General Robot Kinematics Decomposition without Intermediate Markers

Stefan Ulbrich, Vicente Ruiz de Angulo, Tamim Asfour, *Member, IEEE,* Carme Torras, *Senior Member, IEEE,* and Rüdiger Dillmann, *Senior Member, IEEE*

***Abstract*—The calibration of serial manipulators with high numbers of degrees of freedom by means of machine learning is a complex and time-consuming task. With the help of a simple strategy, this complexity can be drastically reduced and the speed of the learning procedure can be increased: When the robot is virtually divided into shorter kinematic chains, these subchains can be learned separately and, hence, much more efficiently than the complete kinematics. Such decompositions, however, require either the possibility to capture the poses of all end-effectors of all subchains at the same time, or they are limited to robots that fulfill special constraints. In this work, an alternative decomposition is presented that does not suffer from these limitations. An offline training algorithm is provided in which the composite subchains are learned sequentially with dedicated movements. A second training scheme is provided to train composite chains simultaneously and online. Both schemes can be used together with many machine learning algorithms. In the simulations, an algorithm using Parameterized Self-Organizing Maps (PSOM) modified for online learning and Gaussian Mixture Models (GMM) were chosen to show the correctness of the approach. The experimental results show that, using a two-fold decomposition, the number of samples required to reach a given precision is reduced to twice the square root of the original number.**

***Index Terms*—Redundant robot kinematics, kinematics decomposition, automatic recalibration, autonomous learning.**

I. Introduction

With higher numbers of degrees of freedom (DoF) the calibration of serial manipulators (e.g., *anthropomorphic* manipulators) becomes increasingly complex and expensive [1]. In such systems, the need for calibration arises more often either due to deformations or—much more interestingly—because of reconfigurations such as tool-use. Instead of the costly traditional calibration routines, machine learning techniques can be used to learn the correlation between the joint angle configuration and the spatial pose of the end-effector, the *forward kinematics (FK)*. Usually, learning is accomplished by observing examples of input/output pairs of valid FK configurations. Many suitable learning algorithms have been proposed for this task. Among them there are the continuous extension of Kohonen maps, the *Parameterized Self Organizing Maps (PSOM)* [2], hierarchical artificial neural networks [3], [4], local learning such as *Locally Weighted Projection Regression (LWPR)* [5], and *Gaussian Mixture Models (GMM)* [6], [7]. However, no learning algorithm can avoid the exponential growth in the number n of DoF required to directly represent the FK with enough accuracy ([8], [9]), i.e., the cost $\mathcal{O}(q^n)$, where q is the number of sample points in each joint dimension (assuming, for the sake of simplicity, that the samples are obtained following rectangular grids). This was the motivation for this work since the number of movements required in our humanoid robot [10] was impractical, even using state-of-the-art methods to learn FK. An effective way to palliate this problem are *decomposition* techniques [8], [11]. Hereby, the robot is virtually divided into two (or more) subchains with fewer DoF each. These subchains can be learned much more efficiently than the complete chain; and the number of required training samples (for a decomposition into two chains) can be reduced to about its square root $\mathcal{O}(q^{\frac{n}{2}})$. The decompositions are known to work with many different learning systems.

Current techniques of learning by decomposition, however, have shortcomings. In [8], a decomposition is proposed that can be easily applied to robot manipulators whose last three axes intersect in a single point. This constraint excludes many possible robot architectures and may not hold anymore after a manipulator has suffered a deformation. A second approach that is general w.r.t. the choice of the robot architecture has been presented in [11]. However, it requires the ability to observe the spatial pose of all subchains' end-effectors at the same time in order to be able to learn. While this may be perfectly appropriate in setups with external cameras, the higher sensorial demand may exclude robots that learn from pure self-observation as it is the case of many humanoid robots.

This work presents a third option that is general w.r.t. the robot architecture and requires only the visibility of the original end-effector, at the expense of a more complex learning scheme. A batch algorithm well-suited for initial learning requires that, during the training of one subchain, the other subchains remain unchanged. This way, enough information can be gathered without the need to know the location of the individual subchains' end-effectors or origins, respectively. After an initial training, the decomposed kinematics can adapt online to many deformations such as a shift in the joint encoders or reconfigurations when using a tool. In contrast to the initial batch learning, this *online learning* allows for

The work described in this paper was partially conducted within the EU Cognitive Systems projects Xperience (FP-7-270273) and GARNICS (FP-7-247947) funded by the European Commission, and the Generalitat de Catalunya through the Robotics group (SGR2009-00155).

V. Ruiz de Angulo acknowledges support from Spanish Ministry of Science and Education, under the project DPI2010-18449.

C. Torras acknowledges support from the Consolider project MIPRCV (CSD2007-00018).

S. Ulbrich, T. Asfour and R. Dillmann are with the Institute for Anthropomatics of the Karlsruhe Institute of Technology.

V. Ruiz de Angulo and C. Torras are with the Institut de Robòtica i Informàtica Industrial, CSIC-UPC.

simultaneous movements of all subchains and it can be used during the operation of the robot.

In the simulations, these principles are validated using the new decomposition in conjunction with the PSOM learning system which is used, in this context, as a method for function approximation (as done in [8]). While PSOM originally does not offer online learning, we could successfully apply the Widrow-Hoff rule (also known as δ-rule) [12] to the weights of this artificial neural network.

In the following two sections, the principles of the new proposed decomposition and the composition of the separately learned functions will be explained, respectively. The following section presents both the batch and the online learning algorithms. The document concludes with simulations and an outlook on future work.

II. Kinematics Decomposition

The proposed decomposition approach consists in using two kinematics functions that depend on disjoint subsets of the joint values. In Fig. 1, an example of the functions is provided for a robot with four rotational degrees of freedom.

We partition the joint variables $\boldsymbol{\theta} = (\theta_1, \theta_2, ..., \theta_n)$ into two subsets $\boldsymbol{\zeta} = (\zeta_1, \zeta_2, ..., \zeta_k) := (\theta_1, \theta_2, ..., \theta_k)$ and $\boldsymbol{\mu} = (\mu_1, .., \mu_{n-k}) := (\theta_{k+1}, ..., \theta_n)$, that is, $\boldsymbol{\zeta}$ is the set of the first k joints and $\boldsymbol{\mu}$ the final $n-k$ ones. Then the direct kinematics function of the robot $K(\boldsymbol{\theta})$ (or $K(\boldsymbol{\zeta}, \boldsymbol{\mu})$ for convenience)[1] can be expressed as

$$K(\boldsymbol{\theta}) = K(\boldsymbol{\zeta}, \boldsymbol{\mu}) = K_{\boldsymbol{\zeta}}(\boldsymbol{\zeta}) \cdot K_{\boldsymbol{\mu}}(\boldsymbol{\mu}), \tag{1}$$

where $K_{\boldsymbol{\zeta}}$ and $K_{\boldsymbol{\mu}}$ are the kinematics of the two subchains of the robot implicitly defined by $\boldsymbol{\zeta}$ and $\boldsymbol{\mu}$, respectively. The joints that form the subchains K_ζ and K_μ must be composed of *adjacent* joints.

The first function in the decomposition is

$$K_1(\boldsymbol{\zeta}; \tilde{\boldsymbol{\mu}}) := K(\boldsymbol{\zeta}, \tilde{\boldsymbol{\mu}}), \tag{2}$$

where $\tilde{\boldsymbol{\mu}}$ is an arbitrarily fixed value for $\boldsymbol{\mu}$. This function can then be reformulated as

$$K_1(\boldsymbol{\zeta}; \tilde{\boldsymbol{\mu}}) = K_{\boldsymbol{\zeta}}(\boldsymbol{\zeta}) \cdot K_{\boldsymbol{\mu}}(\tilde{\boldsymbol{\mu}}) = K_{\boldsymbol{\zeta}}(\boldsymbol{\zeta}) \cdot C_{\tilde{\boldsymbol{\mu}}}, \tag{3}$$

where $C_{\tilde{\boldsymbol{\mu}}}$ is a constant transformation matrix associated to $\tilde{\boldsymbol{\mu}}$.

The second function, $K_2(\boldsymbol{\mu}; \tilde{\boldsymbol{\mu}})$, is the one that transforms $K(\boldsymbol{\zeta}, \tilde{\boldsymbol{\mu}})$ to $K(\boldsymbol{\zeta}, \boldsymbol{\mu})$, that is, it satisfies

$$K(\boldsymbol{\zeta}, \boldsymbol{\mu}) = K(\boldsymbol{\zeta}, \tilde{\boldsymbol{\mu}}) \cdot K_2(\boldsymbol{\mu}; \tilde{\boldsymbol{\mu}}). \tag{4}$$

Using $K_1(\boldsymbol{\zeta}; \tilde{\boldsymbol{\mu}}) = K(\boldsymbol{\zeta}, \tilde{\boldsymbol{\mu}})$ the above equation can be expressed as

$$K(\boldsymbol{\zeta}, \boldsymbol{\mu}) = K_1(\boldsymbol{\zeta}; \tilde{\boldsymbol{\mu}}) \cdot K_2(\boldsymbol{\mu}; \tilde{\boldsymbol{\mu}}). \tag{5}$$

It is easy to check that K_2 is independent of $\boldsymbol{\zeta}$. Solving for K_2,

$$K_2(\boldsymbol{\mu}; \tilde{\boldsymbol{\mu}}) = K(\boldsymbol{\zeta}, \tilde{\boldsymbol{\mu}})^{-1} \cdot K(\boldsymbol{\zeta}, \boldsymbol{\mu}), \tag{6}$$

and developing K into the two component kinematics, one gets

$$\begin{aligned} K_2(\boldsymbol{\mu}; \tilde{\boldsymbol{\mu}}) &:= (K_{\boldsymbol{\zeta}}(\boldsymbol{\zeta}) \cdot K_{\boldsymbol{\mu}}(\tilde{\boldsymbol{\mu}}))^{-1} \cdot K_{\boldsymbol{\zeta}}(\boldsymbol{\zeta}) \cdot K_{\boldsymbol{\mu}}(\boldsymbol{\mu}) \\ &= K_{\boldsymbol{\mu}}(\tilde{\boldsymbol{\mu}})^{-1} \cdot K_{\boldsymbol{\zeta}}(\boldsymbol{\zeta})^{-1} \cdot K_{\boldsymbol{\zeta}}(\boldsymbol{\zeta}) \cdot K_{\boldsymbol{\mu}}(\boldsymbol{\mu}) \\ &= C_{\tilde{\boldsymbol{\mu}}}^{-1} \cdot \ K_{\boldsymbol{\mu}}(\boldsymbol{\mu}). \end{aligned} \tag{7}$$

Now, it is also clear that K_2 has the shape of a kinematics function with $n-k$ degrees of freedom. In the end, we come up with two functions that depend only on one of the two disjoint subsets of variables. We would like to inform the reader that, alternatively, there exists a complementary decomposition not commented in depth in this article[2].

Since K_1 and K_2 are kinematics functions, we can apply the decomposition to one or both of them. In this way, the original chain can be decomposed into as many chains as desired (of course, n being the limit). If the desired number of chains in the decomposition is d, ideally the number of joints in each chain should be as close as possible to n/d as argued in the next section. For this purpose, the following recursive algorithm can be applied to a chain of arbitrary length. The original chain is divided into two subchains – one of them with $\lceil n/d \rceil$ joints (which will be the maximal length of a chain in the decomposition). The recursion proceeds with the remaining subchain of $n - \lceil n/d \rceil$ joints, which is divided again. The algorithm terminates when the chain to be processed is shorter or equal than $\lceil n/d \rceil$. Without loss of generality, we will assume a decomposition into two chains in the remaining of the paper.

III. Kinematics Composition

The forward kinematics (FK) is obtained from (5). $K_1(\boldsymbol{\zeta}; \tilde{\boldsymbol{\mu}})$ and $K_2(\boldsymbol{\mu}; \tilde{\boldsymbol{\mu}})$ will be approximated by two learning systems (e.g, neural networks) N_1 and N_2, respectively. Therefore the FK will be estimated with

$$N(\boldsymbol{\zeta}, \boldsymbol{\mu}) = N_1(\boldsymbol{\zeta}) \cdot N_2(\boldsymbol{\mu}). \tag{8}$$

Now, we can easily justify that the number k, which determines the number of joints in each chain, should be chosen close to $n/2$ in general. As for the whole robot, we can assume that the number of samples that we need to approximate $K_1(\boldsymbol{\zeta}; \tilde{\boldsymbol{\mu}})$ and $K_2(\boldsymbol{\mu}; \tilde{\boldsymbol{\mu}})$ depends on the number of joints in $\boldsymbol{\zeta}$ and $\boldsymbol{\mu}$, respectively. Therefore, the number of samples needed by the decomposition approach is $q^k + q^{n-k}$. The minimum of this quantity as a function of k occurs when $k = n/2$, and increases exponentially as k differs from the minimum $n/2$.

Regarding the inverse kinematics (IK), given a desired pose T, the joint coordinates $\boldsymbol{\theta} = (\zeta_1, \ldots, \zeta_k, \mu_1, \ldots, \mu_{n-k})$ form a valid inverse kinematics solution if

$$K(\boldsymbol{\zeta}, \boldsymbol{\mu}) = K_1(\boldsymbol{\zeta}; \tilde{\boldsymbol{\mu}}) \cdot K_2(\boldsymbol{\mu}; \tilde{\boldsymbol{\mu}}) = T, \tag{9}$$

which can be approximated with

$$N(\boldsymbol{\zeta}, \boldsymbol{\mu}) = N_1(\boldsymbol{\zeta}) \cdot N_2(\boldsymbol{\mu}) = T. \tag{10}$$

[1]All kinematic functions $K_\square$: $\mathbb{R}^n \rightarrow$ SE(3) are defined as mappings from the joint space into the group of rigid motions, whose elements can be expressed by homogeneous transformation matrices, for instance, or dual quaternions. In this work, we have chosen to use homogeneous matrices for the exposition, but our composition approach is also valid when other representations are used.

[2]The alternative decomposition is $L_1(\boldsymbol{\mu}; \tilde{\boldsymbol{\zeta}}) = K(\tilde{\boldsymbol{\zeta}}, \boldsymbol{\mu})$, $L_2(\boldsymbol{\zeta}; \tilde{\boldsymbol{\zeta}}) = K(\boldsymbol{\zeta}, \boldsymbol{\mu}) \cdot L_1(\boldsymbol{\mu}; \tilde{\boldsymbol{\zeta}})^{-1}$. The kinematics composition is obtained from the definition of L_2 , $K(\boldsymbol{\zeta}, \boldsymbol{\mu}) = L_2(\boldsymbol{\zeta}; \tilde{\boldsymbol{\zeta}}) L_1(\boldsymbol{\mu}; \tilde{\boldsymbol{\zeta}})$.

(a) $K_1(\boldsymbol{\zeta};\tilde{\boldsymbol{\mu}}) = K_\zeta(\boldsymbol{\zeta}) \cdot C_{\tilde{\boldsymbol{\mu}}}$ (b) $K_2(\boldsymbol{\mu};\tilde{\boldsymbol{\mu}}) = C_{\tilde{\boldsymbol{\mu}}}^{-1} \cdot K_\mu(\boldsymbol{\mu})$ (c) $K(\boldsymbol{\zeta},\boldsymbol{\mu}) = K_1(\boldsymbol{\zeta};\tilde{\boldsymbol{\mu}}) \cdot K_2(\boldsymbol{\mu};\tilde{\boldsymbol{\mu}})$

Fig. 1. Example of the decomposition for a robot with four rotational degrees of freedom. The first kinematics function K_1 is shown in (a). It is the transformation from the robot base to its end-effector when the last (two) degrees of freedom are assigned to constant values (namely $\tilde{\boldsymbol{\mu}}$). The constant part of the robot is called $C_{\tilde{\boldsymbol{\mu}}}^{-1}$ (see (3)). This part of the robot is rendered transparently in this image. During learning, the end-effector frame is tracked while moving the first two axes. The second kinematics function shown in (b) is K_2. This function is a composition of the last half of the robot (i.e., K_μ) with two active joints and $C_{\tilde{\boldsymbol{\mu}}}$ (see (7)) which is, again, displayed transparently. That is, K_2 is the transformation from the tail of K_1 to the real end-effector frame. When learning this function, the real end-effector (opaque) is tracked while the first two joints are fixed to the reference values in $\tilde{\boldsymbol{\zeta}}$. Consequently, as shown in (c), the combination of K_1 and K_2 results in the complete robot transformation.

The constraint (9) can be rewritten in another form:

$$\begin{aligned} & K_{\boldsymbol{\zeta}}(\boldsymbol{\zeta}) \cdot C_{\tilde{\boldsymbol{\mu}}} \cdot C_{\tilde{\boldsymbol{\mu}}}^{-1} \cdot \; K_{\boldsymbol{\mu}}(\boldsymbol{\mu}) &&= T \\ \Leftrightarrow \quad & K_{\boldsymbol{\zeta}}(\boldsymbol{\zeta}) &&= T \cdot K_{\boldsymbol{\mu}}(\boldsymbol{\mu})^{-1}. \end{aligned} \tag{11}$$

where equations (3) and (7) have been used.

This is the same equality used in [11]. The first subchain of the robot must be the same as the last one reverted and transformed to the desired pose. As mentioned earlier, a limitation of this approach is that, in order to learn $K_{\boldsymbol{\zeta}}(\boldsymbol{\zeta})$ and $K_{\boldsymbol{\mu}}(\boldsymbol{\mu})^{-1}$, one needs to detect the pose of an intermediate marker placed in the k-th link. The advantage of (10) is that, although the underlying constraint is the same, the involved functions K_1 and K_2 can be learned by using only the ability to detect the end-effector pose (see next section).

There exist many ways to satisfy the constraint (10), most of them involving the Jacobian of $N(\boldsymbol{\zeta},\boldsymbol{\mu})$ [13], [14], [15], [16]. This matrix is obtained by combining the partial derivatives of each network, N_1 and N_2, with the outputs of the other network according to (8):

$$\begin{aligned} & \tfrac{\partial}{\partial \zeta_i} N(\boldsymbol{\zeta},\boldsymbol{\mu}) &&= \tfrac{\partial}{\partial \zeta_i} N_1(\boldsymbol{\zeta}) \cdot N_2(\boldsymbol{\mu}) \\ \text{and} \quad & \tfrac{\partial}{\partial \mu_j} N(\boldsymbol{\zeta},\boldsymbol{\mu}) &&= N_1(\boldsymbol{\zeta}) \cdot \tfrac{\partial}{\partial \mu_j} N_2(\boldsymbol{\mu}). \end{aligned} \tag{12}$$

IV. Learning

In this section, we will omit for clarity the parameter $\tilde{\boldsymbol{\mu}}$ from K_1 and K_2. The learning of $K_1(\boldsymbol{\zeta})$ and $K_2(\boldsymbol{\mu})$ can be accomplished with strategies entailing different degrees of parallelism and sophistication. We show the two main ones below. It is important to point out that, in every case, we only require the ability to sense the pose of the end-effector in the chosen configuration $K(\boldsymbol{\zeta},\boldsymbol{\mu})$.

1) Independent learning: The simplest approach is to learn each function independently in a phase preceding the functional operation of the robot. The learning of K_1 and K_2, shown in Algorithms 1 and 2, proceeds sequentially. Algorithm 1 moves the first joints $\boldsymbol{\zeta}$ to random values while fixing $\boldsymbol{\mu}$ to a reference value. In Algorithm 2, a little trick is used to learn K_2. Normally, one should select an input $\boldsymbol{\mu}_i$ and then move to $K(\boldsymbol{\zeta}_i,\boldsymbol{\mu}_i)$ and $K(\boldsymbol{\zeta}_i,\tilde{\boldsymbol{\mu}})$ to obtain the desired output

$$K(\boldsymbol{\zeta}_i,\tilde{\boldsymbol{\mu}})^{-1} \cdot K(\boldsymbol{\zeta}_i,\boldsymbol{\mu}_i),$$

where $\boldsymbol{\zeta}_i$ is arbitrary in each iteration. But if $\boldsymbol{\zeta}_i$ remains always the same, $K(\boldsymbol{\zeta}_i,\tilde{\boldsymbol{\mu}})^{-1}$ is a constant that can be obtained before the loop, and one movement is saved in each iteration. In short, both Algorithms 1 and 2 consist basically in fixing some joints and moving the remaining ones.

There are many possible variations of Algorithm 2. If $\boldsymbol{\mu}$ is constrained for some values of $\boldsymbol{\zeta}$ (e.g., in order to keep the end-effector in the field of view), we can run Algorithm 2 several times with a different selection of $\hat{\boldsymbol{\zeta}}$. If the constraints require a different value of $\boldsymbol{\zeta}$ for each value of $\boldsymbol{\mu}_i$, it is still possible to learn K_2 with only one movement in each iteration. The selection of $\boldsymbol{\zeta}_i$ must be introduced in the loop (line 1 and 2 are removed), the movement must be performed to $(\boldsymbol{\zeta}_i,\boldsymbol{\mu}_i)$ and, finally, $N_1(\boldsymbol{\zeta}_i)^{-1}\, T_i$ must be used as output for N_2. This approximation follows from equation (10). The drawback is that these output data depend on an approximation of K_1. But since K_1 has a low dimensionality and it has been learned previously, the error introduced is negligible.

Algorithm 1: Learning of $K_1(\boldsymbol{\zeta})$.

1 **foreach** $\boldsymbol{\zeta}_i \in Training\_Set$ **do**
2 Move to $(\boldsymbol{\zeta}_i,\tilde{\boldsymbol{\mu}})$ and observe $T_i = K(\boldsymbol{\zeta}_i,\tilde{\boldsymbol{\mu}})$
3 Learn N_1 with $\boldsymbol{\zeta}_i$ as input and T_i as output.

Algorithm 2: Learning of $K_2(\boldsymbol{\mu})$.

1 Select $\hat{\boldsymbol{\zeta}}$
2 Move to $(\hat{\boldsymbol{\zeta}}, \tilde{\boldsymbol{\mu}})$ and observe $T_{\tilde{\boldsymbol{\mu}}} = K(\hat{\boldsymbol{\zeta}}, \tilde{\boldsymbol{\mu}})^{-1}$
3 **foreach** $\boldsymbol{\mu}_i \in Training\_Set$ **do**
4 Move to $(\hat{\boldsymbol{\zeta}}, \boldsymbol{\mu}_i)$ and observe $T_i = K(\hat{\boldsymbol{\zeta}}, \boldsymbol{\mu}_i)$
5 Learn N_2 with $\boldsymbol{\mu}_i$ as input and $T_{\tilde{\boldsymbol{\mu}}}\ T_i$ as output.

2) Concurrent learning: None of the learning strategies above can be used to perform on-line learning, that is, learning that is integrated in the normal working operation. The strategy that we present now parallelizes the learning of all the functions that compose the kinematic model. And, interestingly, it permits carrying out arbitrary movements as, for instance, those required by an application while, at the same time, refining the estimation of the robot kinematics.

In fact, equation (9) implicitly provides values for K_1, $(T \cdot K_2(\boldsymbol{\mu})^{-1})$, and for K_2, $(K_1(\boldsymbol{\zeta})^{-1} \cdot T)$, which depend on one another. It is possible to use their estimates N_1 and N_2 to obtain new training samples as it is shown in Algorithm 3.

Note that $\tilde{\boldsymbol{\mu}}$ is missing completely in Algorithm 3 and, thus, the algorithm can converge to functions with any value of $\tilde{\boldsymbol{\mu}}$. Moreover, the algorithm converges to whatever functions N_1 and N_2 satisfying

$$K(\boldsymbol{\zeta}, \boldsymbol{\mu}) = N_1(\boldsymbol{\zeta}) \cdot N_2(\boldsymbol{\mu}), \tag{13}$$

which, in general, would not have the shape of $K_1(\boldsymbol{\zeta}; \tilde{\boldsymbol{\mu}})$ and $K_2(\boldsymbol{\mu}; \tilde{\boldsymbol{\mu}})$ for any $\tilde{\boldsymbol{\mu}}$. But in Appendix A we show that, given an a priori fixed $\tilde{\boldsymbol{\mu}}$, after convergence N_1 and N_2 can be expressed as

$$N_1(\boldsymbol{\zeta}) = K_1(\boldsymbol{\zeta}; \tilde{\boldsymbol{\mu}})\ N_2(\tilde{\boldsymbol{\mu}})^{-1}, \tag{14}$$

$$N_2(\boldsymbol{\mu}) = N_2(\tilde{\boldsymbol{\mu}})\ K_2(\boldsymbol{\mu}; \tilde{\boldsymbol{\mu}}). \tag{15}$$

There is nothing wrong with these functions, since they constitute a valid composition. But it should be noted that N_1 and N_2 may change suddenly their values when switching from concurrent learning to independent learning. Anyway, a slight modification of Algorithms 1 and 2 would allow to learn the right parts of equations (14) and (15).

The fact that there are many functions yielding a valid kinematics decomposition has a potential advantage. N_1 (or N_2) alone can adapt to certain kinematic changes, absorbing the required changes for K_1 and K_2. This is interesting because learning only one function is much quicker than learning two interdependent functions. For example, if the kinematics of the robot undergoes a deformation equivalent to a linear transformation, that is,

$$K'(\boldsymbol{\zeta}, \boldsymbol{\mu}) = K(\boldsymbol{\zeta}, \boldsymbol{\mu}) \cdot P,$$

the system can be quickly adapted by only learning N_2, as shown in Appendix B. A linear transformation includes the rigid transformations involved in the adaptation to a tool and, also, some effects that result from a poorly calibrated camera such as a scaling of the sensor data.

Note that the learning of N_1 and N_2 is interdependent because, at each iteration, their corrections aim to reduce the same error quantity, $||N_1(\boldsymbol{\zeta}) \cdot N_2(\boldsymbol{\mu}) - T_i||$. To put the learning of N_1 and N_2 on an equal ground, in Algorithm 3, the desired outputs for both functions are calculated *before* any modification takes place. Anyway, special attention has to be payed to the learning rates used to learn N_1 and N_2. If, for instance, N_1 is corrected to make this error 0, a subsequent correction of N_2 of the same magnitude, will result in $N_1(\boldsymbol{\zeta}) \cdot N_2(\boldsymbol{\mu}) - T_i$ having a value opposite to the initial one, and the same error magnitude. Therefore, the learning rates should be such that, the correction of N_1 (or N_2) alone cancel out no more than half of the error or, in any case, the sum of the corrections to N_1 and N_2 must cancel out (partially or completely) $N_1(\boldsymbol{\zeta}) \cdot N_2(\boldsymbol{\mu}) - T_i$ without reverting its sign.

Algorithm 3: Simultaneous learning of $K_1(\boldsymbol{\zeta})$ and $K_2(\boldsymbol{\mu})$.

1 **foreach** $(\boldsymbol{\zeta}_i, \boldsymbol{\mu}_i) \in Training\_Set$ **do**
2 Move to $(\boldsymbol{\zeta}_i, \boldsymbol{\mu}_i)$ and observe $T_i = K(\boldsymbol{\zeta}_i, \boldsymbol{\mu}_i)$
3 Set $T_{i,1} := T_i \cdot N_2(\boldsymbol{\mu}_i)^{-1}$ and $T_{i,2} := N_1(\boldsymbol{\zeta}_i)^{-1} \cdot T_i$
4 Learn N_1 with $\boldsymbol{\zeta}_i$ as input and $T_{i,1}$ as output.
5 Learn N_2 with $\boldsymbol{\mu}_i$ as input and $T_{i,2}$ as output.

V. SIMULATIONS

We used two simulated robots having eight and twelve active DoF, respectively, in the offline learning simulation, and one robot of five DoF in the online learning simulations. The Denavit-Hartenberg parameters of these robots are equal for each segment i, namely $\alpha_i = 90°$, $a_i = 200\,\mathrm{mm}$ and $d_i = 0\,\mathrm{mm}$. This results in arm lengths of $2400\,\mathrm{mm}$, $1600\,\mathrm{mm}$ and $1000\,\mathrm{mm}$ at the rest positions. The samples used for training and testing are generated evaluating the FK in joint angles drawn from $[-45°, 45°]$. In all simulations, there are 1000 samples in the test sets that are generated randomly by sampling uniformly angles from this range. The actual learning is done in all cases by PSOM networks. The orientations of the end-effector are expressed by means of rotation matrices. Each of these matrices' elements are learned independently by the PSOM algorithm. As a result, the output may not always be a valid rotation matrix which can be critical when concatenating the individual networks' outputs. For this reason, a Gram-Schmidt orthonormalization is applied systematically to the rotational part of all networks to improve the output quality[3]. This includes also the orientation parts of N_1 and N_2 in line 3 of Algorithm 3. The calculus of the IK using the FK model will add a numerical error dependent on the algorithm used for this purpose. Because of this, all simulations in this paper focus on the evaluation of the accuracy of the FK representations.

A. Offline Learning

The first simulation examines the offline learning as presented in Algorithms 1 and 2. The kinematics of a robot with

[3]Note that even if one is only interested in learning positions, the orientation part of N_1 and N_2 is also involved in the calculation of the position of the composite kinematics.

eight independent and active degrees of freedom is learned by PSOM networks. The input values are fixed to the nodes of an eight-dimensional rectangular grid that encloses all possible joint angles of the training data. For learning, the output values of the forward kinematics at these joint positions are assigned to the corresponding neurons. Once learned, the PSOM interpolates between the learned pose values in order to estimate the forward kinematics. The number of neurons in each dimension of the grid was different in the PSOMs used in the simulation. They are indicated by the labels of selected data points (with a comma separating the grid dimensions of the two networks in the decomposition case) in Figures 2, 3 and 4.

Fig. 2 shows the mean error on the test data in relation to the number of samples (i.e., neurons) on a logarithmic scale. In this graph, one can directly see that—for higher numbers of neurons—the curves are nearly parallel to each other. The curve of the decomposition lies roughly in the middle between the axis of abscissas and the curve for the single network. This indicates that, in order to get the same level of accuracy, in comparison to the single network, only the square root of the number of samples is required to train the decomposition networks. In Figures 3 and 4, the same relation is shown on a linear scale. The most interesting part is amplified and plotted in Fig. 4. The mean error on the training data of the decomposition drops much quicker as compared to the single network. This point of view emphasizes the advantage of the decomposition when applied to a robot system. Figure 5 shows how many samples are necessary to obtain a certain level of precision. In the diagram, the 95%-quantiles for the decomposition and the single networks are displayed, that is, the precision threshold below which lie 95% of the errors on the test data. Again, a reduction to nearly the square root of the required samples can be appreciated thanks to the logarithmic scale.

We can confirm the visual intuition obtained in previous figures more rigorously. In the introduction, we have hypothesized that the number of samples required to learn a FK with n degrees of freedom is roughly q^n, with q determined by the precision and the workspace. Learning a FK in the same workspace and with the same precision using a decomposition into d kinematic chains requires to learn d FK functions with n/d degrees of freedom. Thus, if the hypothesis is true, learning with the decomposition framework requires $d \cdot q^{n/d}$ samples. In particular, for a two-chain decomposition, if n_s and n_d are the number of samples to reach a fixed precision with the single model and the decomposition, respectively, then holds $n_s \approx q^n$ and $n_d \approx 2 \cdot q^{n/2} \approx 2 \cdot \sqrt{n_s}$ and

$$\frac{\ln(n_s)}{\ln(n_d/2)} \approx \frac{\ln(n_s)}{\ln(\sqrt{n_s})} = 2. \tag{16}$$

Table I shows the high degree of accuracy of the hypothesis for the experimental data. In the last simulation of this section, we use the capability of the decomposition approach to be applied recursively. The subchains resulting from the recursive decomposition are shorter than those using a single decomposition. This makes affordable the learning of hyper redundant kinematic chains. For this experiment, we have used

Precision [mm]	n_s	n_d	$\frac{ln(n_s)}{ln(n_d/2)}$
1,100	1	2	
760	16	10	1.72
520	576	32	2.29
390	864	40	2.26
270	1,296	52	2.2
140	2,916	105	2.01
40	8,748	162	2.07
30	11,664	225	1.98
20	20,736	300	1.98
10	36,864	400	1.98
4.2	65,536	512	2.00
1.3	200,000	945	1.98
0.3	390,625	1,250	2.00

TABLE I
COMPARISON OF THE NUMBER OF SAMPLES NECESSARY TO APPROXIMATELY REACH A GIVEN PRECISION USING A SINGLE PSOM (n_s) VERSUS THE DECOMPOSITION (n_d). AN ADDITIONAL COLUMN SHOWS HOW WELL THE DATA FITS THE SQUARE ROOT HYPOTHESIS (SEE TEXT) MEASURED BY ITS CLOSENESS TO 2.

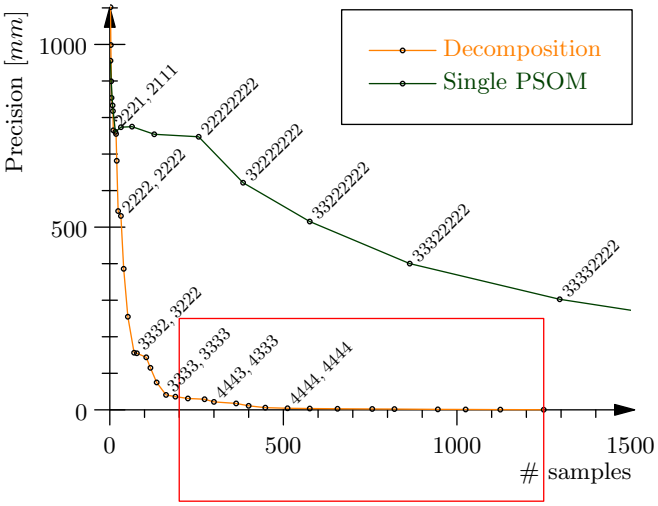

Fig. 3. Performance of the decomposition and a single PSOM when learning offline shown on a linear scale. In Fig. 4, The highlighted area is shown enlarged.

a robot arm with twelve independent degrees of freedom. The kinematics of this robot is first learnt with a single PSOM in a standard offline way. Then a recursive decomposition with three PSOMs is also tested. The robot is first decomposed into two subchains of four and eight DoFs, and this last one is again decomposed into two equally sized subchains. Therefore, three subchains of length four are learned with this recursive decomposition. The result is shown in Fig. 6 in a logarithmic scale in the number of samples. It can be observed that the precision obtained by the single PSOM with 10^6 samples is reached through the triple decomposition with only 110 samples. As expected, the gains obtained here are notoriously larger than those obtained in the previous simulation showed in Fig. 2.

B. Online Learning

Now we investigate how learning and the refinement of the decomposition can be performed during the normal operation

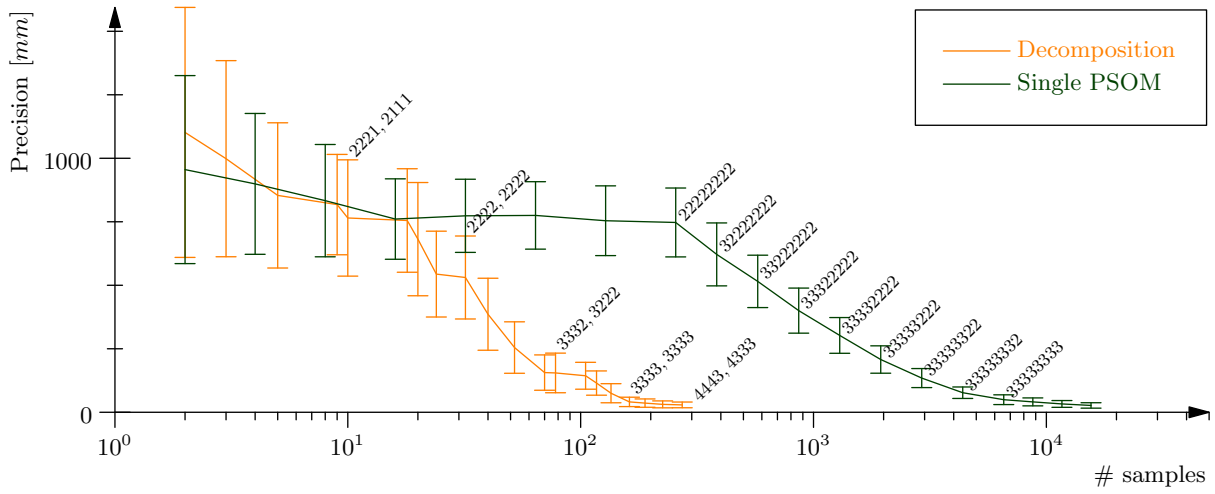

Fig. 2. Comparison of the offline learning using the decomposition and a single PSOM with different training samples and, consequently, different numbers of neurons as indicated by the labels. The diagram uses a logarithmic scale and the standard deviation of the precision is included in form of error bars.

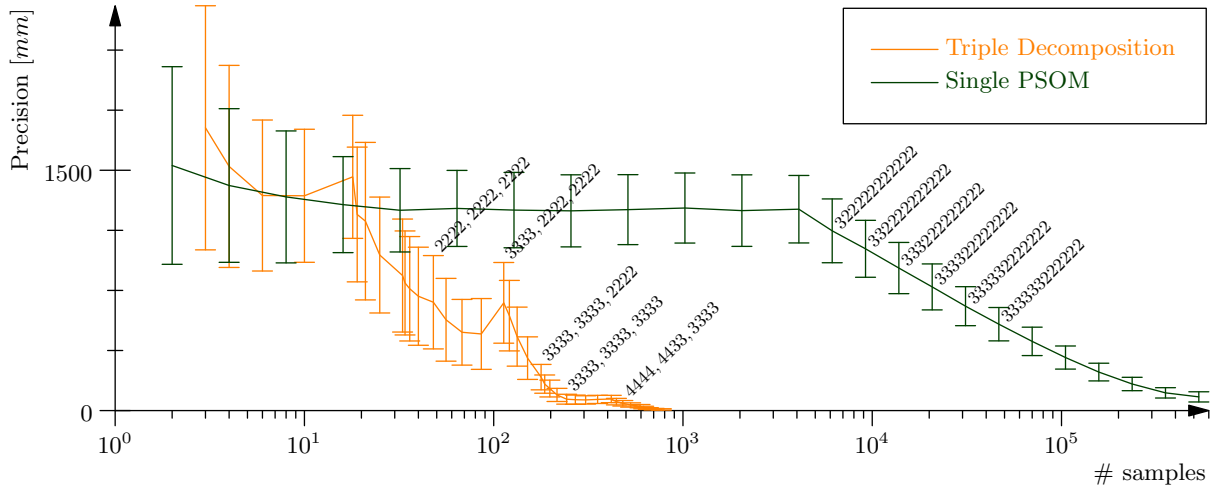

Fig. 6. Learning of a hyper-redundant kinematics of 12 DoF with a recursive decomposition that simultaneously applies three PSOM learning instances. The results are compared to those of a single PSOM learner and displayed on a scale that is logarithmic in the number of training samples.

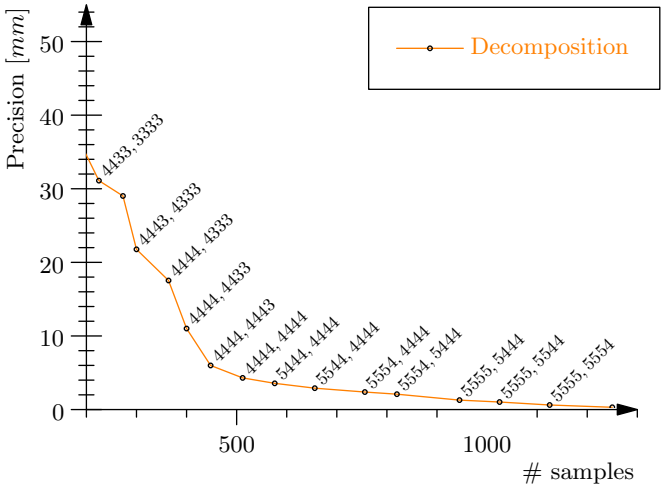

Fig. 4. Closeup showing the number of training samples needed to achieve a high precision on a robot with eight DoF.

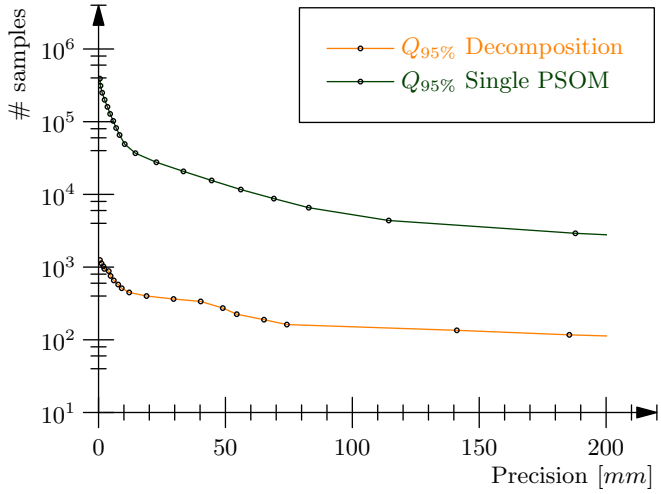

Fig. 5. Convergence of the decomposition and a single PSOM for higher precision. On the logarithmic scale it can be seen that, using the decomposition, the number of training samples required to obtain a given precision is roughly reduced to its square root.

of the robot using Algorithm 3. As the regular PSOM algorithm requires grid-organized data, it is not naturally suited for online learning. Here, we have carried out a grid-preserving supervised adaptation by updating part of the weight of the neurons according to the Widrow-Hoff rule or *normalized least mean squares (NLMS)* method (equivalent to δ-rule for single layer preceptrons):

$$w_{\boldsymbol{a}}^{t+1} = w_{\boldsymbol{a}}^{t} + \epsilon \cdot H_{\boldsymbol{a}}(\boldsymbol{a}, \boldsymbol{\theta}) \cdot (w_{\boldsymbol{a}}^{t} - \boldsymbol{x}), \tag{17}$$

where $w_{\boldsymbol{a}}^{t}$ is the weight subvector of the neuron at grid position $\boldsymbol{a}$ representing the robot pose, $\epsilon \in (0, 1]$ is the learning rate, and $(\boldsymbol{\theta}, \boldsymbol{x})$ is a sample input/output pair. If the learning rate ϵ in equation (17) equals one, the network adapts completely to the currently presented sample, that is, the output of the network then equals $\boldsymbol{x}$. Please note that a variant of PSOM online learning was presented in [17]. However, this method requires to search for a winning neuron in each step and turned out to be generally less suited for the experiments.

According to the discussion at the end of Section IV-2, the learning rates for N_1 and N_2 have been set to 0.5, which adapts completely the combination of the two networks to the presented sample. In this way, the two networks cancel out the same amount of error.

This online learning initially adapts very fast to modifications of the kinematics. In the long term, however, this way of learning is much slower compared to offline learning, that is, a much higher number of samples is required to gain the same level of precision. For this reason, we have reduced the number of effective degrees of freedom to five in this simulation.

In this section, we investigate how the decomposition of a robot with five revolute joints adapts to two modifications that are likely to occur in real application. Training and test samples are generated with the modified robot by moving to random configurations with angles out of the same angular range as during the initial training (i.e., $[-45°, 45°]$). The refinement starts with initial models that are approximations of the intact robot FK consisting of a single PSOM with $5^5 = 3125$ neurons and a decomposition with $5^3 + 5^2 = 150$ neurons.

The first modification of the kinematics is a translation of $400\,\mathrm{mm}$ applied to the end-effector in order to simulate tool use. Another kind of modification can occur with incremental encoders that require calibration upon each startup. We simulated a modification of this type, by adding a constant of $10°$ to all robot joints. The results of learning after these two deformations have taken place are presented in the diagrams in Fig. 7 and Fig. 9, respectively. In both diagrams, it can be immediately seen that the decomposition leads to better levels of accuracy much more quickly as compared to the single network. Note also that the error bars of the single PSOM curve remain in both figures almost constant, while in that of the decomposition they shrink notoriously. Adaptation for the first training samples is very fast and afterwards the curves converge to the optimal solution even though slowly. For the first deformation, we further investigated if learning can be accelerated by adapting only one of the individual networks N_1 and N_2 (see Fig. 8). One can see that only the second network N_2 is able to compensate the deformation and, as

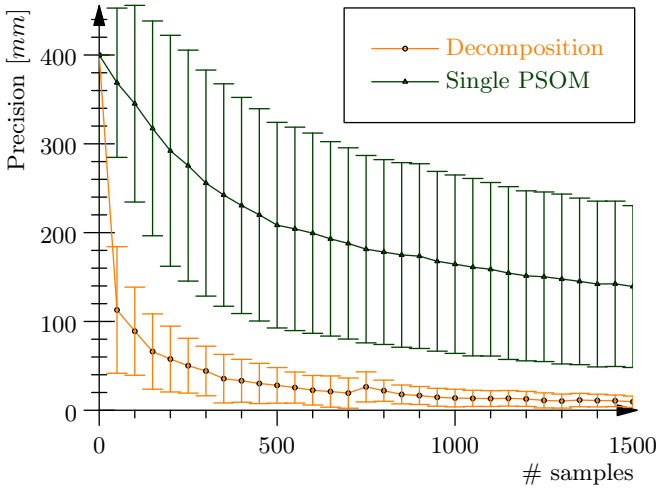

Fig. 7. Learning curves of the new incremental online learning algorithm after a deformation simulating tool use (last element extended 400 mm). Leaning begins from models of the original kinematics learned offline. The Standard deviations are shown as error bars.

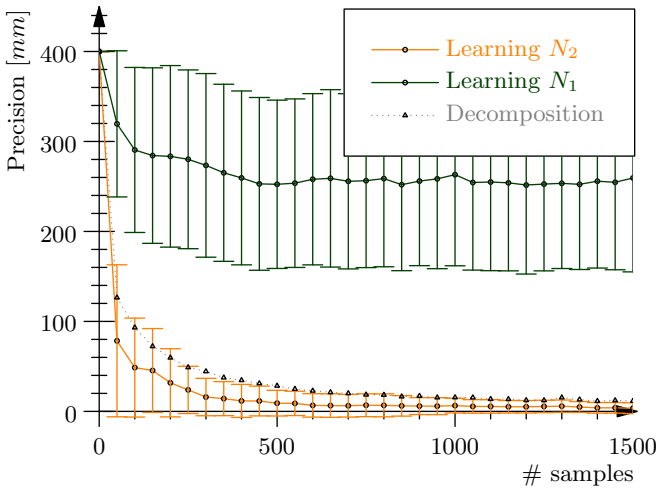

Fig. 8. This image shows the performance of learning only one of the networks in the decomposition after the same deformation as in Fig. 7).

a matter of fact, it does significantly quicker than learning simultaneously both functions: the error reached after learning 500 samples with N_2 alone is lower than that obtained after adapting to learn 1500 samples both networks. Consequently, this learning strategy is useful to learn deformations known to be linear transformations of the original kinematics. The most prominent example in this context is tool-use.

C. Alternative Learners

The decomposition scheme breaks long kinematic chains down into smaller but still valid kinematic functions. Consequently, the decomposition can be combined with any machine learning technique that is suitable for learning kinematics. In order to show this property, simulations with *Gaussian Mixture Models* (GMM) and *Gaussian Mixture Regression* [6], [18] will be presented in this section. GMM are a prominent knowledge representation in robotics where they recently are mostly used for learning of trajectories [19] and imitation [20].

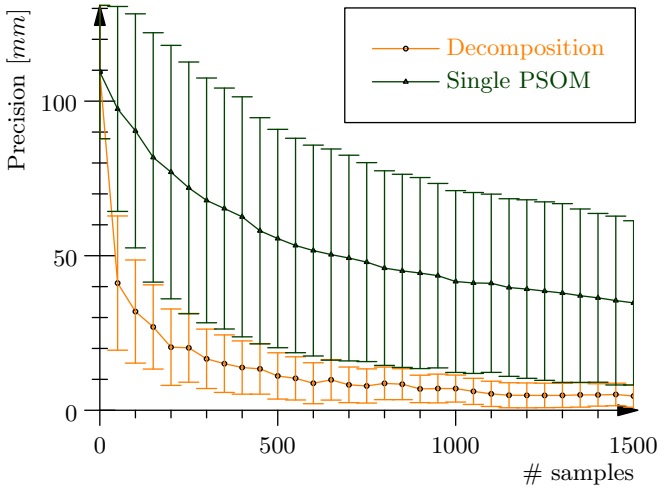

Fig. 9. Learning curves of the new incremental online learning algorithm when suddenly a constant of 10° is added to each angle.

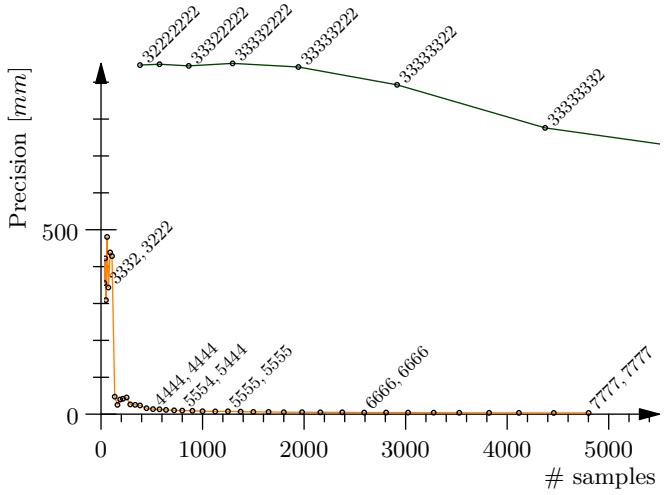

Fig. 10. Result of the decomposition when *Gaussian Mixture Models* are used as learning algorithm. It can be seen that using the decomposition a comparable speed-up as in the previous experiments is attained.

However, they are also well suited to learn a direct model of a robot kinematics [21].

GMM store the learned knowledge in form of the combination of a number of probability density functions. Obtaining the parameters of the models can be done via the *expectation maximization* algorithm. Once the GMM have been trained, the *Gaussian Mixture Regression* algorithm can be used to find missing components of a query vector, that is, to solve direct or inverse problems (*generalization* see [6]). Thus, GMM are very similar to PSOM w.r.t. the flexible way in which they can be queried.

The simulation uses two GMM in the decomposition approach and, again, a single instance learns the complete chain for comparison. Although not required by GMM, but in order to make this experiment more similar to those in Sec. V-A, data points are arranged in a regular, PSOM-like grid. The expectation maximization algorithm is used to optimize the parameters of the gaussian models whereas PSOM interpolates data points directly. Consequently, learning with PSOM can be much faster, while GMM are very tolerant to noise and outliers and store knowledge in a very compact form (i.e., they do not need to store each data point). In contrast, GMM interpolate less accurately with noise-free points (at least in this application), which obliged us to halve the length a_i of each robot link i and reduce the movement of all joints to $[-22.5°, 22.5°]$ to get a meaningful comparison with respect to other simulations. Otherwise, the experiment is performed under very similar conditions to those in Sec. V-A. The only parameter that has to be determined manually dependingon the application is the actual number of gaussian mixtures. We have optimized this number for a large number of samples and found that this optimum is approximately proportional to the number of samples. The results are presented in Fig. 10. It can be clearly seen that the speed-up provided by the decomposition in Fig. 3 with a PSOM is similarly obtained when learning with GMM.

VI. Conclusion

In this paper, we pointed out the importance of modeling kinematics functions by means of machine learning techniques. The main difficulty, hereby, lies in the fact that the number of training samples required to acquire an adequately accurate model grows exponentially with the number of degrees of freedom. Decomposition techniques have proved to be an effective means to solve this problem by reducing the amount of training samples to about twice its square root (in the case of one single decomposition). However, the decomposition schemes presented in previous works either impose restrictions to the kinematics (e.g., three intersecting axes) or require more parts of the robot to be visible, increasing the demand for additional sensors.

This work presents a new strategy to learn a decomposition that overcomes these restrictions. The kinematic function is split up into two dependent sub-functions that can either be learned offline—one after another—or can be simultaneously refined in an incremental online learning process. The theoretical insights were verified using several simulated robots with twelve, eight and five active revolute degrees of freedom, respectively. We chose the parameterized self-organizing maps (PSOM) as the underlying machine learning algorithm and further enhanced it by incorporating a supervised incremental learning rule—namely the Widrow-Hoff rule. In a series of simulations, we demonstrated that the learning was sped up drastically (i.e., the number of required training samples was reduced to its square root) as predicted and we showed the relation between learning speed and the resulting model precision. Moreover, we showed the scalability of the approach by applying the decomposition recursively to a robot with 12 DoF.

In further simulations, we showed that the decomposition can enormously speed up the convergence of the online refinement of initial models, for example in the case of tool-use or while recovering from a shift in the joint encoders. Altogether, the combination of both learning methods—creating an initial model, in simulation for instance, and refining it online afterwards—leads to a very efficient method to learn

the complete kinematics of even very complex robots with many active degrees of freedom. The new decomposition is compatible with most of the algorithms devised to learn FK [22], [23], [24], and we have shown that a similar speed-up to that obtained with PSOM is obtained with GMM/GMR as well. Furthermore, the decomposition can make the use of a learned FK affordable to those approaches using a known FK to obtain IK information [25], [26], [16].

The presented approach offers gains similar to those obtained with the previous ones in [8] and [11], because they all rely on approximating the kinematics of chains having half of the number of joints of the robot. However, there are more criteria to be evaluated in the comparison of these approaches. The approach in [8] is a complex decomposition in which four functions are involved. This decomposition can only be applied to a robot whose position and orientation are uncoupled by having its last three joint axes crossing at a point. Instead, the decomposition presented here involves only two functions and, more importantly, it can be applied to any serial robot. The basic idea in [11] is to learn the kinematics of two subchains of the robot, one from the base to a marker on an intermediate link, and another one from the marker to the end-effector. Thus, the reference frames of the marker in the intermediate point and of the end-effector must be provided by the sensory system, a task that can be seriously hindered by auto-occlusions. If one wants learning to be more efficient by dividing the robot into three subchains, three reference frames in the robot must be collected. The advantage of the new decomposition over [11] is that only the reference frame of the end-effector is needed in any case. If only offline learning is required, the new decomposition must therefore be preferred to [11]. If on-line is required, it is somewhat simpler and quicker in [11], because the learned functions are not interdependent, but this must be counterbalanced with the added sensorial requirements. Our future plans include the application of this decomposition technique to the ARMAR humanoid robot [10].

APPENDIX

A. Functions satisfying the decomposition

We will prove that all functions N_1, N_2 satisfying the composition equation (13) used in Algorithm 3, have the form

$$\begin{array}{rcl} N_1(\boldsymbol{\zeta}) & = & K_1(\boldsymbol{\zeta};\tilde{\boldsymbol{\mu}})\cdot C^{-1} \\ N_2(\boldsymbol{\mu}) & = & C\cdot K_2(\boldsymbol{\mu};\tilde{\boldsymbol{\mu}}), \end{array} \tag{18}$$

where C is equal to $N_2(\tilde{\boldsymbol{\mu}})$.

First, we show that functions of the same form as (18) do in fact satisfy (13),

$$\begin{array}{rcl} N_1(\boldsymbol{\zeta})\cdot N_2(\boldsymbol{\mu}) & = & K_1(\boldsymbol{\zeta};\tilde{\boldsymbol{\mu}})\cdot C^{-1}\cdot C\cdot K_2(\boldsymbol{\mu};\tilde{\boldsymbol{\mu}}) \\ & = & K_1(\boldsymbol{\zeta};\tilde{\boldsymbol{\mu}})\cdot K_2(\boldsymbol{\mu};\tilde{\boldsymbol{\mu}}) \\ & = & K(\boldsymbol{\zeta},\boldsymbol{\mu}), \end{array} \tag{19}$$

and that, given that form, C must equal $N_2(\tilde{\boldsymbol{\mu}})$:

$$N_2(\tilde{\boldsymbol{\mu}}) = C\cdot K_2(\tilde{\boldsymbol{\mu}};\tilde{\boldsymbol{\mu}}) = C\cdot I = C, \tag{20}$$

where I is the identity matrix. Now, we show that no form other than (18) is possible for N_1, N_2. We begin by defining the functions

$$\begin{array}{rcl} \epsilon_1(\boldsymbol{\zeta}) & \equiv & K_1(\boldsymbol{\zeta};\tilde{\boldsymbol{\mu}})^{-1}\cdot N_1(\boldsymbol{\zeta}) \\ \epsilon_2(\boldsymbol{\mu}) & \equiv & N_2(\boldsymbol{\mu})\cdot K_2(\boldsymbol{\mu};\tilde{\boldsymbol{\mu}})^{-1}. \end{array} \tag{21}$$

Note that these functions always exist, because K_1 and K_2 are rigid transformations, and thus invertible. Multiplying ϵ_1 and ϵ_2:

$$\epsilon_1(\boldsymbol{\zeta})\cdot\epsilon_2(\boldsymbol{\mu}) = K_1(\boldsymbol{\zeta};\tilde{\boldsymbol{\mu}})^{-1}\cdot N_1(\boldsymbol{\zeta})\cdot N_2(\boldsymbol{\mu})\cdot K_2(\boldsymbol{\mu};\tilde{\boldsymbol{\mu}})^{-1},$$

using the composition equation (13) that N_1 and N_2 are assumed to satisfy,

$$\epsilon_1(\boldsymbol{\zeta})\cdot\epsilon_2(\boldsymbol{\mu}) = K_1(\boldsymbol{\zeta};\tilde{\boldsymbol{\mu}})^{-1}\cdot K(\boldsymbol{\zeta},\boldsymbol{\mu})\cdot K_2(\boldsymbol{\mu};\tilde{\boldsymbol{\mu}})^{-1},$$

and applying (4) and (2),

$$\epsilon_1(\boldsymbol{\zeta})\cdot\epsilon_2(\boldsymbol{\mu}) = K_1(\boldsymbol{\zeta};\tilde{\boldsymbol{\mu}})^{-1}\cdot K_1(\boldsymbol{\zeta};\tilde{\boldsymbol{\mu}}),$$

we obtain:

$$\epsilon_1(\boldsymbol{\zeta})\cdot\epsilon_2(\boldsymbol{\mu}) = I. \tag{22}$$

Since ϵ_1 and ϵ_2 are functions dependent on different variables, they cannot cancel out the variable dependency of each other by means of multiplication. The only way of satisfying (22) is having $\epsilon_1 = C^{-1}$ and $\epsilon_2 = C$ for some constant C. Substituting ϵ_1 and ϵ_2 by these constants in (21),

$$\begin{array}{rcl} C^{-1} & = & K_1(\boldsymbol{\zeta};\tilde{\boldsymbol{\mu}})^{-1}\cdot N_1(\boldsymbol{\zeta}) \\ C & = & N_2(\boldsymbol{\mu})\cdot K_2(\boldsymbol{\mu};\tilde{\boldsymbol{\mu}})^{-1}, \end{array} \tag{23}$$

yielding that (18) is the only form that N_1 and N_2 can exhibit.

We have demonstrated that all possible decompositions build by multiplying two functions of the two subsets of joints are the same up to a constant. This is the case for functions K_1 and K_2 with different reference values, $\tilde{\boldsymbol{\mu}}$ and $\tilde{\boldsymbol{\mu}}'$:

$$\begin{array}{rcl} K_1(\boldsymbol{\zeta};\tilde{\boldsymbol{\mu}}) & = & K_1(\boldsymbol{\zeta};\tilde{\boldsymbol{\mu}}')\cdot C^{-1}_{\tilde{\boldsymbol{\mu}}'}\cdot C_{\tilde{\boldsymbol{\mu}}} \\ K_2(\boldsymbol{\mu};\tilde{\boldsymbol{\mu}}) & = & C^{-1}_{\tilde{\boldsymbol{\mu}}}\cdot C_{\tilde{\boldsymbol{\mu}}'}\cdot K_2(\boldsymbol{\mu};\tilde{\boldsymbol{\mu}}'). \end{array} \tag{24}$$

These relations are deduced from (3) and (7), respectively.

The result applies also to the alternative decomposition mentioned in Section II,

$$K(\boldsymbol{\zeta},\boldsymbol{\mu}) = L_2(\boldsymbol{\zeta})\cdot L_1(\boldsymbol{\mu}),$$

for which it can be shown that

$$\begin{array}{rcl} L_2(\boldsymbol{\zeta}) & = & K_1(\boldsymbol{\zeta};\tilde{\boldsymbol{\mu}})\cdot C \\ \text{and} \qquad L_1(\boldsymbol{\mu})^{-1} & = & C^{-1}\cdot K_2(\boldsymbol{\mu}). \end{array}$$

B. Deformations learnable with only one function

When the learning of N_2 is removed from Algorithm 3 (i.e., only N_1 is learned), it is still possible to adapt the composition to certain deformations. Let K' denote the new deformed kinematics and let K'_1 and K'_2 be the new component functions for the chosen $\tilde{\boldsymbol{\mu}}$. All deformations for which there exists a constant C satisfying

$$K'(\boldsymbol{\zeta},\boldsymbol{\mu})\cdot N_2(\boldsymbol{\mu})^{-1} = K'_1(\boldsymbol{\zeta};\tilde{\boldsymbol{\mu}})\cdot C \tag{25}$$

can be learned by N_1 alone. The left side of the equation is the function learned by N_1 in Algorithm 3 when N_2 is fixed. The right side is the form of the functions that N_1 is allowed to encode to yield a valid composition. If N_2 is assumed to be correctly learned before the deformation (i.e., $N_2(\boldsymbol{\mu}) = C_{old} \cdot K_2(\boldsymbol{\mu}; \tilde{\boldsymbol{\mu}})$), a simpler condition can be stated:

$$K_2'(\boldsymbol{\mu}; \tilde{\boldsymbol{\mu}}) = C \cdot K_2(\boldsymbol{\mu}; \tilde{\boldsymbol{\mu}}) \tag{26}$$

In fact, using the assumption, it is easy to prove that (26) implies (25):

$$\begin{aligned} K'(\boldsymbol{\zeta}, \boldsymbol{\mu}) \cdot N_2(\boldsymbol{\mu})^{-1} &= K'(\boldsymbol{\zeta}, \boldsymbol{\mu}) \cdot (C_{old} \cdot K_2(\boldsymbol{\mu}; \tilde{\boldsymbol{\mu}}))^{-1} \\ &= K'(\boldsymbol{\zeta}, \boldsymbol{\mu}) \cdot K_2(\boldsymbol{\mu}; \tilde{\boldsymbol{\mu}})^{-1} \cdot C_{old}^{-1} \\ &= K'(\boldsymbol{\zeta}, \boldsymbol{\mu}) \cdot \\ &\quad (C^{-1} \cdot K_2'(\boldsymbol{\mu}; \tilde{\boldsymbol{\mu}}))^{-1} \cdot C_{old}^{-1} \\ &= K'(\boldsymbol{\zeta}, \boldsymbol{\mu}) \cdot K_2'(\boldsymbol{\mu}; \tilde{\boldsymbol{\mu}})^{-1} \cdot C \cdot C_{old}^{-1} \\ &= K_1'(\boldsymbol{\zeta}; \tilde{\boldsymbol{\mu}}) \cdot C \cdot C_{old}^{-1}. \end{aligned}$$

The condition equivalent to (26) for the case of learning N_2 alone is that

$$K_1'(\boldsymbol{\zeta}; \tilde{\boldsymbol{\mu}}) = K_1(\boldsymbol{\zeta}; \tilde{\boldsymbol{\mu}}) \cdot C \tag{27}$$

for some C.

Now it is easy to see that if the kinematics of the robot undergoes a deformation equivalent to a linear transformation, that is,

$$K'(\boldsymbol{\zeta}, \boldsymbol{\mu}) = K(\boldsymbol{\zeta}, \boldsymbol{\mu}) \cdot P,$$

the system can be quickly adapted by learning N_2 alone. A linear transformation includes rigid transformations, as those involved in adaptation to a tool. And also includes some camera miscalibrations leading for example to a scaling of the sensor data. In effect, since

$$\begin{aligned} K_1'(\boldsymbol{\zeta}; \tilde{\boldsymbol{\mu}}) &= K'(\boldsymbol{\zeta}, \tilde{\boldsymbol{\mu}}) = K(\boldsymbol{\zeta}, \tilde{\boldsymbol{\mu}}) \cdot P & (28) \\ &= K_1(\boldsymbol{\zeta}; \tilde{\boldsymbol{\mu}}) \cdot P, & (29) \end{aligned}$$

condition (27) is fulfilled. Instead, learning N_1 alone does not work. Using (6),

$$\begin{aligned} K_2'(\boldsymbol{\mu}; \tilde{\boldsymbol{\mu}}) &= K'(\boldsymbol{\zeta}, \boldsymbol{\mu})^{-1} \cdot K'(\boldsymbol{\zeta}, \tilde{\boldsymbol{\mu}}) & (30) \\ &= (K(\boldsymbol{\zeta}, \tilde{\boldsymbol{\mu}}) \cdot P)^{-1} \cdot K(\boldsymbol{\zeta}, \boldsymbol{\mu}) \cdot P & (31) \\ &= P^{-1} \cdot K(\boldsymbol{\zeta}, \tilde{\boldsymbol{\mu}})^{-1} \cdot K(\boldsymbol{\zeta}, \boldsymbol{\mu})\ P, & (32) \end{aligned}$$

and using again (6),

$$K_2'(\boldsymbol{\mu}; \tilde{\boldsymbol{\mu}}) = P^{-1} \cdot K_2(\boldsymbol{\mu}; \tilde{\boldsymbol{\mu}}) \cdot P. \tag{33}$$

There is no possibility to satisfy (26), except for the case when $P = I$.

REFERENCES

[1] Y. Zhao and C. C. Cheah, "Neural network control of multifingered robot hands using visual feedback," *IEEE Trans. Neural Netw.*, vol. 20, no. 5, pp. 758–767, May 2009.

[2] J. Walter and H. Ritter, "Rapid learning with parametrized self-organizing maps," *Neurocomputing*, vol. 12, no. 2-3, pp. 131–153, 1996.

[3] C. Nölker and H. Ritter, "Visual recognition of continuous hand postures," *IEEE Trans. Neural Netw.*, vol. 13, no. 4, pp. 983–994, 2002.

[4] V. Ruiz de Angulo and C. Torras, "Self-calibration of a space robot," *IEEE Trans. Neural Netw.*, vol. 8, no. 4, pp. 951–963, 1997.

[5] S. Vijayakumar, A. D'Souza, and S. Schaal, "Incremental online learning in high dimensions," *Neural Computation*, vol. 17, pp. 2602–2634, 2005.

[6] S. Calinon, F. Guenter, and A. Billard, "On Learning, Representing and Generalizing a Task in a Humanoid Robot," *IEEE Trans. Syst., Man, Cybern. B*, vol. 37, no. 2, pp. 286–298, 2007.

[7] C. Constantinopoulos and A. Likas, "Unsupervised Learning of Gaussian Mixtures Based on Variational Component Splitting," *IEEE Trans. Neural Netw.*, vol. 18, no. 3, pp. 745–755, 2007.

[8] V. Ruiz de Angulo and C. Torras, "Speeding up the learning of robot kinematics through function decomposition," *IEEE Trans. Neural Netw.*, vol. 16, no. 6, pp. 1504–1512, 2005.

[9] G. Metta, G. Sandini, G. S, and L. Natale, "Sensorimotor interaction in a developing robot," in *1st Int. Workshop Epigenetic Robotics: Modeling Cognitive Development in Robotic Systems*. Lund University Press, 2001, pp. 18–19.

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

[11] V. Ruiz de Angulo and C. Torras, "Learning inverse kinematics: Reduced sampling through decomposition into virtual robots," *IEEE Trans. Syst., Man, Cybern. B*, vol. 38, no. 6, pp. 1571–1577, 2008.

[12] B. Widrow and M. E. Hoff, "adaptive switching circuits," in *1960 IRE WESCON Convention Record, Part 4*. New York: IRE, 1960, pp. 96–104.

[13] D. Whitney, "Resolved motion rate control of manipulators and human prostheses," *IEEE Trans. Man-Mach. Syst.*, vol. 10, no. 2, pp. 47–53, June 1969.

[14] A. Ligeois, "Automatic supervisory control of the configuration and behavior of multibody mechanisms," *IEEE Trans. Syst., Man, Cybern.*, vol. 7, no. 12, pp. 868–871, Dec. 1977.

[15] L. Li, W. Gruver, Q. Zhang, and Z. Yang, "Kinematic control of redundant robots and the motion optimizability measure," *IEEE Trans. Syst., Man, Cybern. B*, vol. 31, no. 1, pp. 155–160, Feb. 2001.

[16] Y. Xia, G. Feng, and J. Wang, "A primal-dual neural network for online resolving constrained kinematic redundancy in robot motion control," *IEEE Trans. Syst., Man, Cybern. B*, vol. 35, no. 1, pp. 54–64, Feb. 2005.

[17] J. Walter, "Rapid learning in robotics," Ph.D. dissertation, Technische Fakultät, Universität Bielefeld, 1996.

[18] D. Gu, "Distributed EM Algorithm for Gaussian Mixtures in Sensor Networks," *IEEE Trans. Neural Netw.*, vol. 19, no. 7, pp. 1154 –1166, july 2008.

[19] S. Calinon, F. D'halluin, E. Sauser, D. Caldwell, and A. Billard, "Learning and reproduction of gestures by imitation: An approach based on Hidden Markov Model and Gaussian Mixture Regression," *IEEE Robot. Autom. Mag.*, vol. 17, no. 2, pp. 44–54, 2010.

[20] D. B. Grimes, D. R. Rashid, and R. P. N. Rao, "Learning nonparametric models for probabilistic imitation," in *Advances in Neural Information Processing Systems (NIPS)*. MIT Press, 2006, pp. 521–528.

[21] M. Lopes and B. Damas, "A Learning Framework for Generic Sensory-Motor Maps," in *Int. Conf. Intelligent Robotic Systems (IROS)*, San Diego, USA, 2007, pp. 1533–1538.

[22] 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.

[23] R. Martinez-Cantin, M. Lopes, and L. Montesano, "Body schema acquisition through active learning," in *Proc. IEEE Int. Conf. Robotics ad Automation*, 2010, pp. 1860–1866.

[24] G. Sun and B. Scassellati, "Reaching through learned forward model," in *Proc. IEEE-RAS Int. Conf. Humanoid Robots*, vol. 1, Nov. 2004, pp. 93–112.

[25] M. Rolf, J. Steil, and M. Gienger, "Goal babbling permits direct learning of inverse kinematics," *IEEE Trans. Auton. Mental Develop.*, vol. 2, no. 3, pp. 216–229, 2010.

[26] Y. Zhang and S. Ge, "Design and analysis of a general recurrent neural network model for time-varying matrix inversion," *IEEE Trans. Neural Netw.*, vol. 16, no. 6, pp. 1477–1490, Nov. 2005.

Stefan Ulbrich received the diploma degree in computer science from the University of Karlsruhe (TH), Germany in 2007. He is currently a Ph.D. student at the Karlsruhe Institute of Technology (KIT) where he is a member of the Humanoids Research Group. His major research interests are kinematics learning, body schema learning, robot modeling and simulation, benchmarking tools for grasping and manipulation, as well as marker-based human motion tracking

Vicente Ruiz de Angulo was born in Miranda de Ebro, Burgos, Spain. He received the B.Sc. and Ph.D. degrees in computer science from the Universidad del País Vasco, Bilbao, Spain. During the academic year 1988–1989, he was an Assistant Professor with the Universitat Politècnica de Catalunya, Barcelona, Spain. In 1990, he was with the Neural Network Laboratory, Joint Research Center of the European Union, Ispra, Italy. From 1995 to 1996, he was with the Institut de Cibernètica, Barcelona, participating in the ESPRIT project entitled "Robot Control Based on Neural Network Systems" (CONNY). He also spent six months with the Istituto Dalle Molle di Studi Sull' Inteligenza Articiale di Lugano, working in applications of neural networks to robotics. Since 1996, he has been with the Institut de Robòtica i Informàtica Industrial (CSIC–UPC), Barcelona. His interests in neural networks include fault tolerance, noisy and missing data processing, and their application to robotics and computer vision.

Carme Torras (M'07, SM'11) is Research Professor at the Spanish Scientific Research Council (CSIC). She received M.Sc. degrees in Mathematics and Computer Science from the Universitat de Barcelona and the University of Massachusetts, Amherst, respectively, and a Ph.D. degree in Computer Science from the Technical University of Catalonia (UPC). Prof. Torras has published five books and about two hundred papers in the areas of robot kinematics, geometric reasoning, computer vision, and neurocomputing. She has been local project leader of several European projects, such as "Planning RObot Motion" (PROMotion), "Robot Control based on Neural Network Systems" (CONNY),"Self-organization and Analogical Modelling using Subsymbolic Computing" (SUBSYM), "Behavioural Learning: Sensing and Acting" (B-LEARN), "Perception, Action and COgnition through Learning of Object-Action Complexes" (PACO-PLUS), and the ongoing 7th framework projects "GARdeNIng with a Cognitive System" (GARNICS) and "Intelligent observation and execution of Actions and manipulations" (IntellAct). Prof. Torras is an Associate Editor of the IEEE Transactions on Robotics.

Tamim Asfour is a senior research scientist and leader of the Humanoid Research Group at Humanoids and Intelligence Systems Lab, Institute for Anthropomatics, Karlsruhe Institute of Technology (KIT). In 2003, he was awarded with the Research Center for Information Technology (FZI) prize for his outstanding Ph.D. thesis on sensorimotor control in humanoid robotics and the development of the humanoid robot ARMAR. His major research interest is humanoid robotics.

Rüdiger Dillmann is full professor at the Karlsruhe Institute of Technology (KIT), Department of Informatics, and head of the Humanoids and Intelligence Systems Laboratory at the Institute for Anthropomatics. His major research interests include humanoid robotics and human-centered robotics, robot programming by demonstration, machine learning, robot vision, cognitive cars, service robots, and medical applications of informatics.