of $\mathfrak{q}_0$, $\mathfrak{q}_1$ and $\mathfrak{q}_2$ be ω_0, ω_1 and ω_2, respectively. To be an ellipse, ω_0=ω_2 and $\omega_1/\omega_0 < 1$ must be satisfied. Remember 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 $ones()$ is defined as in section II-C and γ is an arbitrary constant minor than one.

The values of the ω's are then

$$\omega_0 = \sum_{i_2,\ldots,i_d=0}^{2} B^2_{i_2,\ldots,i_d}(s_2,\ldots,s_d)\cdot\gamma_{0,i_2,\ldots,i_d}$$

$$\omega_1 = \sum_{i_2,\ldots,i_d=0}^{2} B^2_{i_2,\ldots,i_d}(s_2,\ldots,s_d)\cdot\gamma_{1,i_2,\ldots,i_d}$$

$$\omega_2 = \sum_{i_2,\ldots,i_d=0}^{2} B^2_{i_2,\ldots,i_d}(s_2,\ldots,s_d)\cdot\gamma_{2,i_2,\ldots,i_d}$$

Everything in the development of ω_0 is the same as that in ω_2, except the first index in the weights, which is 0 for ω_0 and 2 for ω_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 ω_0=ω_2. Similarly, ω_0 and ω_1 differ only in the first index of all involved weights. Those in ω_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 ω_0 multiplied by γ. Therefore, the conditions $\omega_0 = \omega_2$ and $\omega_1/\omega_0 = \gamma < 1$ 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 (23) are ellipses.

Appendix III
Bézier Spline Circles

The parameter transformation τ quickly produce large function values when it approaches the pole $\theta = \pm\pi$ (see App. I). The main idea to avoid this numerical problem is to divide each main direction ellipse of the tensor product representation into three curve segments. Each segment will have its own Bézier form whose parameter will always lie within the safe domain $[-\frac{\pi}{3}, \frac{\pi}{3})$. The domain of θ is subdivided into three sub-ranges, $[-\pi, -\frac{\pi}{3})$, $[-\frac{\pi}{3}, \frac{\pi}{3})$ and $[\frac{\pi}{3}, \pi)$. For each input joint angle, the right Bézier has to be chosen depending on the sub-range θ lies on. Since the non-central control points of each Bézier will be shared by two Bézier's, we need in total six control points. In a Bézier curve with control points $\boldsymbol{b}_0$, $\boldsymbol{b}_1$ and $\boldsymbol{b}_2$, the straight segment $\boldsymbol{b}_0\boldsymbol{b}_1$ is tangent to the curve at $\boldsymbol{b}_0$ and the segment $\boldsymbol{b}_1\boldsymbol{b}_2$ is tangent at point $\boldsymbol{b}_2$. Therefore, to guarantee an smooth connection from one Bézier to the next, the control point common to two Bézier curves must be collinear with the central control points of the two Bézier forms. Moreover, we will require that this common point is just in the middle of the two central control points. These additional constraints compensate for the increase in the number of control points that, otherwise, would require also a higher amount of training data to determine the model. Finally we set $\alpha = \cos\frac{\pi}{3}$, because in the case of the ellipse being a circle, this setting allows to represent it without error (see the equilateral triangle formed by the control points in Fig. 17).

All this will be illustrated for the blue spline segment at the top of Figure 17. The affine part of the spline is

$$\boldsymbol{b}_0\cdot B_0(\tau(\theta)) + \alpha\cdot\boldsymbol{b}_1\cdot B_1(\tau(\theta)) + \boldsymbol{b}_2\cdot B_2(\tau(\theta)), \quad (26)$$

while the homogeneous coordinate is

$$B_0(\tau(\theta)) + \alpha\cdot B_1(\tau(\theta)) + B_2(\tau(\theta)), \quad (27)$$

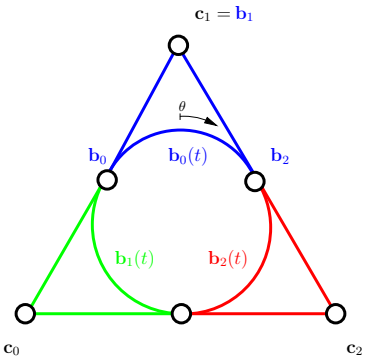


Fig. 17. The subdivision of a circle into the three segments of the *Bézier spline* representation.

which is the same for the three splines. After applying the new constraints we get:

$$\begin{aligned}\boldsymbol{b}_0 &= \frac{\boldsymbol{c}_0 + \boldsymbol{c}_1}{2},\\ \boldsymbol{b}_1 &= \boldsymbol{c}_1,\\ \boldsymbol{b}_2 &= \frac{\boldsymbol{c}_2 + \boldsymbol{c}_1}{2}.\end{aligned} \tag{28}$$

Inserted in eq. 26, this results in

$$\begin{aligned}\frac{\boldsymbol{c}_0 + \boldsymbol{c}_1}{2} \cdot B_0(\tau(\theta)) + \alpha \cdot \boldsymbol{c}_1 \cdot B_1(\tau(\theta))\\ + \frac{\boldsymbol{c}_2 + \boldsymbol{c}_1}{2} \cdot B_2(\tau(\theta)).\end{aligned}$$

Finally, this leads to a reformulation of this spline segment:

$$\boldsymbol{c}_0 \cdot C_0^0(\theta) + \boldsymbol{c}_1 \cdot C_1^0(\theta) + \boldsymbol{c}_2 \cdot C_2^0(\theta), \tag{29}$$

using a new set of basis polynomials defined as:

$$\begin{aligned}C_0^0(\theta) &:= \tfrac{1}{2} \cdot B_0(\tau(\theta)),\\ C_1^0(\theta) &:= \tfrac{1}{2} \cdot B_0(\tau(\theta)) + \alpha \cdot B_1(\tau(\theta)) + \tfrac{1}{2} \cdot B_2(\tau(\theta))\\ &= \tfrac{1}{2} \cdot \tau(\theta)^2 + \cos 60° \cdot 2(1-\tau(\theta)) \cdot \tau(\theta)\\ &\quad + \tfrac{1}{2}(1-\tau(\theta))^2 = \tfrac{1}{2}(\tau(\theta) + (1-\tau(\theta)))^2\\ &= \tfrac{1}{2},\\ C_2^0(\theta) &:= \tfrac{1}{2} \cdot B_2(\tau(\theta)).\end{aligned} \tag{30}$$

The other spline segments can be constructed analogously by swapping the control vertices in eq. 30 and mapping the angle θ into the support interval of the Bézier segments. Hence, the more general definition of the whole spline curve is given by:

$$\mathbb{b}(\theta) = \boldsymbol{c}_0 \cdot C_0(\theta) + \boldsymbol{c}_1 \cdot C_1(\theta) + \boldsymbol{c}_2 \cdot C_2(\theta),$$

where

$$C_0(\theta) = \begin{cases} \frac{1}{2}, & \theta \in [-\pi, -\frac{\pi}{3})\\ \frac{1}{2} \cdot B_0(\tau(\theta)), & \theta \in [-\frac{\pi}{3}, \frac{\pi}{3})\\ \frac{1}{2} \cdot B_2(\tau(\theta - \frac{\pi}{3})), & \theta \in [\frac{\pi}{3}, \pi)\end{cases}$$

$$C_1(\theta) = \begin{cases} \frac{1}{2} \cdot B_2(\tau(\theta + \frac{\pi}{3})), & \theta \in [-\pi, -\frac{\pi}{3})\\ (\tau(\theta))\frac{1}{2}, & \theta \in [-\frac{\pi}{3}, \frac{\pi}{3})\\ \frac{1}{2} \cdot B_0(\tau(\theta - \frac{\pi}{3})), & \theta \in [\frac{\pi}{3}, \pi)\end{cases} \tag{31}$$

$$C_2(\theta) = \begin{cases} \frac{1}{2} \cdot B_0(\tau(\theta + \frac{\pi}{3})), & \theta \in [-\pi, -\frac{\pi}{3})\\ \frac{1}{2} \cdot B_2(\tau(\theta)), & \theta \in [-\frac{\pi}{3}, \frac{\pi}{3})\\ \frac{1}{2}, & \theta \in [\frac{\pi}{3}, \pi)\end{cases}.$$

Still, all techniques presented in the sections above also apply for the new form of the Bézier models—one only has to substitute all Bernstein polynomials by the new basis $C_{\boldsymbol{i}}(\boldsymbol{\theta})$.

Appendix IV
The Jacobian of the Forward Kinematics

The Jacobian of the Forward Kinematics

$$J_{\boldsymbol{b}}(\boldsymbol{\theta}) := \left(\frac{\partial}{\partial \tau(\theta_1)} \boldsymbol{b}(\tau(\boldsymbol{\theta})), \ldots, \frac{\partial}{\partial \tau(\theta_d)} \boldsymbol{b}(\tau(\boldsymbol{\theta})) \right).$$

plays an important role, for instance, in iterative algorithms that try to solve the inverse kinematics problem. As it has to be obtained frequently, a rapid calculation can be crucial for real-time applications. When using the Bézier form of the forward kinematics, its calculation is very fast as only matrix-vector multiplications are involved that can be directly accelerated by parallel hardware. A partial derivative of a regular tensor product Bézier function of degree n is another Bézier function of degree $n-1$. Therefore, the Jacobian is constructed by calculating a control net for each partial derivation. These control nets are invariant w.r.t. the function parameters, allowing to be computed only once during an initializing process. For the KBM, however, we use a sightly different construction to that in eq. 2 in order to speed up the calculus:

$$\frac{\partial}{\partial s_k} \boldsymbol{b}(\boldsymbol{s}) = n \cdot \sum_{\boldsymbol{i}} \Delta^k \boldsymbol{b}_{\boldsymbol{i}} \cdot B_{\boldsymbol{i}}^2(\boldsymbol{s}), \tag{32}$$

$$\text{where} \quad \Delta^k \boldsymbol{b}_{\boldsymbol{i}} = \begin{cases} \boldsymbol{b}_{\boldsymbol{i}+\mathbf{1}_k} - \boldsymbol{b}_{\boldsymbol{i}} & , \quad i_k \in \{0, 2\},\\ \frac{1}{2} \cdot (\Delta^k \boldsymbol{b}_{\boldsymbol{i}+\mathbf{1}_k} + \boldsymbol{b}_{\boldsymbol{i}-\mathbf{1}_k}) & , \quad i_k = 1,\end{cases} \tag{33}$$

$\boldsymbol{i}$ goes through $\mathcal{I}_2$, and $\mathbf{1}_k$ denotes a vector with a one at the k-th position and otherwise zeros, and i_k the k-th component of the index vector $\boldsymbol{i}$. A degree elevation [20] in direction k takes place directly after the differentiation by defining an intermediate *collinear* control vector in eq. 33 (in the case of $i_k = 1$). This way the derivative is also a quadratic a function. The advantage of this redundant representation is that the set of Bernstein polynomials in (32) is the same for all partial derivatives and the original function. Thanks to this, the evaluation and calculation of the Jacobian can be greatly accelerated. The partial derivatives of rational polynomials $\boldsymbol{b}(\boldsymbol{s}) = \frac{\mathbf{b}(\boldsymbol{s})}{\gamma(\boldsymbol{s})}$ resulting from the projection onto the affine space

of homogeneous poylynomials

$$\mathbb{b}(\boldsymbol{s}) = \begin{bmatrix} \sum_{\boldsymbol{i}} \gamma_{\boldsymbol{i}} \boldsymbol{b_i} \cdot B_{\boldsymbol{i}}(\boldsymbol{s}) \\ \gamma(\boldsymbol{s}) \end{bmatrix} = \begin{bmatrix} \mathbf{b}(\boldsymbol{s}) \\ \gamma(\boldsymbol{s}) \end{bmatrix}$$

are computed by means of the quotient rule without the need of a lot of additional calculations:

$$\frac{\partial}{\partial s_i} \boldsymbol{b}(\boldsymbol{s}) = \frac{\partial}{\partial s_i} \frac{\mathbf{b}(\boldsymbol{s})}{\gamma(\boldsymbol{s})} = \frac{\frac{\partial}{\partial s_i}\mathbf{b}(\boldsymbol{s}) \cdot \gamma(\boldsymbol{s}) + \mathbf{b}(\boldsymbol{s}) \cdot \frac{\partial}{\partial s_i}\gamma(\boldsymbol{s})}{\gamma(\boldsymbol{s})^2}. \tag{34}$$

Note that we still have to apply the inner derivative of $\tau(\cdot)$ in order to obtain the final partial derivative.

ACKNOWLEDGMENT

The authors would like to thank S. Klanke and H. Ritter for providing the PSOM+ toolbox and G. Karich who helped performing the experiments involving the PSOM+.

REFERENCES

[1] S. Ulbrich, V. Ruiz, T. Asfour, C. Torras, and R. Dillmann, "Rapid learning of humanoid body schemas with kinematic bézier maps," in *Proc. IEEE-RAS International Conference on Humanoid Robots-09*, Paris, France, Dec. 2009, pp. 431–438.

[2] J. Schmudderich, V. Willert, J. Eggert, S. Rebhan, C. Goerick, G. Sagerer, and E. Korner, "Estimating object proper motion using optical flow, kinematics, and depth information," *IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics*, vol. 38, no. 4, pp. 1139 –1151, Aug. 2008.

[3] V. de Angulo and C. Torras, "Self-calibration of a space robot," *IEEE Transactions on Neural Networks*, vol. 8, no. 4, pp. 951 –963, Jul 1997.

[4] H. Ritter, T. Martinetz, and K. Schulten, *Neural Computation and Self-Organizing Maps; An Introduction*. Boston, MA, USA: Addison-Wesley Longman Publishing Co., Inc., 1992.

[5] A. D'Souza, S. Vijayakumar, and S. Schaal, "Learning inverse kinematics," in *Proc. IEEE International Conference on Intelligent Robots and Systems (IROS-01)*. Piscataway, New Jersey: IEEE, 2001.

[6] S. Vijayakumar, A. D'souza, T. Shibata, J. Conradt, and S. Schaal, "Statistical learning for humanoid robots," *Autonomous Robots*, vol. 12, no. 1, pp. 55–69, Jan 2002.

[7] D. Whitney, "Resolved motion rate control of manipulators and human prostheses," *IEEE Transactions on Man-Machine Systems*, vol. 10, no. 2, pp. 47–53, June 1969.

[8] A. Ligeois, "Automatic supervisory control of the configuration and behavior of multibody mechanisms," *IEEE Transactions on Systems, Man and Cybernetics*, vol. 7, no. 12, pp. 868 –871, Dec. 1977.

[9] L. Li, W. Gruver, Q. Zhang, and Z. Yang, "Kinematic control of redundant robots and the motion optimizability measure," *IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics*, vol. 31, no. 1, pp. 155 –160, Feb. 2001.

[10] Y. Xia, G. Feng, and J. Wang, "A primal-dual neural network for online resolving constrained kinematic redundancy in robot motion control," *IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics*, vol. 35, no. 1, pp. 54 –64, Feb. 2005.

[11] G. Sun and B. Scassellati, "Reaching through learned forward model," in *IEEE-RAS International Conference on Humanoid Robots-04*, vol. 1, Nov. 2004, pp. 93–112.

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

[13] S. Klanke and H. J. Ritter, "PSOM+ : Parametrized self-organizing maps for noisy and incomplete data," in *Proc. of the 5th Workshop on Self-Organizing Maps (WSOM-05)*, Paris, France, Sept. 2005.

[14] L. C. T. Wang and C. C. Chen, "A combined optimization method for solving the inverse kinematics problems of mechanical manipulators," *IEEE Transactions on Robotics and Automation*, vol. 7, no. 4, pp. 489–499, 1991.

[15] V. R. de Angulo and C. Torras, "Speeding up the learning of robot kinematics through function decomposition," *IEEE Transactions on Neural Networks*, vol. 16, no. 6, pp. 1504–1512, 2005.

[16] ——, "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.

[17] M. Hersch, E. Sauser, and A. Billard, "Online learning of the body schema," *International Journal of Humanoid Robotics*, vol. 5, no. 2, pp. 161–181, 2008.

[18] 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," in *Proc. IEEE-RAS International Conference on Humanoid Robots-06*, Dec. 2006, pp. 169–175.

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

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

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

[22] R. Penrose, "A generalized inverse for matrices," in *The Cambridge Philosophical Society*, ser. 51, 1955, pp. 406–413.

[23] B. Widrow and M. E. Hoff, "Adaptive switching circuits," in *IRE WESCON Convention Record-60*. IRE, 1960, pp. 96–104, reprinted in *Neurocomputing: Foundations of Research* (MIT Press, 1988).