A Learning Framework for Generic Sensory-Motor Maps

Manuel Lopes Bruno Damas

***Abstract*— We present a new approach to cope with unknown redundant systems. For this we present i) an online algorithm that learns general input-output restrictions and, ii) a method that, given a partial set of input-output variables, provides an estimate of the remaining ones, using the learned restrictions. We show applications of the algorithm using examples of direct and inverse robot kinematics.**

***Index Terms*— manifold learning, humanoid robots, redundancy, sensory-motor coordination**

I. MOTIVATION

Forward-backward models represent generic maps between different spaces. These models can represent relations between motor control and sensor readings, or the relation among different sensor modalities, *e.g.*, the sound and the motion of a bell. Learning a map between such different spaces has applications in brain research [1], robot control [2] and sensory-motor coordination [3], among others fields. In many different settings it is necessary to learn these models without any knowledge about their structure.

The direct and inverse kinematics are fundamental functions for robot control and provide a very good example of the application of forward-backward maps. If we know a kinematic chain then it is possible to compute some of these functions analytically [4], [5]. Although the direct for serial mechanism and the inverse for parallel mechanisms are very easy to solve, the inverse problems may need more sophisticated solutions. These solutions can be either numeric or symbolic [4], [5] but, in general, are difficult to apply in today's robots with more than forty degrees of freedom. In this high-dimensional setting, planning a trajectory or computing the kinematics usually becomes a difficult task. Such redundant robots can solve the same task in many different ways and the main difficulty is to select the best option available.

Besides complexity, if the kinematic structure of the systems is unknown then the only solution may be to directly *learn* these functions from real data. Since robot kinematics are not, in general, bijective functions, they cannot, without further manipulations, be learned with a function fitting method [6], [7]. Even if one of the maps is injective, its inverse may not be. And so we will need to learn both maps independently.

Learning a structure including both maps can provide significant advantages, if such knowledge can later be used to recover both maps. Also, it is advantageous to be able able to deal with situations where multiple outcomes arise from a single input. Typical manifolds resulting of several sensory-motor coordination tasks have been studied in [8].

Note that all this information can be obtained with no necessity to conduct further learning, *i.e.*, one can easily change from an inverse kinematics problem to a forward one maintaining the manifold previously learned. Another desired feature is to have an online learning algorithm that could provide estimates at any point of the learning process, and that could accommodate new data without the need to perform heavy computations.

In this paper we present a new approach to work with unknown redundant systems. For this we present:

- An online algorithm that learns input-output restrictions of a generic smooth map;
- A method that, given a partial set of input-output variables, provides an estimate of the remaining ones, using the learned restrictions.

Referring to our problem, the manifold estimate can be used to obtain the direct and inverse robot kinematics, i.e., to provide an estimate of the observed variables given an actuation value, or, inversely, obtain the actuators position that leads to a desired observation.

This paper is organized as follows: Section II presents the related work. The formulation of our problem is presented in Section III. The algorithm is presented in two sections: Section IV presents the learning algorithm and Section V shows how can the model be used to recover direct and inverse kinematics. In section VI some examples illustrate the proposed algorithm and finally, in section VII, some conclusions and future work are presented.

II. RELATED WORK

Learning input-output functions can be done with local regression methods [9], neural networks [10], among others [11]. Of special importance is to know if the methods work under redundancy: a regression can only be made in many-to-one maps and not the opposite, so not only is necessary to learn the maps but also to be able to use them under redundancy.

The least-weighted projection regression (LWPR) algorithm [12] has been proposed to incrementally learn general non-linear functions with several local linear models. The authors proposed an approach to learn in humanoid robots [2] that includes position and velocity in the input to predict velocity in the output. We note that by including this extra information the inverse kinematics becomes injective,

M. Lopes and B. Damas are with the Instituto de Sistemas e Robótica, Instituto Superior Técnico, Lisboa, Portugal. B. Damas is also with Escola Superior de Tecnologia de Setúbal, Portugal. {macl,bdamas}@isr.ist.utl.pt

This work was (partially) supported by the FCT Programa Operacional Sociedade de Informação (POSI) in the frame of QCA III, and by the EU Proj. (IST-004370) RobotCub.

except in the singularities. For this method the problem then becomes the estimation of the direct kinematics because it is not easy to invert the learned model. Another approach that tries to circumvent the redundancy problem is to decompose the robot degrees of freedom in redundant and non-redundant ones [7]. The problem of this approach is that the decomposition is done manually, working only with a predefined selection of variables.

Many approaches have been used to solve the well known redundancy problem. One of them is the damped least-squares, where the inversion is made jointly with an energy minimization [13], originally introduced to solve the problem of controlling robots near singularities. Another procedure is called the redundancy formalism [14]: in this setting, the extra degrees of freedom can be used to solve other tasks, provided that the corresponding motion is done along a direction in the null space of the main motion. The work of [15] presents a robot under visual control, where redundancy is used to obtain better trajectories in a visual servoing task. In [16] these formalism is applied by learning the necessary jacobian information. Other works have dealt with planning in humanoid robots. A system able to decompose the forces in order to act both in the task domain, but also and independently in the robot posture, is presented in [17].

Instead of doing an input-output regression it is possible to learn the restrictions between both spaces. If they define a manifold is is possible to use a manifold learning mechanism. *ISOMAP* [18] and *LLE* [19] are two standard methods to do generic manifold estimation and both provide convergence proofs. These methods try to map the given manifold into a lower dimensional space. They are not applicable directly to our setting because they do not provide a parametric model of the high-dimensional data and do not work online.

III. Problem formulation

This paper presents a new approach to learn forward-backward models, allowing to easily recover the relation among any set of variables. The key point of our approach is to consider the problem from an unsupervised point of view, where data points consist of vectors containing *both* input and output variables. These vectors define a surface that can be seen as the graphic of a function. Consider D_c the number of controlled — or independent — variables and D_o the number of observed variables. A point $\mathbf{x}$ belonging to the manifold in a $D = D_c + D_o$ dimensional space will lie in a sub-space of dimension D_c. This manifold can be represented by the implicit function

$$\mathbf{H}(\mathbf{x}) = \mathbf{0}\ , \tag{1}$$

where $\mathbf{H}(\mathbf{x})$ imposes the $D - D_c$ restrictions arising from kinematics considerations. Note that the dimension of the manifold is D_c because this corresponds to the number of independent variables. The observed variables are generic smooth, frequently non-injective functions of the independent variables. In almost all cases these manifolds are highly nonlinear, hard to parameterize without any *a priori* knowledge. However, they are smooth and so can be approximated by local linear parameterizations estimated from sample data.

Unsupervised learning of a D_c-dimensional manifold in a D-dimensional space can be interpreted as a probability density estimation problem: given a set of (possibly corrupted with noise) sample points $\mathbf{x}_i$ belonging to the manifold, $i = 1 \ldots N$, estimate the probability of a point $\mathbf{x}$ belonging to the manifold, *i.e.*,

$$p(\mathbf{H}(\mathbf{x}) = \mathbf{0}\ |\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_N)\ . \tag{2}$$

For small regions the manifold can be approximately described by a hyperplane with dimension D_c. The covariance of a set of data points belonging to a small neighborhood provides a description of the manifold around that location. The D_c first principal components (in the sense of Principal Component Analysis (PCA)) of the covariance span a hyperplane given a local estimate of the manifold in that point. The remaining PCA dimensions can be neglected as they represent the noise affecting data.

In this paper a generic forward-backward model will be approximated by a collection of M models, where each model describes the manifold in the vicinity of its center μ_m by a local covariance matrix $\mathbf{C}_m$. This gaussian mixture representation provides a good estimate of the manifold if there are sufficient local models to appropriately cover the entire manifold, so that in the domain of each model the true manifold is approximately linear.

IV. Learning

A popular method to estimate a mixture of Gaussians is the expectation-maximization algorithm (EM) [20]. It is a method prone to local minima, specially if the data dimensionality is high. The number of Gaussians must also be defined beforehand, although some variations of EM exist that dynamically allocate new models [21].

A. Parameter update

Given a fixed number of models and assuming fixed and equal priors for each model, the EM algorithm iterates between the expectation step

$$p(m|\mathbf{x}_j) = \frac{p(\mathbf{x}_j|m)}{\sum_{m'=1}^{M} p(\mathbf{x}_j|m')} = \frac{p(\mathbf{x}_j|\mu_m, \mathbf{C}_m)}{\sum_{m'=1}^{M} p(\mathbf{x}_j|\mu_{m'}, \mathbf{C}_{m'})} \tag{3}$$

and the maximization step

$$\mu_m = \frac{\sum_{j=1}^{N} p(m|\mathbf{x}_j)\mathbf{x}_j}{\sum_{j=1}^{N} p(m|\mathbf{x}_j)} \tag{4}$$

$$\begin{aligned}\mathbf{C}_m &= \frac{\sum_{j=1}^{N} p(m|\mathbf{x}_j)(\mathbf{x}_j - \mu_m)(\mathbf{x}_j - \mu_m)^T}{\sum_{j=1}^{N} p(m|\mathbf{x}_j)} \\ &= \frac{\sum_{j=1}^{N} p(m|\mathbf{x}_j)\mathbf{x}_j\mathbf{x}_j^T - \mu_m\mu_m^T \sum_{j=1}^{N} p(m|\mathbf{x}_j)}{\sum_{j=1}^{N} p(m|\mathbf{x}_j)}\end{aligned} \tag{5}$$

until convergence.

In its original formulation EM is a batch algorithm. The sequential nature of data acquisition in real robots suggests modifying it to an online version, making it able to provide estimatives while acquiring new data.

Since the memory traces $\sum_j p(m|\mathbf{x}_j)$, $\sum_j p(m|\mathbf{x}_j)\mathbf{x}_j$ and $\sum_j p(m|\mathbf{x}_j)\mathbf{x}_j\mathbf{x}_j^T$ provide sufficient statistics to make online corrections for each model, this algorithm can be implemented with very low memory consumption. Whenever a new data point $\mathbf{x}_i$ arrives, these quantities are updated according to:

$$\begin{array}{lcl} \sum_j p(m|\mathbf{x}_j) & \longleftarrow & \sum_j p(m|\mathbf{x}_j) + p(m|\mathbf{x}_i) \\ \sum_j p(m|\mathbf{x}_j)\mathbf{x}_j & \longleftarrow & \sum_j p(m|\mathbf{x}_j)\mathbf{x}_j + p(m|\mathbf{x}_i)\mathbf{x}_i \\ \sum_j p(m|\mathbf{x}_j)\mathbf{x}_j\mathbf{x}_j^T & \longleftarrow & \sum_j p(m|\mathbf{x}_j)\mathbf{x}_j\mathbf{x}_j^T + p(m|\mathbf{x}_i)\mathbf{x}_i\mathbf{x}_i^T \end{array}$$

These quantities can then be used in expressions (4) and (5).

B. Structure update

The proposed online version of the EM algorithm allows an easy scheme to dynamically allocate new models. The algorithm is initialized with only one model, with center and covariance given, respectively, by the sample mean and sample covariance of the first $D+1$ points acquired (more points may be needed if the first D+1 points don't span the entire space). Each time a new data point $\mathbf{x}_i$ is acquired, the models μ_m and $\mathbf{C}_m$ are updated according to expressions (4) and (5). A new model is created if, for a new data point $\mathbf{x}_i$, the probability $p(\mathbf{x}_i|m)$ is smaller than a threshold p_{gen} for all existing models. The center of this new model becomes $\mu = \mathbf{x}_i$, while the covariance is equal to the covariance of the model that maximizes $p(\mathbf{x}_i|m)$.

C. Topology

From the collection of models obtained by the learning process we can also estimate the topology of the manifold, which gives a rough measure of the distance between points along the manifold.

Two different models m_1 and m_2 are considered adjacent (neighbors) if the point $\mathbf{x}$ that maximizes the likelihood product $p(\mathbf{x}|m_1)p(\mathbf{x}|m_2)$ is not a very unlikely one when considering each model.

Maximizing $p(\mathbf{x}|m_1)p(\mathbf{x}|m_2)$ is equivalent to minimize

$$J_2 = (\mathbf{x}-\mu_1)^T\mathbf{C}_1^{-1}(\mathbf{x}-\mu_1)+(\mathbf{x}-\mu_2)^T\mathbf{C}_2^{-1}(\mathbf{x}-\mu_2)\ , \quad (6)$$

which can be accomplished by the value $\mathbf{x}$ that satisfies

$$\frac{dJ_2}{d\mathbf{x}} = 0\ .$$

With some straightforward algebra the point $\mathbf{x}$ is given by:

$$\mathbf{x} = (\mathbf{C}_1^{-1} + \mathbf{C}_2^{-1})^{-1}(\mathbf{C}_1^{-1}\mu_1 + \mathbf{C}_2^{-1}\mu_2)\ . \quad (7)$$

If the quantity $p(\mathbf{x}|m)$ remains above a given threshold for both models m_1 and m_2 a neighborhood relation is considered to exist between models m_1 and m_2. If this procedure is repeated for every possible pair of models a neighborhood matrix can be constructed whose entry $\{ij\}$ equals 1 if models m_i and m_j are neighbors and 0 otherwise.

V. Query

Last section presented a method to estimate a manifold representing a generic input-output map restriction. This section shows how can be obtained, given a partial set of input-output variables (query), an estimate of the remaining ones.

Suppose data points $\mathbf{x}$ are divided into a query component and an answer component, $\mathbf{x} = [\mathbf{x}_q^T\ \mathbf{x}_a^T]^T$, such that $D_q + D_a = D$, where D_q is the query dimension and D_a is the answer dimension, not necessarily equal to D_c and D_o. The answer component is the set $\mathbf{x}_a$ of elements of $\mathbf{x}$ to be estimated given a specific value of the remaining elements $\mathbf{x}_q$. For instance, for a forward kinematics problem $\mathbf{x}_q$ corresponds to the actuation variables, while for an inverse kinematics problem $\mathbf{x}_q$ matches the observed variables.

Note that if the dimension of the query exceeds D_c, the manifold dimension, the estimation problem is overdetermined and a solution may not exist. Conversely, if the dimension of the query is lower than D_c, the estimation problem is under-determined and a continuum of solutions exist — in this case, as will be explained later, our algorithm will provide multiple answers that can be interpreted as a sampling of that continuous solution.

The M local models that describe the learned manifold can be used to provide an estimate $\hat{\mathbf{x}}_a$ for a specific query $\mathbf{x}_q$. For a *single* model m, we can choose the estimative $\hat{\mathbf{x}}_a$ to be the value that maximizes the likelihood of the data point $\mathbf{x}$ given model m, *i.e.*, that maximizes $p(\mathbf{x}|m)$. Maximization of this likelihood can be achieved by minimizing the corresponding Mahalanobis distance to the center of the model m[1]:

$$J_1 = (\mathbf{x}-\mu)^T\mathbf{C}^{-1}(\mathbf{x}-\mu)\ . \quad (8)$$

To do this, consider $\bar{\mathbf{x}}_q = \mathbf{x}_q - \mu_q$ and $\bar{\mathbf{x}}_a = \mathbf{x}_a - \mu_a$, where $\mu = [\mu_q^T\,\mu_a^T]^T$. Consider also the decomposition

$$\mathbf{C}^{-1} = \left[\begin{array}{c|c} \mathbf{C}_{qq} & \mathbf{C}_{qa} \\ \hline \mathbf{C}_{aq} & \mathbf{C}_{aa} \end{array}\right]\ ,$$

where $\mathbf{C}_{qq}$, $\mathbf{C}_{qa}$, $\mathbf{C}_{aq}$ and $\mathbf{C}_{aa}$ are respectively $D_q \times D_q$, $D_q \times D_a$, $D_a \times D_q$ and $D_a \times D_a$. Then expression (8) can be written as

$$\begin{array}{rcl} J_1 & = & [\bar{\mathbf{x}}_q^T \quad \bar{\mathbf{x}}_a^T] \left[\begin{array}{cc} \mathbf{C}_{qq} & \mathbf{C}_{qa} \\ \mathbf{C}_{aq} & \mathbf{C}_{aa} \end{array}\right] \left[\begin{array}{c} \bar{\mathbf{x}}_a \\ \bar{\mathbf{x}}_a \end{array}\right] \\ & = & \bar{\mathbf{x}}_q^T\mathbf{C}_{qq}\bar{\mathbf{x}}_q + \bar{\mathbf{x}}_q^T\mathbf{C}_{qa}\bar{\mathbf{x}}_a + \bar{\mathbf{x}}_a^T\mathbf{C}_{aq}\bar{\mathbf{x}}_q + \bar{\mathbf{x}}_a^T\mathbf{C}_{aa}\bar{\mathbf{x}}_a \\ & = & \bar{\mathbf{x}}_q^T\mathbf{C}_{qq}\bar{\mathbf{x}}_q + 2\,\bar{\mathbf{x}}_a^T\mathbf{C}_{aq}\bar{\mathbf{x}}_q + \bar{\mathbf{x}}_a^T\mathbf{C}_{aa}\bar{\mathbf{x}}_a\ , \quad (9) \end{array}$$

where the simmetry of C^{-1} implies $\mathbf{C}_{qa} = \mathbf{C}_{aq}^T$.

With $\hat{\mathbf{x}}_q$ fixed, the estimate $\hat{\mathbf{x}}_a$ satisfies:

$$\frac{dJ_1}{d\mathbf{x}_a} = 0\ ,$$

and after some simple calculations we get the estimate

$$\hat{\mathbf{x}}_a(\mathbf{x}_q) = -\mathbf{C}_{aa}^{-1}\mathbf{C}_{aq}(\mathbf{x}_q - \mu_q) + \mu_a\ . \quad (10)$$

[1]for the sake of simplicity the model indexes are omitted.

Note that each estimate $\hat{\mathbf{x}}_a$ can associated with a corresponding "degree of confidence", given by $p(\hat{\mathbf{x}}|m)$, where $\hat{\mathbf{x}} = [\mathbf{x}_q^T \ \hat{\mathbf{x}}_a^T]^T$. In this way, models that produce high probability estimates have a higher degree of confidence.

After obtaining an estimate $\hat{\mathbf{x}}_{am}$ for each model m a question that remains to be addressed is how to combine these estimates in order to arrive to a definitive solution.

Combining the models answers — for instance, using a weighted average — won't work when a query can produce multiple answers: the planar robot described in section VI, for instance, usually has two different actuator configurations that cause the same end effector position, and an average of these configurations can lead to a point not belonging to the kinematics manifold. Note that a weighted average does not take into account the manifold distance between different models. The correct solution can, however, be produced if models are grouped according to their neighborhood, thus preventing multiple solutions, when they exist, from being mixed into a unique solution.

Using the neighborhood relations between different models we can devise a simple procedure to obtain a better solution by combining several local estimates. First, estimates $\hat{\mathbf{x}}_{am}$ are ordered according to $p(\hat{\mathbf{x}}|m)$, discarding models whose "degree of confidence" are below a given threshold. The remaining models are then grouped according to their neighborhood. The procedure starts by considering only one set S_1 containing the first model of the ordered list of models. Then each other model, starting from the second, checks if it is a neighbor of the first element of each set S_k, being appended in each set for which the neighborhood relationship holds. A new set is created and initialized with a particular model m if m was not inserted in any existing set.

After this procedure is finished the number of sets generated, N_S, corresponds to the number of different answers that can be estimated for the given query. Since models are now arranged according to their neighborhood, it is straightforward to obtain the set of possible answers: for the k^{th} answer, with $1 \leq k \leq N_S$, we have

$$\hat{\mathbf{x}}_a^k = \frac{\sum_{m \in S_k} \hat{\mathbf{x}}_{am} p(\hat{\mathbf{x}}|m)}{\sum_{m \in S_k} p(\hat{\mathbf{x}}|m)} \ . \tag{11}$$

A "degree of confidence" for each answer $\hat{\mathbf{x}}_a^k$ can also be found using the following relationship:

$$p^k = \frac{\sum_{m \in S_k} p(\hat{\mathbf{x}}|m)}{\sum_{j=1}^{N_S} \sum_{m \in S_j} p(\hat{\mathbf{x}}|m)} \ . \tag{12}$$

This quantity can be used, for instance, to discard estimates originated by a deficient manifold learning — these will typically have a low value of p^k.

In this section we presented a way to extract information from the parametric representation of the data manifold. Next section presents numerical simulations to verify the proposed algorithm.

VI. Experiments

We performed several simulations to assess the quality of the proposed algorithm, starting with a one-dimensional manifold embedded in a two-dimensional space. This manifold is the curve $(cos(t/10), t)$ parametrized by $t \in [-1 \ldots 1]$. 200 points were randomly generated from the function and thereafter we run our algorithm with this dataset. Figure 1 shows the learned manifold. We can easily confirm that the curve is well approximated with the given local models. More models were assigned to regions of high curvature (regions corresponding to $t = \pi/2 \pm \pi$) to correctly approximate the curve.

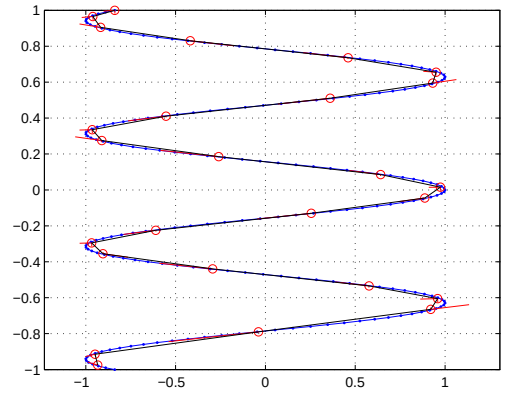

Fig. 1. Learned manifold ($D = 2$, $D_c = 1$). Red circles represent model centers, red lines represent the principal components of the covariance for each model, black lines represent the neighborhood relations between models and blue dots stand for the sample points.

The learned local models can be used to recover both the forward and backward models, *i.e.*, the cosine or the inverse cosine relation. Assuming the latter one, figure 2 shows the resulting predictions for $x_q = 0$, $x_q = 0.5$, $x_q = 0.9$ and $x_q = 1.1$. The algorithm presented in Section V correctly estimates the output variable, although, for $x_q = 1.1$, it still provides an answer when none exist. This situation is, of course, a consequence of the probabilistic approach to the description of the manifold, and vanishes as the number of models increases, allowing a better representation of high curvature zones of the manifold. For a value of $x_q > 1.3$ the algorithm does not provide an answer anymore, as would be expected.

Figure 3 shows a one dimensional manifold embedded in a three dimensional space. As we can observe, after the learning step the distance along the manifold prevails over the Euclidean distance between model centers when defining the neighborhood relationships. Figure 4 illustrates the estimation of multiple outcomes for a query point given by $q = 0.5$. Any of the remaining two dimensions, of course, could instead be used as the query variable.

The final example depicts a simple planar robot with two rotational degrees of freedom. In this particular case each link length is given by $L_1 = 1$ and $L_2 = 1$. If we consider the end effector cartesian coordinates (x, y) as observed

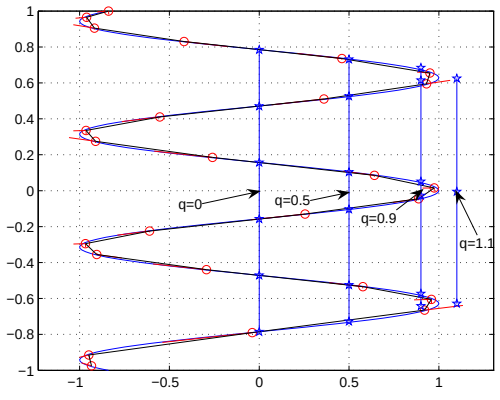

Fig. 2. Recovering the forward model embedded in the manifold. The several predictions corresponding to each query are represented by black asterisks.

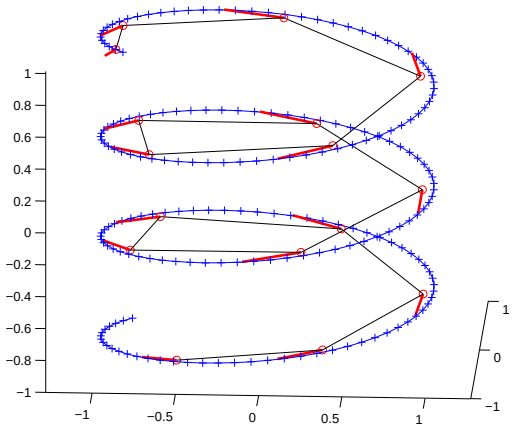

Fig. 3. Learned manifold ($D = 3$, $D_c = 1$). Red circles represent model centers, red lines represent the principal components of the covariance for each model, black lines represent the neighborhood relations between models and blue crosses stand for the sample points.

variables and the rotational variables (θ_1, θ_2) as the actuation variables, the following relations hold:

$$\begin{cases} x &= L_1 cos(\theta_1) + L_2 cos(\theta_1 + \theta_2) \\ y &= L_1 sin(\theta_1) + L_2 sin(\theta_1 + \theta_2) \end{cases}$$

These relations define a two dimensional manifold in a four dimensional space. Figure 5 plots the observed variable x against the actuation variables, representing also the models obtained after learning the manifold over 8 000 random data points. The topology for this problem is successfully learned, as it is illustrated in Figure 6. Although not clearly visible in Figure 5, there are two centers in the central region of Figure 6 for which no neighborhood is defined: these correspond to erroneous models that in the beginning of the learning process have grown too much; usually its covariance matrix has a much larger determinant compared to the correct models, and so it can be easily identified. Another way to

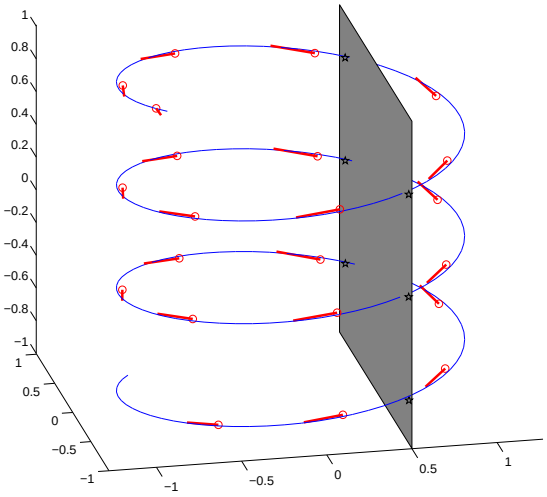

Fig. 4. Recovering the forward model embedded in the manifold. With $x_q = 0.5$ the six possible outcomes are successfully estimated (represented in the figure by black asterisks).

detect such outliers is to inspect the neighborhood relations, since these points usually become isolated ones.

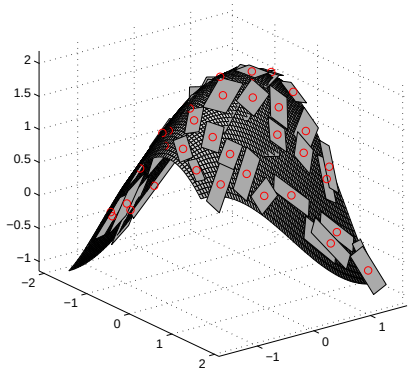

Fig. 5. Learned manifold ($D = 4$, $D_c = 2$). For convenience only the x variable is shown. Red circles represent model centers, while rectangles represent the two principal directions of each covariance matrix.

The proposed learning scheme is also able to tolerate a reasonable level of noise in the sample data. For the previous problem, illustrated in Figure 5, several experiments were conducted, with increasing levels of noise, and the mean square error of the estimative was obtained. The results are shown in Table I.

VII. Conclusions

We presented a general algorithm to learn the sensory-motor manifold resulting from robotic kinematic structures. This algorithm is computationally very efficient: the most time consuming operations are the local models covariance matrices inversions after the update of equation (5), *i.e.*, M

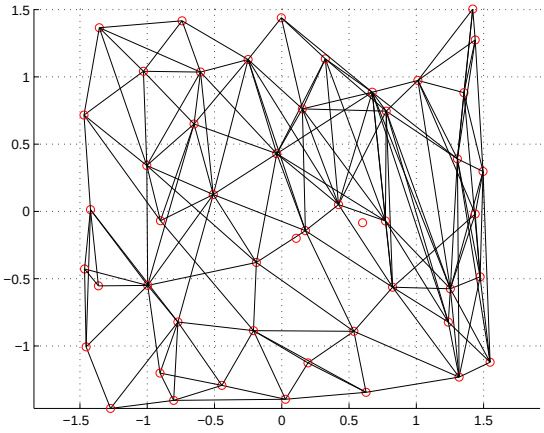

Fig. 6. Resulting topology of the manifold. The red circles represent the local models and the black lines represent the neighborhood. The model without neighbors correspond to outliers.

TABLE I

MEAN SQUARED ERROR FOR DIFFERENT LEVELS OF NOISE IN THE TRAINING SET.

noise	0.0001	0.001	0.01	0.1
mse	0.00003	0.00004	0.00005	0.0148

$N \times N$ matrix inversions must be performed each time a new point is incrementally incorporated by the learning mechanism. Considering only a subset of the nearest models in this update step can significantly speed up the process when the number of models starts to grow. Note also that in high dimension spaces the inversion of the updated covariance matrix can be done efficiently using the Woodbury identity.

This online learning algorithm can provide a robot with real-time learning without a previous acquisition of data, and it does not need any previous information about the underlying manifold. It can also be easily modified to incorporate a forgetting factor in the updates described in Section IV, thus potentially being able to track slow time-varying data.

The convergence of this modified EM algorithm is a key issue when dealing with high dimension data. So far, the proposed learning algorithm can only deal efficiently with low dimension data. As N increases, the solution tends to be stuck in poor local maxima of the likelihood function. Also, it becomes increasingly difficult to tune the few parameters of the algorithm. Work is currently being done to deal with these severe limitations. It should be stressed, however, that the major claim of this work is the way we can efficiently use a learned mixture of gaussians in the product space of the actuated and observed variables to infer both the forward and backward kinematics. Such a scheme can even be used to infer a mixture of both controlled and observed variables, and has enough flexibility to be easily replaced by any other mixture of gaussians learning method (see, for example, [22]).

The proposed method deals naturally with the classical problem of redundancy and non-injectivity in the forward-backward maps, being able to extract multiple solutions when multiple answers do exist. These solutions can then be used by a higher level algorithm to choose the final solution, *e.g.*, taking into account obstacle avoidance or energy minimization schemes.

REFERENCES

[1] D. M. Wolpert and M. Kawato. Multiple paired forward and inverse models for motor control. *Neural Networks*, 11(7-8):1317–1329, 1998.

[2] Aaron D'Souza, Sethu Vijayakumar, and Stephan Schaal. Learning inverse kinematics. In *International Conference on Intelligent Robots and Systems*, Hawaii, USA, 2001.

[3] R. Pfeifer and C. Scheier. Sensory-motor coordination: The metaphor and beyond. *Robotics and Autonomous Systems*, 20(22):151–178, Jun. 1997.

[4] Lung-Wen Tsai. *Robot Analysis: The Mechanics of Serial and Parallel Manipulators*. Wyley-Interscience, 1999.

[5] Richard M. Murray, Zexiang Li, and S. Shankar Sastry. *A Mathematical Introduction to Robotic Manipulation*. CRC, 1994.

[6] J. Nakanishi, R Cory, M. Mistry, J. Peters, and S. Schaal. Comparative experiments on task space control with redundancy resolution. In *IEEE International Conference on Intelligent Robots and Systems (IROS 2005)*, 2005.

[7] Manuel Lopes and José Santos-Victor. Learning of sensory-motor maps in redundant robots. In *IEEE Intelligent Robotic Systems (IROS'06)*, Beijing, China, 2006.

[8] R.A. Peters II and O.C. Jenkins. Uncovering manifold structures in robonaut's sensory-data state space. In *IEEE-RSJ International Conference on Humanoid Robotics*, Tsukuba, Japan, December 2005.

[9] Chistopher G. Atkeson, Andrew W. Moore, and Stefan Schaal. Locally weighted learning. *Artificial Intelligence Review*, 11(1-5):11–73, 1997.

[10] Tin-Yau Kwok and Dit-Yan Yeung. Constructive algorithms for structure learning in feedforward neural networks for regression problems. *IEEE Trans. on Neural Networks*, 8(3):630–645, 1997.

[11] Trevor Hastie, Robert Tibshirani, and Jerome Friedman. *The Elements of Statistical Learning*. Springer, 2001.

[12] S. Schaal and C. G. Atkeson. Constructive incremental learning from only local information. *Neural Computation*, 10(8):2047–2084, 1998.

[13] C.W. Wampler. Manipulator inverse kinematics solution based on damped least-squares solutions. *IEEE Trans. Systems, Man and Cybernetics*, 16(1), 1986.

[14] C. Samson, M. Le Borgne, and B. Espiau. *Robot Control: the Task Function Approach*. Clarendon Press, Oxford, United Kingdom, 1991.

[15] N. Mansard and F. Chaumette. Tasks sequencing for visual servoing. In *IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS'04)*, Sendai, Japan, November 2004.

[16] Nicolas Mansard, Manuel Lopes, José Santos-Victor, and François Chaummette. Jacobian estimation methods for task sequencing in visual servoing. In *IEEE - Intelligent Robotic Systems (IROS'06)*, Beijing, China, 2006.

[17] Oussama Khatib, Oliver Brock, Kyong-Sok Chang, Diego Ruspini, Luis Sentis, and Sriram Viji. Human-centered robotics and interactive haptic simulation. *International Journal of Robotics Research*, 23(2), 2004.

[18] J. B. Tenenbaum, V. de Silva, and J. C. Langford. A global geometric framework for nonlinear dimensionality reduction. *Science*, 290(5500):2319–2323, December 2000.

[19] Sam Roweis and Lawrence Saul. Nonlinear dimensionality reduction by locally linear embedding. *Science*, 290(5500):2323–2326, december 2000.

[20] A. P. Dempster, N. M. Laird, and D. B. Rubin. Maximum likelihood from incomplete data via the em algorithm. *J. Royal Statistical Society B*, 39:1–38, 1977.

[21] Nikos Vlassis and Aristidis Likas. A greedy em algorithm for gaussian mixture learning. *Neural Processing Letters*, 15(1):77–87, 2002.

[22] Matthew Brand. Charting a manifold. *Advances in Neural Information Processing Systems*, 15:985–992, 2003.