Population based Mean of Multiple Computations Networks: A Building Block for Kinematic Models

Manuel Baum and Martin Meier and Malte Schilling
Center of Excellence 'Cognitive Interaction Technology' (CITEC),
Bielefeld University, P.O. Box 10 01 31, D-33501 Bielefeld, Germany
Email: mschilli@techfak.uni-bielefeld.de

***Abstract*—Population based encodings allow to represent probabilistic and fuzzy state estimates. Such a representation will be introduced and applied for the case of a redundant manipulator. Following the Mean of Multiple Computations principle, a neural network model (PbMMC) is presented in which the overall complexity is divided into multiple local relationships. This allows to solve inverse, forward and mixed kinematic problems. The local transformations in between the kinematic variables can be sufficiently well learned by small single MLP layers. The population codes of the kinematic variables are based on nested periodic receptive fields which allow to express multiple weighted state estimates. Therefore, the model as such is quite flexible as it can keep track of multiple possible solutions at the same time.**

I. Introduction

The notion of a schematic representation of the body is assumed a central part of the motor system which mediates interactions of an agent's body with the environment. Therefore, internal body models have been widely applied in robotics [1]. Internal models are used in order to solve the inverse kinematic problem as required in reaching tasks and visuomotor mappings are learned that allow to produce guided movements. In most cases these internal models are serving one specific function or are even restricted to one specific behavior.

In contrast, findings from behavioral and cognitive research have pointed out that internal representations in humans are quite flexible and modular [2]. In particular, a model of the own body appears to be a central representation [3] which can be recruited in different contexts, like motor control, perception or planning ahead [4]. This presupposes that the internal model of the body is not restricted to one specific function, but can be exploited in service of different functions and serving different tasks. Such models therefore have to be quite flexible and allow to be applied in different sets of tasks, e.g. can solve inverse kinematic problems, but at the same time should be predictive.

In this article, we introduce a neural network based model which can serve as an internal body model subserving any kinematic task. The model is following the Mean of Multiple Computations (MMC) principle [5] which basically divides the overall complexity into simple local and redundant relationships that are afterwards integrated again. This method has proven to be quite flexible as it allows to solve inverse [6], forward or any mixed kinematic problem [7]. The principle has been applied to different kinds of representations in the past [8], [9], but in each case the single neurons of the networks directly encoded the represented values as activations. In the traditional MMC example, a segment position is encoded as a vector in two neurons. Therefore, while an MMC network allows to provide only a selected set of inputs (e.g. one segment orientation and the end effector position), it can not deal with probabilistic or undetermined inputs. An MMC network can only assume a single configuration of the body at each point in time. In contrast, experimental findings indicate that the brain holds multiple active hypotheses at the same time and that humans can switch between these [10]. Such distributed representations are needed in order to cope with unreliable and partial input or when multiple possible solutions shall be tracked. As an example, consider when such a model is applied in perception. The visual mapping onto the body model should guide the assumed configuration, but when parts of the observed body are occluded, the model can only rely on vague input. For example, if the model can not observe most of an arm, then it can not reliably decide if the arm is in an elbow up or down configuration. It has to keep track of both possibilities until more information is available. Here we present the Population based Mean of Multiple Computation (PbMMC) approach, in which local kinematic representations are encoded in populations of neurons. This allows to represent and compute probability balanced configurations of the body inside the model.

The article will introduce the PbMMC approach and apply it to the example of a three segmented arm that works in a plane. Already this example is challenging, as the manipulator is redundant [11]. We will briefly introduce the classical MMC principle as such and explain how this can be transferred onto neural populations that encode the kinematic variables. In this case, the local relationships, which describe transformations in between the variables, are realized as single layers of MLPs.

MLPs have been used in the past to learn internal models, e.g. Giorelli et al. [12] learned a force based inverse model of a redundant cable driven manipulator. There, an MLP network is employed to overcome the problems regarding the generation of a Jacobi matrix for a highly redundant manipulator. After optimizing the network parameters, low positioning errors have been achieved. A network composed of radial basis functions (RBF) is applied in [13] for the control of planar movement of a SCARA robot. By learning the weights of the network and the centers as well as deviations of the RBFs, their approach outperformed an MLP network as well as a PID controller. Such approaches highlight the capabilities of MLPs in connection with radial basis functions. In the approach we suggest, periodic basis functions at multiple scales are utilized, which further improves upon radial basis functions as representations to encode joint angle and effector position

information in a population based way. In contrast to other approaches, the redundancy of the kinematic manipulator will be resolved through the distribution of the overall complexity onto multiple local relationships following the MMC principle.

As mentioned, the next section introduces the basic MMC principle, followed by the introduction of the population based neural fields and an explanation of the MLPs that connect the different fields. The result section will analyze in detail the case of a two-segmented arm in the inverse as well as the forward kinematic case and will extend this at last to a three segment redundant manipulator. The conclusions will briefly relate the approach to comparable current approaches and give an outlook on possible extensions.

II. The PbMMC model

Mean of Multiple Computations (MMC) is a principle by which an RNN can be constructed to solve forward, inverse and mixed kinematic problems within one single model [5], [8], [14]. This is achieved by shaping the models attractor space to represent valid combinations of joint-configurations and end-effector positions. The principle is applicable to non-redundant as well as redundant manipulators.

Let us consider as an example a serial kinematic manipulator with three rotational joints and three segments s_0, s_1 and s_2. We denote the end-effector as l_0. For the construction of an MMC network, auxiliary variables m_0 and m_1 are introduced. The kinematic chain can be decomposed into a number of triangular relations as visualized in Fig. 1. These triangular relations are defined in (1a - 1d).

$$\vec{l_0} = \vec{s_0} + \vec{m_1} \tag{1a}$$
$$\vec{l_0} = \vec{m_0} + \vec{s_2} \tag{1b}$$
$$\vec{m_0} = \vec{s_0} + \vec{s_1} \tag{1c}$$
$$\vec{m_1} = \vec{s_1} + \vec{s_2} \tag{1d}$$

In the classical MMC approach, each of the triangular relations is solved for each of the contained variables. Equation (1a) is reformulated as $\vec{s_0} = \vec{l_0} - \vec{m_1}$ and $\vec{m_1} = \vec{l_0} - \vec{s_0}$. As this leads to multiple equations for each variable, the individual formulas are combined by an unweighted mean computation. This leads to a redundant system of easily solvable equations which can directly be interpreted as the weight matrix of a recurrent neural network. The coefficients of the resulting equations are interpreted as input-weights to neurons in the network. This way, the network's attractor space is shaped to correctly represent kinematic constraints of the manipulator. As one additional connection in the recurrent neural network a form of self excitation is introduced which dampens the dynamics of the model and prevents oscillations [8]. The resulting network can be used to iteratively find configurations for all of the model's variables, if a subset of its variables is fixed to certain target values. Forward kinematics are computed by fixing joint-configurations and letting the network converge to a state with suitable end-effector coordinates. Inverse kinematics are computed by fixing end-effector coordinates and reading out joint-configurations after a short period of simulating the network's dynamics.

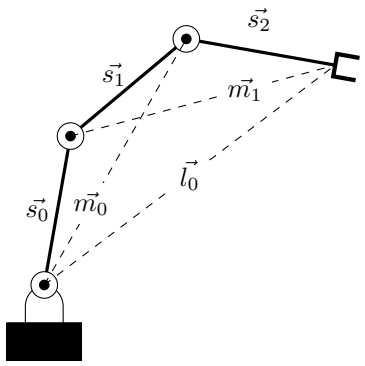

Fig. 1. Graphical representation of a planar (2D) arm consisting of three segments: upper arm ($\vec{s_0}$), lower arm ($\vec{s_1}$) and hand ($\vec{s_2}$). The vector $\vec{l_0}$ points to the position of the end effector (tip of the hand). Variables $\vec{m_0}$ and $\vec{m_1}$ represent additional diagonal vectors.

We extended the classical MMC approach by introducing population coded representations. Inspiration for this is drawn from findings on biological population codings based on head-direction cells [15], hippocampal place cells and entorhinal cortex grid cells [16], [17]. These distributed codings allow to represent ambiguous state estimates during computation. Not only the computational dynamics are altered this way, but also the network's input and output can carry more information. It is now possible to specify uncertainty in the input data or detailed temporary constraints on the permitted value range of certain variables. The chosen representations are described in Section II-B. As the coding of variables has been changed, we also needed to adapt the transformations between the variables. Section II-A features a description of how transformations have been implemented.

A. Transformations

Input-weights to neurons can be directly computed in classical MMC networks. This is because neurons directly correspond to individual coordinates and the vector-addition and vector-subtraction operations can be directly translated to the weighted input-summation as realized in neurons. Unfortunately, this advantageous property is not preserved when local representations are changed to population coding. For this reason, transformations between local modules have to be realized differently. We investigated two different approaches, both based on the introduction of additional hidden layers. These strategies differ locally in architecture and applied learning algorithms. Nevertheless these two approaches and classical MMC networks still share a common global structure and working principle. The transformation scheme that's based on learning of prototypes is described in detail in [18]. In this document we will focus on transformations realized by local feedforward networks as it requires a much lower number of mediating neurons and leads to a simpler structure.

As explained above, the main idea of the MMC approach is to decompose the complex kinematic structure into simpler and easy solvable local kinematic relationships. In the example, the overall structure consisting of multiple segments and the end-effector position are decomposed into local chains of three vectors. Each of these triangles, as shown in Fig. 1, defines one equation. And each of these equations contains three variables.

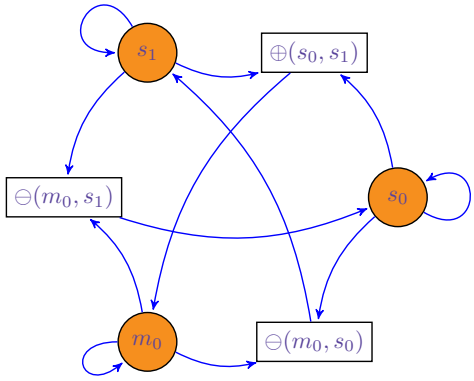

Fig. 2. Local architecture for one triangle. Orange nodes correspond to neural fields which represent kinematic variables. White boxes represent MLPs that transform and distribute activation.

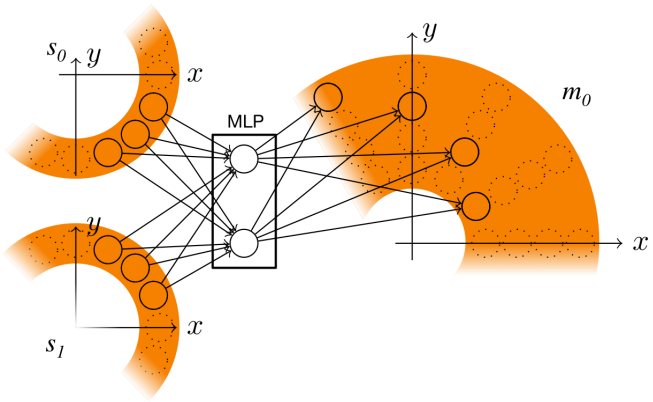

Fig. 3. Local architecture for one triangle. Transformation $m_0 = \oplus(s_0, s_1)$ realized by an MLP with one hidden layer.

Following the MMC principle, each equation will be solved for each of the three variables which leads to three reformulations for every equation. The triangle relationship (s_0, s_1, m_0) for example, defines the three rearranged equations $m_0 = s_0 + s_1$, $s_0 = m_0 - s_1$ and $s_1 = m_0 - s_0$. The population encoded representations used for variables s_0, s_1 and m_0 does not directly support addition and subtraction. It is therefore necessary to implement comparable operations for the population coded variables. We denote $\oplus$ the transformation equivalent to addition and $\ominus$ the transformation that corresponds to subtraction. Each specific instance of these transformations is realized by a single MLP. See Fig. 2 for a graphical description of the resulting local architecture. This representation of the triangular relationship constitutes the basic building block as needed in the MMC approach and is also a working kinematic model for a manipulator with two segments. Concerning the inverse kinematic case, the model can already solve the model selection problem between elbow-up and elbow-down solutions. Which solution is chosen depends on the initialization of the activations in the neural fields. Randomized activations as well as activations that represent a-priori information about individual segments can both be used for initialization.

When this approach is used to model the kinematics of a manipulator with more than two segments, multiple MLPs project activity onto a single neural population. In the current state, an unweighted mean is used to fuse the outputs of different MLPs. This could be enhanced by using a weighted mean computation to control the influence of individual variables. This would allow to incorporate additional sensor readings. Sensor failures could be detected and accounted for, similar to the error-handling that's described in [19]. Again, as an additional influence and to stabilize the dynamics, comparable to the classical MMC approach [8], the current activation of a neuron is also fed back as an input to that neuron and is fused with the outputs of the MLPs. In that particular case a damping value is used to weight the different inputs (it describes the fraction of the input which is given through self excitation).

B. Segment representations

In a traditional MMC network, kinematic variables are represented as vectors and each of a vector's coordinates is represented by exactly one neuron. The following two properties result from this: First, input and output to the network are specified by constraining the combined joint-end-effector-space to single values in a subset of its individual dimensions. This limits the range of applications that the network can be used for. Second, at each time step the network can hold only a single estimate for each local variable. Other kinds of local representations can potentially hold more useful information.

We changed the local representation of each variable to a distributed population coding, which allows to represent a higher amount of information about the variable's current state. Each variable is represented by a population of n neurons and each of those neurons has an associated receptive field in the kinematic modality that it is associated with. Neurons in the population that codes the end-effector position have receptive fields tuned to specific regions in end-effector space, while neurons in populations that code individual joints or segments have receptive fields in the respective angular spaces. Population codes empower the network to represent ambiguous state estimates, which makes it possible to achieve particle filter like computations.

1) One dimensional representations: Receptive fields in one dimensional spaces, as the parameter space of a rotational joint, can be modeled by one dimensional functions mapping onto the neuron's activation space. Neurobiological research has found evidence for the existence of head-direction cells. A population of head-direction cells jointly codes the global direction that an animal's head is facing and does so by the different receptive fields associated with the population's individual neurons. Each head direction cell's tuning curve is generally triangular or Gaussian in shape and contains a single peak [15]. In a previous study, place-field like Gaussian radial basis functions have been used to model kinematics using the MMC principle[18]. Fig. 3 schematically shows what our architecture locally looked like if such unimodal receptive fields were connected by an MLP.

Receptive fields with multiple peaks at different scales, such as grid-cells or stripe-cells, offer greater precision [20]. Biological evidence shows that such cells play a major role for spatial navigation in the mammalian brain [16]. As kinematics and navigation are related spatial reasoning domains, it seems

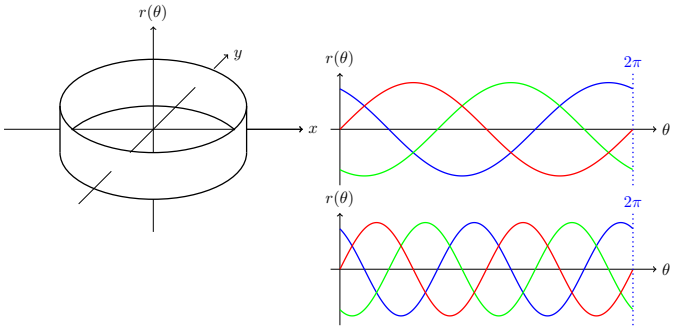

Fig. 4. Periodic receptive fields at multiple scales and shifted for different distances. The three-dimensional plot to the left indicates the spatial arrangement of the receptive fields. The basis functions virtually lie on this band around the rotational segment's axis.

sensible to examine usability of such receptive fields in the kinematics domain.

We use a set of shifted sine-functions on multiple scales as receptive fields in one-dimensional domains. These receptive field are defined by

$$r(\theta)_{\alpha,\delta} = \sin(\alpha(\theta + \delta\frac{2\pi}{\alpha})), \tag{2}$$

where α is a scaling factor and δ specifies the receptive field's displacement. It is possible to randomly sample values for α and δ to generate a set of receptive fields. We turned to a non-random uniform sampling of this parameter space

$$\{(\alpha = i, \delta = \frac{j}{N_s})|i < N_l, i \in \mathbb{N}, j < N_s, j \in \mathbb{N}\}, \tag{3}$$

where N_l is the number of levels in the hierarchy and N_s is the number of receptive fields on each level of the hierarchy. A population for $N_l = 2$ and $N_s = 3$ is depicted in Fig. 4.

2) Two dimensional representations: The working range of the manipulator's end-effector as well as the motion range of intermediate kinematic variables as m_0 and m_1 for an arm with three segments have the shape of a two-dimensional disc or an annulus. Receptive fields similar to those of place-cells, grid-cells or stripe-cells [17] can be used to encode spatial information in such a two-dimensional domain.

The receptive fields of stripe-cells are modeled as one-dimensional sine-functions in the two-dimensional plane. As in the one-dimensional representations described above, the functions occur at different scaling levels and with different offsets. The receptive field can be expressed as

$$r(x, y)_{\alpha,\delta,\phi} = \sin(\alpha(x\cos(\phi) + y\sin(\phi) + \delta\frac{2\pi}{\alpha})), \tag{4}$$

where α and δ specify the functions frequency and displacement as above. The rotation of the sine-function is defined by parameter ϕ. See Fig. 5 for a plot of three stripe-functions at different rotations.

We decided to construct populations by using a non-random uniform sampling of the parameter space. When the number of levels (N_l), number of rotations on each level (N_r) and number of displacements for each rotation (N_s) are

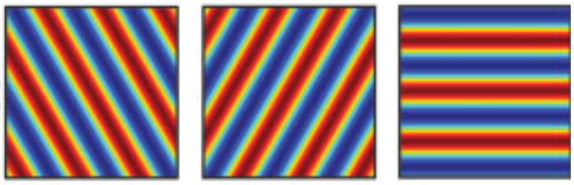

Fig. 5. Stripe-cell receptive fields can be modeled as rotated sine-functions in the two-dimensional plane [21].

specified, the used populations are uniquely defined by the parameter set

$$\{(\alpha = i, \delta = \frac{j}{N_s}, \phi = k\frac{\pi}{N_r})|i < N_l, i \in \mathbb{N}, \\ j < N_s, j \in \mathbb{N}, k < N_r, k \in \mathbb{N}\} \tag{5}$$

C. Stimulation and Estimation

As sensory data often comes in rate-coding and motor-commands equally often need to be issued in a rate-coded fashion, it is important to be able to convert between rate and population codings.

The conversion from rate to population coding can be interpreted as the presentation of a rate-coded stimulus to the population. The neurons' activations are then computed by evaluating their receptive fields at the position of the presented stimulus. This general recipe works no matter which kind of receptive fields are present in the population.

We interpreted the estimation of a population's coded value as maximization of the population's joint activity. We used grid-search with a high resolution as a maximization technique. This is computationally expensive and may be replaced with more sophisticated optimization techniques in future work.

D. Training

Training of the network is done individually for each of the local triangular sub-networks. The structure of such a local subnetwork has been explained above (Fig 2). While the representation fields' basis functions remain fixed, the network-weights into and out of the MLPs' hidden-layers are subject to learning. We describe the training process exemplarily for the triangle $m_0 = s_0 + s_1$.

In the triangle $m_0 = s_0 + s_1$, there are three transformations that are realized using MLPs and we need to distinguish forward and inverse transformations. We call the transformation $m_0 = \oplus(s_0, s_1)$ a forward transformation, as it is being used to compute forward kinematics and the transformations $s_0 = \ominus(m_0, s_1)$ and $s_1 = \ominus(m_0, s_0)$ are respectively called inverse transformations. We will first describe the training of forward transformations and afterwards the training of inverse transformations, as their training requires certain adaptations.

To train an MLP we need to generate training data and in the case of transformation $m0 = \oplus(s0, s1)$, this training data is generated by random sampling of example poses. We randomly sample joint values for s_0 and s_1 and use direct application of forward kinematics to compute the end-effector position for m_0. For each example pose, the suitable activation patterns in each representational population are being computed by

evaluating the population's receptive fields. When a set of activation patterns for s_0, s_1 and m_0 has been computed, these activations are used to train the $\oplus$-MLP by standard backpropagation.

During training of the inverse transformations two such hidden layers have to be trained at the same time. As there is recurrency in their combined graph structure, training of these $\ominus$-MLPs demands a different approach. Only the training of the MLP that realizes $s_0 = \ominus(m_0, s_1)$ is used to simplify the following explanations, but the other $\ominus$-transformation is learned in the same fashion.

First, of course the training data should consist of possible solutions to a given inverse problem. A naive approach to learning, based on activation patterns directly derived from example poses as in training of the $\oplus$-MLPs is not sufficient. The reason is that the input-data generated this way does not match the inputs that the MLP encounters during later simulations. When applied in a real scenario, the combination of activation patterns in m_0 and s_1, being inputs to $s_0 = \ominus(m_0, s_1)$, can differ dramatically from ideal pairs of input activation patterns. While m_0 is set to a certain target activation pattern that defines the inverse kinematic target, s_1 could still represent a joint-configuration from a previous kinematic computation and would most likely not coincide with one of the partial possible solutions. Such a combination of inputs differs drastically from the inputs observed during training.

More importantly, the fields s_0, s_1 and m_0 store complete activation patterns that are usually quite broadly distributed and noisy, especially after initialization and in the beginning of the simulation process when the network has not yet converged towards an attractor state. When the network is only trained with activation patterns that correspond to ideal solutions, a large part of the network's attractor space is not learned and the network will behave in a quasi chaotic way in the unexplored regions.

For these reasons, we have to extend the training and generate training data which fits the activation patterns that can be observed while applying both $\ominus$-transformations and running the model. The general idea is to alternate between simulating the network's dynamics and training the MLPs on basis of activation patterns collected during simulation. The activation patterns collecting in the beginning of the training process are quite chaotic and broadly sample the network's state space as we start from untrained MLPs $\ominus(m_0, s_0)$ and $\ominus(m_0, s_1)$ that do not yet shape the network's dynamics in a goal-directed way.

Given the initially chaotic input patterns from s_1 and those from m_0, we have to generate sensible target activation patterns for s_0 in order to train the MLP $\ominus(m_0, s_1)$. In order to come up with such an activation, first, a joint-configuration for segment s_1 is estimated from s_1's activation pattern (even though the activation can be quite broad and fuzzy). This estimate can then be used to search for a joint configuration for s_0, which minimizes the distance in the end-effector space. We used the following calculation rule:

$$\vec{s_0} = \begin{cases} (\vec{m_0} - \vec{s_1})\frac{r_{min}(s_0)}{|\vec{m_0} - \vec{s_1}|}, & \text{if } |\vec{m_0} - \vec{s_1}| \geq r_{min}(s_0) \\ (\vec{m_0} - \vec{s_1})\frac{r_{max}(s_0)}{|\vec{m_0} - \vec{s_1}|}, & \text{if } |\vec{m_0} - \vec{s_1}| \leq r_{max}(s_0) \\ \vec{m_0} - \vec{s_1}, & \text{otherwise} \end{cases}$$

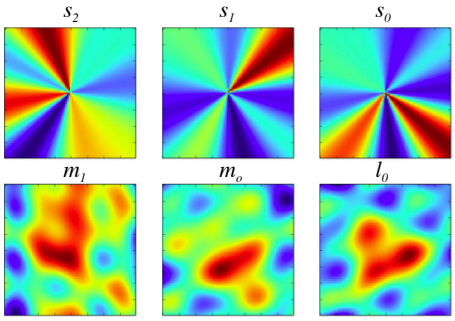

Fig. 6. Activation patterns in an unconverged state. Plots in the top row show representations for segments attached to rotational joints. These activation patterns are only dependent on the angular component of a stimulus. The bottom three plots show representations for m_0, m_1 and l_0.

The best estimate in the configuration space of s_0 then has to be translated to an activation pattern for s_0. This activation pattern can now be used as a target for training $\ominus(m_0, s_1)$ on the input activation patterns in the neural fields s_1 and m_0.

The described procedure has shown to be sufficient to explore the attractor space during learning. Starting from a random exploration of the global activation space, the model is robustly steered towards the attractor subspace that represents solutions for the inverse kinematic problem.

III. Simulation Results

We generated simulation results for a kinematic model with both, two and three rotational joints. The model with two rotational degrees of freedom (DoF) is at the same time PbMMC's basic triangular building block for longer kinematic chains, but also a simple kinematic model for a non-redundant manipulator (we are only interested in the two dimensional end-effector position, not its directional component). The model with three rotational DoFs is being used as a minimal example of a redundant manipulator. We will give a quantitative analysis of the models' capabilities to solve forward and inverse kinematic problems and also a qualitative analysis with an outlook on usability and theoretical advantages of the PbMMC model will be given.

The data presented in this section has been generated using MLPs with one hidden layer. $Tanh$ activation functions were chosen for both, hidden and output neurons. These were used because of good convergence properties during training and because the value range of $[-1.0, 1.0]$ is identical to that of the $sine$-functions used to model the stripe-cell receptive-fields. All errors numerically specified in this Section are computed as euclidean distances between target and estimation $\vec{x}_{target} - \vec{x}_{estimate}$ in the end-effector space. To increase comparability, these errors have additionally been normalized by the maximum radius of the end-effector working-space (i.e. $e = \frac{1.0}{6.0}(\vec{x}_{target} - \vec{x}_{estimate})$ for the 3DoF arm).

Each of the manipulators' segments was set to a length of 2.0, so the maximum radius of the manipulators' working ranges is 4.0 for two segments and 6.0 for three segments, respectively. The circular neural representations for segments (s_0,s_1,s_2) and discoidal representations for intermediate and

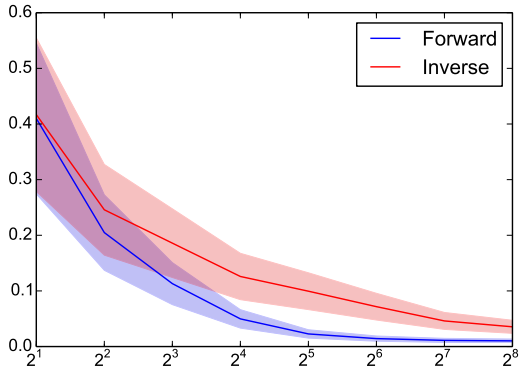

Fig. 7. Forward and inverse kinematics mean estimation error for the 2 DoF model plotted over number of hidden neurons per MLP. The error displayed on the y-axis consists of the mean euclidean error in end-effector space, normalized by the maximum radius $r_{max}(l_0) = 4.0$ of the end-effector variable. The plot has been generated by sampling 5000 random poses $(\theta(s_0), \theta(s_1))$ from pose-space and computing mean euclidean error in end-effector space. The plots are averaged over three different MLPs for each parameter set (number of neurons). Standard deviations of errors are plotted as shaded areas.

end-effector variables (m_0,m_1,l_0) have been constructed with the parameters specified in Table I. See Fig 6 for a plot of the activations of the neural fields when the network has not yet converged to an attractor state. It becomes obvious that even with a low number of neurons, the representations can hold rich information about a variables state.

A. Simulation results for the two-segmented case

Fig. 7 shows mean forward and inverse estimation errors of the two-segment model over the number of neurons per hidden layer. A larger number of neurons increases the model's performance for both, forward and inverse kinematics, but starts to hit a plateau at around 2^7(128) hidden neurons in both cases. Forward kinematic problems are solved with a low error. Inverse kinematics are solved with a somewhat greater error, yet still showing acceptable performance. When comparing the inverse kinematics error to the performance of other systems, it is important to note that our model not only optimizes a certain pose but solves the problem to decide which class of solutions shall be used. In the two-segment case the network decides between elbow-up and elbow-down solutions.

A more detailed look on the model's estimate in an inverse kinematics problem is given in Figs. 8, 9, 10 (128 hidden neurons per MLP). Fig. 8 shows the network's activation patterns and estimates after 10 iterations. As the model has been trained using both, elbow-up and elbow-down solutions, it converges to one of these classes of solutions and correctly solves the model-selection problem. But a specific solution can also be enforced by partially specifying the pose. The plots in Figure 9 show that the network correctly completes the sub-

	n_l	n_s	n_r	neurons
circle-field	5	3	-	15
disk-field	3	3	3	27

TABLE I. PARAMETERS OF THE NEURAL FIELDS

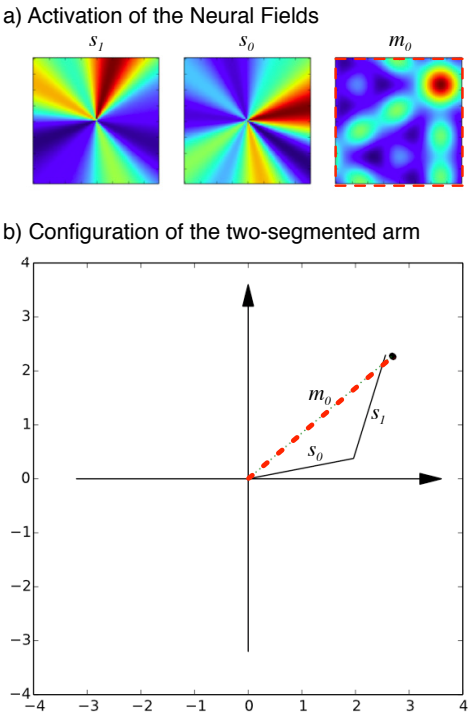

Fig. 8. The model has successfully solved an inverse kinematics problem. The dashed red line in the estimation plot in Subfigure b) indicates the end-effector target. The activation patterns from which the plotted estimations of segment-configuration have been computed can be seen in Subfigure a). The normalized euclidean error for this configuration is 0.037.

pose for s_1 for the same end-effector target, if s_0 is artificially set to its local part of the elbow-up solution.

Because training data has been shaped accordingly, the arm even finds the best possible solution if an exact solution is not possible. Figure 10 shows the network's solution for the same end-effector target with s_0 set to a configuration where a perfect solution is not possible. s_1 settles into a local pose that solves the problem as good as possible.

The training data used here has been generated in order to form the attractor space in a specific way. In this set of simulations, we want the network's state to converge in the end towards a single solution to the kinematic problem. By adapting the training data, it would also be possible not to reduce but further distribute uncertainty in the input data. In forward kinematics, for example, uncertainty about a certain segment's configuration could be transferred into uncertainty about the end-effector position. The model's behavior can be shaped in many different ways, subject to the choice of training data depending on the purpose of the application.

B. Simulation results for the three-segment manipulator

Mean forward and inverse kinematic errors of the model with three segments are displayed in Fig. 11. Errors are plotted over the number of neurons in each of the MLPs. The prediction error falls with an increased neuron count. As in the case of the model consisting of two segments, the error for the inverse and forward kinematic case is quite small. It shows that

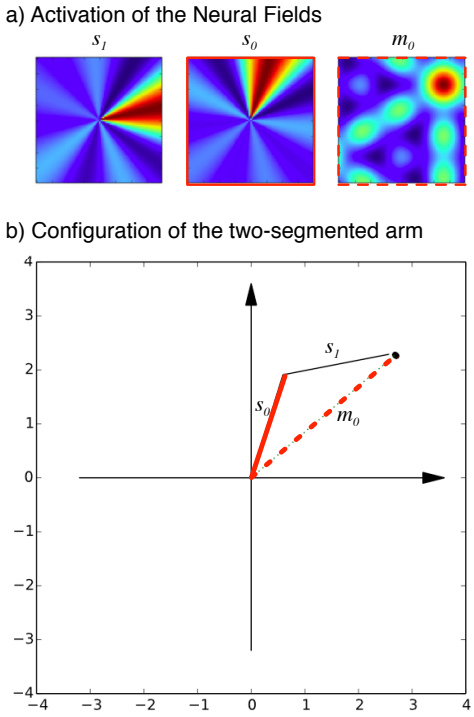

Fig. 9. The model can also find a solution if the same end-effector target as in Fig 8 is defined and an additional activation pattern is enforced for s_0 (indicated by the solid red line). Field s_1 is correctly set to the suitable other part of the elbow-up solution. Normalized euclidean error: 0.037

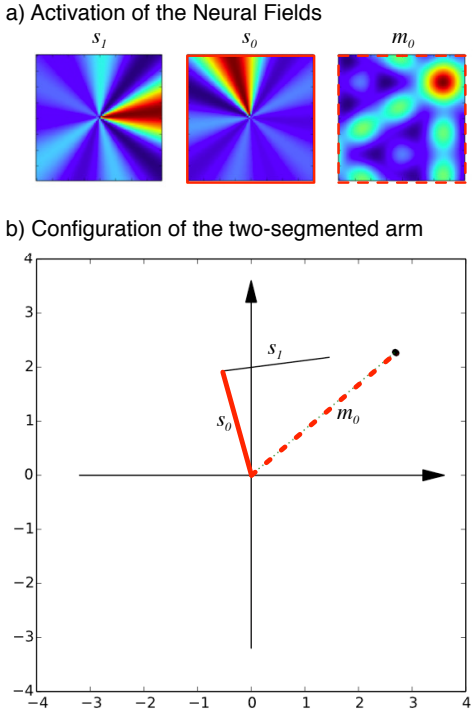

Fig. 10. As in Fig 9, s_0 has artificially been set to a certain configuration. But in contrast to the previous case, the new constraint does not permit a perfect solution. Field s_1 is set to a local solution that solves the problem as good as possible given the constraints. Normalized euclidean error: 0.314.

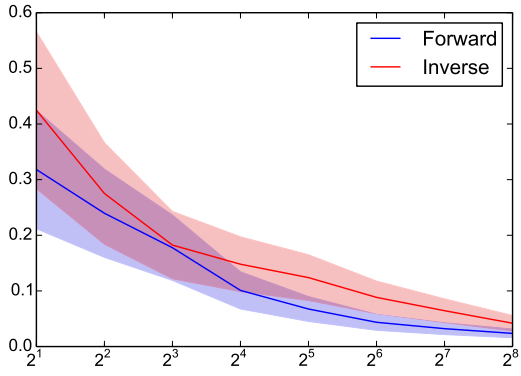

Fig. 11. Forward and inverse kinematics mean estimation error for the 3 DoF model plotted over number of hidden neurons per MLP. The error displayed on the y-axis consists of the mean euclidean error in end-effector space, normalized by the maximum radius $r_{max}(l_0) = 6.0$ of the end-effector variable. Each error plot has been generated using 2000 random targets and is averaged over three different MLPs for each different parameter set (number of neurons). Standard deviations of errors are plotted as shaded areas.

the model is able to solve the inverse and forward kinematic problem sufficiently well for a redundant manipulator.

A single solution to an inverse kinematics computation is presented in Figure 12. Activations alongside estimations are plotted after the network has already converged. The network has successfully decided for a certain class of poses and refined the pose to a solution of decent precision. The normalized euclidean error 0.059 for this solution lies between the model's mean inverse errors of 0.064 for $n = 128$ hidden neurons and 0.042 for $n = 256$ hidden neurons per MLP hidden layer.

As highlighted for the model with two segments, the model for the redundant three-segment manipulator also supports the solution of mixed kinematic problems. It manages to find solutions for partially defined poses, even when no perfect solution is possible given the constraints. This feature lends the model great flexibility in future applications, such as grasping while avoiding obstacles.

IV. Conclusion

The PbMMC model presented in this work extends the MMC model [5] by means of population coding, which allows a more fine-grained representation of segments' states. Additionally, the neural representation of segment relations creates building blocks which can be combined to create models for more complex manipulators, going beyond the ones used in this work to evaluate the PbMMC model. After evaluating the required dimensionality of the hidden layers of the employed MLPs, the model shows low error rates for forward and inverse kinematic queries with two and three segments. It is noteworthy, that in mixed queries with one segment set to a fixed position which prohibits an exact solution, the PbMMC achieves good results with the remaining DoFs. Future extension to three-dimensional spaces will demand introduction of three-dimensional basis-functions and slightly altered generation of training-data. We expect these to be only minor obstacles. Compared to similar approaches which exploit the combination of simple, local

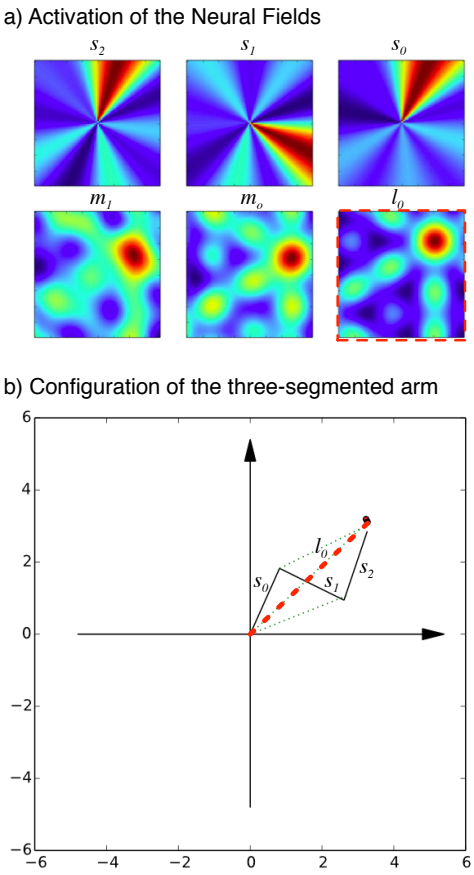

Fig. 12. The model with three segments has successfully solved an inverse kinematics problem. The end-effector target is indicated by the dashed red line. Subfigure a) shows the activation patterns in fields s_2,s_1 and s_0 in the top row. Activations for m_1,m_0 and l_0 are displayed in the bottom row. The estimations that have been computed from these activation patterns, plotted in Subfigure b) show that the network has indeed computed a valid solution. Normalized euclidean error: 0.059

equations in conjunction with population codes, for example the SURE_REACH model [22], the PbMMC approach does not require to explicitly learn prototypic postures but only local relations among segments. The explicit representation of prototypic postures also imposes a problem regarding the scalability of the model with respect to the complexity of the manipulator. To overcome these drawbacks, the authors presented the neural Modular Modality Frame (nMMF) in [19]. The nMMF model relies only on local representations and is in this respect comparable to the MMC approach as it follows in principle a similar idea. But it still needs a distinct model for the forward and inverse case. An advantage of the nMMF model is the possibility to fuse different kinds of sensor modalities, like orientation and locations frames, together. This possibility is also given by the underlying MMC model, so utilizing the PbMMC in multi-modal tasks should be also possible. For example, during body tracking tasks using mainly visual input, the possibility of mixed queries can augment episodes in which occlusion occurs by utilizing the kinematic body model.

References

[1] M. Hoffmann, H. G. Marques, A. Hernandez Arieta, H. Sumioka, M. Lungarella, and R. Pfeifer, "Body schema in robotics: a review," *Autonomous Mental Development, IEEE Transactions on*, vol. 2, no. 4, pp. 304–324, 2010.

[2] N. Cothros, J. D. Wong, and P. L. Gribble, "Are there distinct neural representations of object and limb dynamics?" *Experimental Brain Research*, vol. 173, no. 4, pp. 689–697, Sep 2006.

[3] G. Carruthers, "Types of body representation and the sense of embodiment," *Conscious. Cogn.*, vol. 17, no. 4, pp. 1302 – 1316, 2008.

[4] M. Anderson, "Neural reuse: A fundamental organizational principle of the brain," *Behavioral and Brain Sciences*, vol. 33, pp. 254–313, 2010.

[5] H. Cruse and U. Steinkühler, "Solution of the direct and inverse kinematic problems by a common algorithm based on the mean of multiple computations," *Biol. Cybern.*, vol. 69, no. 4, pp. 345–351, 1993.

[6] M. Schilling, J. Paskarbeit, J. Schmitz, A. Schneider, and H. Cruse, "Grounding an internal body model of a hexapod walker control of curve walking in a biologically inspired robot," in *Intelligent Robots and Systems (IROS), 2012 IEEE/RSJ International Conference on*, Oct 2012, pp. 2762–2768.

[7] M. Schilling and H. Cruse, "What's next: Recruitment of a grounded predictive body model for planning a robot's actions," *Frontiers in Cognition*, vol. 3, no. 383, p. doi:10.3389/fpsyg.2012.00383, 2012.

[8] M. Schilling, "Universally manipulable body models—dual quaternion representations in layered and dynamic mmcs," *Autonomous Robots*, vol. 30, no. 4, pp. 399–425, 2011.

[9] M. Schilling, "Dynamic equations in mmc networks: Construction of a dynamic body model," in *Proceedings of the 12th International Conference on Climbing and Walking Robots*, 2009, pp. 1047–1054.

[10] M. Shiffrar, "Movement and event perception," in *The Blackwell Handbook of Perception*, B. Goldstein, Ed. Blackwell Publishers, Oxford, 2001, pp. 237–272.

[11] N. A. Bernstein, *The Co-ordination and regulation of movements*. Pergamon Press Ltd., Oxford, 1967.

[12] M. Giorelli, F. Renda, G. Ferri, and C. Laschi, "A feed-forward neural network learning the inverse kinetics of a soft cable-driven manipulator moving in three-dimensional space," in *Intelligent Robots and Systems, 2013 IEEE/RSJ International Conference on*, 2013, pp. 5033–5039.

[13] M.-J. Lee and Y.-K. Choi, "An adaptive neurocontroller using rbfn for robot manipulators," *Industrial Electronics, IEEE Transactions on*, vol. 51, no. 3, pp. 711–717, 2004.

[14] M. Schilling, "Learning by seeing–associative learning of visual features through mental simulation of observed action," *Proceedings of the ECAL 2011*, pp. 731–738, 2011.

[15] J. S. Taube, "The head direction signal: origins and sensory-motor integration," *Annu. Rev. Neurosci.*, vol. 30, pp. 181–207, 2007.

[16] P. K. Pilly and S. Grossberg, "How do spatial learning and memory occur in the brain? coordinated learning of entorhinal grid cells and hippocampal place cells," *Journal of cognitive neuroscience*, vol. 24, no. 5, pp. 1031–1054, 2012.

[17] S. Grossberg and P. K. Pilly, "Coordinated learning of grid cell and place cell spatial and temporal properties: multiple scales, attention and oscillations," *Philosophical Transactions of the Royal Society of London B: Biological Sciences*, vol. 369, no. 1635, 2013.

[18] M. Baum, "Modeling kinematics of a redundant manipulator using population coding and the mmc principle," Master's thesis, Bielefeld University, Bielefeld, Germany, 2014.

[19] S. Ehrenfeld, O. Herbort, and M. V. Butz, "Modular neuron-based body estimation: maintaining consistency over different limbs, modalities, and frames of reference," *Front Comput Neurosci*, vol. 7, 2013.

[20] A. Mathis, A. V. Herz, and M. B. Stemmler, "Resolution of nested neuronal representations can be exponential in the number of neurons," *Physical review letters*, vol. 109, no. 1, p. 018103, 2012.

[21] T. Solstad, E. I. Moser, and G. T. Einevoll, "From grid cells to place cells: a mathematical model," *Hippocampus*, vol. 16, no. 12, pp. 1026–1031, 2006.

[22] M. V. Butz, O. Herbort, and J. Hoffmann, "Exploiting redundancy for flexible behavior: unsupervised learning in a modular sensorimotor control architecture." *Psychol. Rev.*, vol. 114, no. 4, p. 1015, 2007.