Trajectory Inverse Kinematics by Conditional Density Modes

Chao Qin Miguel Á. Carreira-Perpiñán

***Abstract*— We present a machine learning approach for trajectory inverse kinematics: given a trajectory in workspace, to find a feasible trajectory in angle space. The method learns offline a conditional density model of the joint angles given the workspace coordinates. This density implicitly defines the multivalued inverse kinematics mapping for any workspace point. At run time, given a trajectory in the workspace, the method (1) computes the modes of the conditional density given each of the workspace points, and (2) finds the reconstructed angle trajectory by minimising over the set of modes a global, trajectory-wide constraint that penalises discontinuous jumps in angle space or invalid inverses. We demonstrate the method with a PUMA 560 robot arm and show how it can reconstruct the true angle trajectory even when the workspace trajectory contains singularities, and when the number of inverse branches depends on the workspace location.**

I. Introduction

We consider the problem of *trajectory inverse kinematics* (IK) [1] of a (say) robot arm, where given a sequence of positions $\mathbf{x}_1, \dots, \mathbf{x}_N$ in (Cartesian) workspace of the end-effector, we want to obtain a feasible sequence of joint angles $\boldsymbol{\theta}_1, \dots, \boldsymbol{\theta}_N$ that produce the $\mathbf{x}$-sequence (we do not consider dynamics in this paper). Given the joint angles, the end-effector position is given by the forward kinematics mapping, $\mathbf{x} = \mathbf{f}(\boldsymbol{\theta})$, which is usually (but not necessarily) known. However, the inverse $\mathbf{f}^{-1}(\mathbf{x})$ can take multiple values, or for redundant manipulators (where $\dim \boldsymbol{\theta} > \dim \mathbf{x}$), an infinite number of them; this makes it difficult to represent and compute $\mathbf{f}^{-1}$. At the same time, we want the recovered sequence of joint angles to trace a continuous, realisable trajectory. Importantly, our goal is not only to solve IK at each trajectory point, but also to obtain an angle trajectory that is globally feasible (e.g. avoiding discontinuities or forbidden regions). Similar IK-related problems arise in other areas: in computer graphics, where one wants to achieve realistic animation of articulated characters (e.g. [2]); in the articulatory inversion problem of speech, where an acoustic waveform ("position $\mathbf{x}$") may be produced by different vocal tract shapes ("angles $\boldsymbol{\theta}$") [3], [4]; and in the protein loop closure problem in computational biology [5].

The forward kinematics mapping $\mathbf{f}$ can usually be obtained in closed form for a kinematic chain as a product of homogeneous transformation matrices, one per link (however, we remark that this is not always the case, as in articulatory inversion). There exist many approaches to IK; we briefly review some of them here (see [6], [7], [8] for review). In *analytic* approaches, one tries to obtain the IK mapping in closed form (e.g. [9]); this is only possible for certain types of manipulators, and even then it can be complicated. *Local* methods [10], [11], [8] are based on linearising the forward mapping to obtain $\dot{\mathbf{x}} = \mathbf{J}(\boldsymbol{\theta})\dot{\boldsymbol{\theta}}$, where $\mathbf{J}$ is the Jacobian of $\mathbf{f}$ (Resolved Motion Rate Control [10]). This equation can then be integrated numerically in order to obtain the global trajectory for $\boldsymbol{\theta}$. For redundant manipulators, where $\mathbf{J}$ has more columns than rows, a unique value of $\dot{\boldsymbol{\theta}}$ may be obtained by optimising a suitable objective (such as energy) over the nullspace of the Jacobian; in particular, one can obtain the pseudoinverse method by minimising $\|\dot{\boldsymbol{\theta}}\|^2$, yielding $\dot{\boldsymbol{\theta}} = \mathbf{J}^+(\boldsymbol{\theta})\dot{\mathbf{x}}$, at a computational cost $\mathcal{O}(m^2 n)$ where $m = \dim \mathbf{x} < n = \dim \boldsymbol{\theta}$. However, the idea breaks down at singularities $\boldsymbol{\theta}^*$, where $\mathbf{J}(\boldsymbol{\theta}^*)$ becomes singular; this is caused by the existence of multiple inverse branches intersecting at $\boldsymbol{\theta}^*$. Also, the numerical error can accumulate over time, and the computational cost is high since many pseudoinverses of non-sparse Jacobians must be computed. Other local methods [8] use an augmented set of variables $(\dot{\mathbf{x}}, \boldsymbol{\theta})$ rather than just $\dot{\mathbf{x}}$. Another local method (well-known in articulatory inversion) is *analysis-by-synthesis*, which directly finds an inverse value $\boldsymbol{\theta}$ of $\mathbf{f}$ by iteratively minimising the squared error $E(\boldsymbol{\theta}) = \|\mathbf{x} - \mathbf{f}(\boldsymbol{\theta})\|^2$ with a numerical optimisation method, e.g. gradient descent, where $\nabla E = 2\,\mathbf{J}(\boldsymbol{\theta})^T(\mathbf{f}(\boldsymbol{\theta}) - \mathbf{x})$. Unfortunately, which inverse value is found depends on the initial value for $\boldsymbol{\theta}$, and the iteration may also get stuck at non-inverse values where $\mathbf{J}(\boldsymbol{\theta})^T(\mathbf{f}(\boldsymbol{\theta}) - \mathbf{x}) = \mathbf{0}$ but $\mathbf{f}(\boldsymbol{\theta}) \neq \mathbf{x}$. However, the method is useful if the initial $\boldsymbol{\theta}$ is sufficiently close to the inverse sought. *Global* methods [12], [13] propose a variational approach where the trajectory of $\boldsymbol{\theta}$ minimises a functional $\int_{t_0}^{t_1} G(\boldsymbol{\theta}, \dot{\boldsymbol{\theta}}, t)\, dt$ (such as energy and manipulability) subject to the forward kinematic constraint $\mathbf{x}(t) = \mathbf{f}(\boldsymbol{\theta}(t))$ and appropriate boundary conditions. The trajectory is obtained by numerical integration of the corresponding Euler-Lagrange equation. However, the method still suffers from singularities [13] and needs the user to provide boundary conditions that are often unknown. Thus, an important problem of many of these methods are the singularities of the Jacobian. These correspond to the intersection of multiple inverse branches (violating the inverse function theorem), and while locally any of these branches is valid a priori, globally perhaps only one is valid.

A different type of methods are based on machine learning (data-driven) techniques. These methods estimate the inverse mapping using a training set of input-output pairs $(\mathbf{x}, \boldsymbol{\theta})$. While the forward mapping may be learnt by fitting a (say) neural net directly to pairs $(\boldsymbol{\theta}, \mathbf{x})$, learning the inverse mapping by fitting a neural net to pairs $(\mathbf{x}, \boldsymbol{\theta})$ would average the different inverse branches, yielding invalid solutions. For example, consider the function $x = f(\theta) = \theta^2$; the least-

EECS, School of Eng., University of California, Merced, P.O. Box 2039 Merced, CA 95344, US, {cqin,mcarreira-perpinan}@ucmerced.edu.

squares error $\|\theta - g(x)\|_2^2$ is minimised by $g(x) = 0$, the average of the two branches $\pm\sqrt{x}$, and this is what a neural net would yield. The distal learning approach [14] first trains a neural net to model the forward kinematics $\mathbf{f}$; then another net is prepended to this, and the resulting, cascaded network is retrained to learn the identity but keeping unchanged the weights of the forward model. This results in the prepended portion of the network learning one of the possible inverses (with the other branches being irreversibly lost). DeMers and Kreutz-Delgado [7], [15] try to identify (by clustering) subsets of the data corresponding to different branches, i.e., representing one-to-one mappings, and then they fit to each of them a neural net. However, in practice it is hard to identify such subsets. D'Souza et al [8] fit a locally weighted projection regression to map $(\dot{\mathbf{x}}, \boldsymbol{\theta})$ to $\dot{\boldsymbol{\theta}}$ as in the local methods discussed above.

We present a different method based on *directly representing multivalued mappings using density models*. It is a machine learning method that learns trajectory IK given a training set of input-output pairs $(\mathbf{x}, \boldsymbol{\theta})$ and possibly but not necessarily given the forward mapping $\mathbf{f}$. It is a global method in that it disambiguates multiple branches by minimising a trajectory-wide constraint. The goal of the method is to obtain a $\boldsymbol{\theta}$-trajectory that, while not perfectly accurate, is sufficiently close at each point to the correct trajectory that it can be refined (if desired) using a local method. We describe the method in section II, demonstrate it in experiments in section III and discuss it in section IV.

II. Trajectory inverse kinematics by conditional density modes

Assume we have a training set of pairs $(\boldsymbol{\theta}, \mathbf{x})$ with $\mathbf{x} = \mathbf{f}(\boldsymbol{\theta})$, where $\mathbf{f}$ is the forward kinematics mapping (if we do not know $\mathbf{f}$, we could estimate it by fitting e.g. a neural net). At run time, assume we are given a trajectory $\mathbf{x}_1, \ldots, \mathbf{x}_N$ in workspace and want to obtain a trajectory $\boldsymbol{\theta}_1, \ldots, \boldsymbol{\theta}_N$ of joint angles that yields the $\mathbf{x}$-trajectory while avoiding discontinuous jumps in $\boldsymbol{\theta}$-space. Our method works as follows. *Offline*, it estimates a density model $p(\boldsymbol{\theta}, \mathbf{x})$ for both variables, or just a conditional density $p(\boldsymbol{\theta}|\mathbf{x})$, using the training set. At *run time*, for each $n = 1, \ldots, N$ we obtain the conditional density $p(\boldsymbol{\theta}|\mathbf{x}_n)$ and its modes; the latter explicitly represent the multiple inverse solutions at each $\mathbf{x}_n$. Then, we obtain the $\boldsymbol{\theta}$-trajectory by minimising a constraint over the entire set of modes. Let us describe each step in detail. This method is a particular case of a more general approach proposed for reconstruction of sequences with missing data [16].

A. Density model

The key property of the conditional distribution $p(\boldsymbol{\theta}|\mathbf{x})$ is that it will be peaked around the inverse solutions for $\mathbf{x}$ (see fig. 1). Thus, its modes are representatives of the inverse values. If we represent the density with a Gaussian mixture, then we can approximate the true data density to arbitrary accuracy by using a sufficiently large number of components [20]. Thus, the modes can approximate the inverse values as closely as desired (at a corresponding computational cost). A more interesting issue is that the topology of the manifold of inverse branches can be very complex, where the number of inverse branches may depend on the value of $\mathbf{x}$; for example, at a singularity two different branches intersect (fig. 1). The density can deal with these topology changes in that e.g. two different modes can merge into a single one, and vice versa. This is a useful property that frees us from having to guess the number of inverses and fix it beforehand.

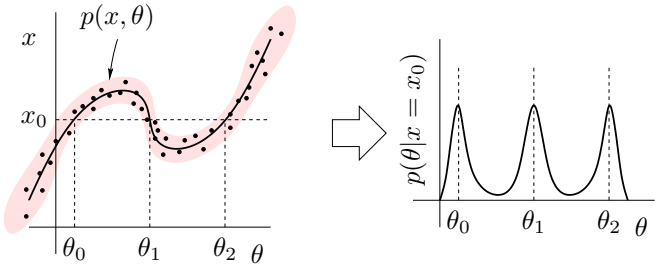

Fig. 1. *Left*: joint probability density $p(x, \theta)$ (shaded area) and forward mapping f (thick black line). *Right*: multivalued mapping $\theta = f^{-1}(x_0) = \{\theta_0, \theta_1, \theta_2\}$ from the multimodal conditional distribution $p(\theta|x)$. Depending on the value of x_0, there may be 3 modes (as shown), 2 or 1, and correspondingly 3, 2 or 1 inverses for x_0.

We can estimate the conditional density with statistical machine learning techniques given the dataset of pairs $(\boldsymbol{\theta}, \mathbf{x})$. One option is to learn a *full* density model $p(\mathbf{x}, \boldsymbol{\theta})$ and then obtain from it the conditional distribution

$$p(\boldsymbol{\theta}|\mathbf{x}) = p(\mathbf{x}, \boldsymbol{\theta})/p(\mathbf{x}) = p(\mathbf{x}, \boldsymbol{\theta})/\textstyle\int p(\mathbf{x}, \boldsymbol{\theta})\, d\boldsymbol{\theta}.$$

A convenient density model for which computing conditional distributions is straightforward are isotropic Gaussian mixtures (GM), $p(\mathbf{y}) = \sum_{m=1}^{M} \pi_m p(\mathbf{y}|m)$ where $p(\mathbf{y}|m) \sim \mathcal{N}(\mathbf{y}; \boldsymbol{\mu}_m, \sigma^2\mathbf{I})$. The GM parameters π_m, $\boldsymbol{\mu}_m$ and σ^2 (proportion, mean and covariance) are estimated from the training set by maximum likelihood with the EM algorithm [20]. Another option is to learn directly a *conditional* density model $p(\boldsymbol{\theta}|\mathbf{x})$; this is more efficient because it needs to model a density in fewer dimensions (only $\boldsymbol{\theta}$, not $\boldsymbol{\theta}$ and $\mathbf{x}$). We study both choices (full and conditional) in section III.

B. Mode finding

Assume we have a conditional density $p(\boldsymbol{\theta}|\mathbf{x})$ which has the form of a GM. Efficient algorithms for finding all the modes of a GM exist [17] that iterate a hill-climbing algorithm from every centroid of the GM. In particular, the Gaussian mean-shift algorithm iterates

$$\boldsymbol{\theta}^{(\tau+1)} = \textstyle\sum_{m=1}^{M} p(m|\boldsymbol{\theta}^{(\tau)}; \mathbf{x})\boldsymbol{\mu}_m(\mathbf{x})$$

where the posterior probability $p(m|\boldsymbol{\theta}^{(\tau)}; \mathbf{x})$ is the normalised version of

$$\pi_m(\mathbf{x})\, \exp\big(-\tfrac{1}{2}\, \|(\boldsymbol{\theta}^{(\tau)} - \boldsymbol{\mu}_m(\mathbf{x}))/\sigma_m(\mathbf{x})\|^2\big).$$

Gaussian mean-shift does not require inverting matrices and takes $\mathcal{O}(kM^2)$ where M is the number of components in the GM and k the average number of iterations per component. The computational time can be drastically reduced, for example discarding low-probability components of the GM (having small $\pi_m(\mathbf{x})$); see [18] for other accelerations.

C. Global optimisation with dynamic programming

Assume we have collected for each step n in the trajectory all the modes (candidate inverses). In principle, each of these modes represents a correct solution for step n (following a certain solution branch), but a given branch that is valid for part of the trajectory may be invalid for another part (e.g. because certain joint angles' values are forbidden due to mechanical constraints). In order to determine the solution, we minimise a global, trajectory-wide constraint over the set of modes. In this paper, we consider a constraint of the form $\mathcal{C} + \lambda\mathcal{F}$ (for $\lambda \geq 0$), where:

- $\mathcal{C} = \sum_{n=1}^{N-1} \|\boldsymbol{\theta}_{n+1} - \boldsymbol{\theta}_n\|$ represents a *continuity constraint* (integrated 1st derivative). This penalises discontinuous jumps in $\boldsymbol{\theta}$-space and encourages short trajectories. We also use $\mathcal{S} = \sum_{n=1}^{N-2} \|\boldsymbol{\theta}_{n+2} - 2\boldsymbol{\theta}_{n+1} + \boldsymbol{\theta}_n\|$, which represents *smoothness* (integrated 2nd derivative).
- $\mathcal{F} = \sum_{n=1}^{N} \|\mathbf{x}_n - \mathbf{f}(\boldsymbol{\theta}_n)\|$ represents a *forward constraint* (integrated workspace error), and penalises invalid inverses, i.e., modes $\boldsymbol{\theta}_n$ that do not map near the desired $\mathbf{x}_n$. This helps to eliminate spurious modes produced by ripple in the density model.

Effectively, this is a form of planning in angle space. *Global* minimisation of the constraint can be obtained by dynamic programming in $\mathcal{O}(N\nu^2)$ where ν is the average number of modes per step (usually very small), thus in linear time on the trajectory length N. Computationally, this is generally negligible compared to the mode-finding step.

III. Experiments

We show proof-of-concept experiments for several simple robot arms. Our goal is to illustrate the methods' performance with known ground truth for different settings. We consider the following methods: the Jacobian pseudoinverse (local method baseline); a conditional mean method, which estimates a univalued inverse mapping (as a neural net would do); and our conditional modes method, which estimates multivalued mappings and disambiguates the solution by minimising a global constraint. We study different choices of the density model (full and conditional) and of the global constraint ($\mathcal{C}$, $\mathcal{S}$, $\mathcal{F}$).

A. Planar 2-link robot arm

First, we consider a 2-dof planar robot arm (fig. 2) for which it is possible to visualise the conditional density and study the method. The forward mapping is

$$x_1 = l_1 \cos\theta_1 + l_2 \cos(\theta_1 + \theta_2)$$
$$x_2 = l_1 \sin\theta_1 + l_2 \sin(\theta_1 + \theta_2)$$

where $l_1 = 0.8$ and $l_2 = 0.2$. The inverse mapping can be computed analytically and has 2 solutions (elbow up/down). Singularities occur when $|\mathbf{J}(\boldsymbol{\theta})| = |l_1 l_2 \sin\theta_2| = 0 \Leftrightarrow \theta_2 = 0,\ \pm\pi$, i.e., when the arm is fully stretched or folded. To make the problem more complex, we limit the $\boldsymbol{\theta}$-domain to $[0.3, 1.2] \times [1.5, 4.7]$ rad so that certain branches are invalid in certain regions of the workspace. For example, the region at the right end of the workspace is only reachable as elbow-up, and the region at the left end as elbow-down. More generally, the feasible $\boldsymbol{\theta}$-domain could have a very complicated shape, where the range of allowed values for a single angle θ_i depends on the values of other angles θ_j, e.g. to avoid self-intersections in a humanoid robot. Respecting these constraints is simple in our method, since the training set will only contain feasible configurations by construction, and the density modes will always lie on high-density regions (not so the mean!). The trajectory in fig. 2 goes through singularities (when the arm is fully stretched); a local method may choose a branch that later on is unable to reach the trajectory, but our method can choose the correct branch by keeping track of all local solutions and then disambiguating them with the global constraint.

We generated a training set of 2 000 pairs $(\boldsymbol{\theta}, \mathbf{x})$ by uniformly sampling the $\boldsymbol{\theta}$-space[1] and mapping with $\mathbf{f}$ (black dots in fig. 2). We trained density models by max. likelihood:

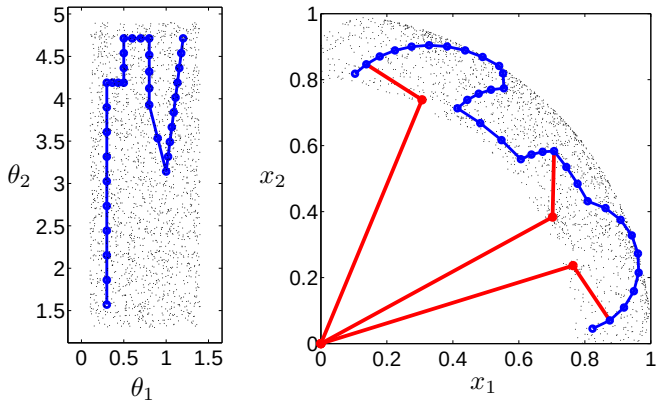

Fig. 2. Geometry of the 2-link planar arm of sec. III-A (left: $\boldsymbol{\theta}$-space, right: $\mathbf{x}$-space). The black dots are the training set of pairs $(\boldsymbol{\theta}, \mathbf{x}) \in \mathbb{R}^4$, which indicate the reachable region of workspace. The blue curve is the trajectory to be reconstructed, and the red lines schematically represent the robot arm in 3 different configurations. Points near the two ends of the workspace can only be reached by one configuration because of limits on θ_1.

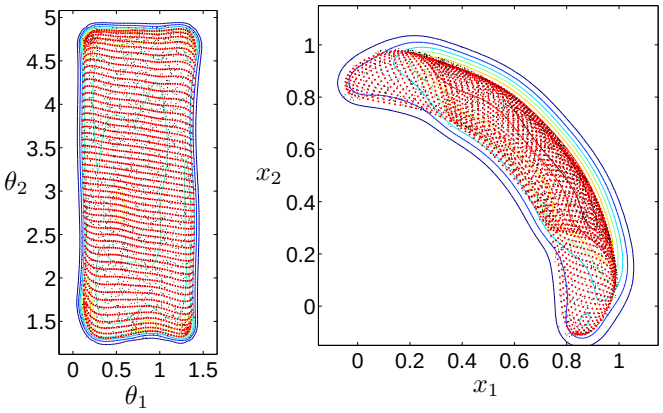

Fig. 3. Marginal densities $p(\boldsymbol{\theta})$ (left) and $p(\mathbf{x})$ (right) for the fine GTM model $p(\boldsymbol{\theta}, \mathbf{x})$ (4–dimensional), as a contour plot. The component centres of the Gaussian mixture are indicated by red dots.

[1] We included samples in a slightly larger domain $[0.1, 1.4] \times [1.3, 4.9]$ to avoid boundary effects in the density model. For GTM, we also added a bit of noise (stdev 0.05) to improve the smoothness of the resulting density.

- Full density $p(\mathbf{x}, \boldsymbol{\theta})$: we could have trained a GM directly, but instead we trained a generative topographic mapping (GTM) model [19], since the intrinsic dimensionality of $(\mathbf{x}, \boldsymbol{\theta})$ is 2 (not 4, because of the forward mapping). GTM is a latent variable model that yields a GM constrained to lie in a low-dimensional manifold. We tried 2 GTM models, one coarse (with $M = 225$ components in the GM) and one fine (with $M = 2\,500$). Fig. 3 shows the resulting density, or rather the marginals $p(\boldsymbol{\theta})$ and $p(\mathbf{x})$ for visualisation purposes.
- Conditional density $p(\boldsymbol{\theta}|\mathbf{x})$: we used a mixture density network (MDN) [20]. This is a particular case of mixtures of experts [21] that yields a GM

$$p(\boldsymbol{\theta}|\mathbf{x}) = \sum_{m=1}^{M} \pi_m(\mathbf{x}) \mathcal{N}(\boldsymbol{\theta}; \boldsymbol{\mu}_m(\mathbf{x}), \sigma_m(\mathbf{x}))$$

where the functions $\pi_m(\mathbf{x})$, $\boldsymbol{\mu}_m(\mathbf{x})$ and $\sigma_m(\mathbf{x})$ are neural nets. We used $M = 2$ components and 2-layer neural nets with 10 hidden units. Note a MDN is different from a neural net; the latter is a univalued function, while the MDN represents multimodal densities, whose number of modes depends on $\mathbf{x}$.

Fig. 4 shows, for each model, the conditional density for a particular $\mathbf{x}$ value. The conditional density model (MDN) gives a sharply peaked density with 2 modes near the true inverses. The fine GTM model gives also a bimodal density but less sharp, and the coarse GTM model gives a multimodal density where spurious modes arise along the line connecting the true inverses. The reason for this is the interference from the additional dimensions (for $\mathbf{x}$) that GTM is modelling, so that more components are necessary to achieve an accurate conditional density. However, as seen below, all 3 models succeed in recovering the true trajectory thanks to the forward constraint $\mathcal{F}$ (which filters out the spurious modes).

Figs. 5–6 show the reconstructed trajectories for each density model (we obtained similar results with other trajectories). We also show the trajectory that results from using the mean of the conditional density. This yields the GM regression mapping and is essentially equivalent to fitting a neural net directly to pairs $(\mathbf{x}, \boldsymbol{\theta})$. Since it can only represent a univalued mapping, it averages the two inverse branches, resulting in the fully stretched configuration, which is invalid (i.e., it does not equal the desired $\mathbf{x}$) for most $\mathbf{x}$; it is valid where the inverse is univalued, namely at the ends of the workspace. When using conditional modes, all 3 density models (MDN, coarse GTM, fine GTM) succeed in reconstructing the true trajectory with good accuracy, but more importantly, yielding a globally correct trajectory that chooses the appropriate branch at all steps.

It is very interesting to note that the $\mathbf{x}$-trajectory of fig. 2 can actually be produced by different $\boldsymbol{\theta}$-trajectories (fig. 7). In theory, they all have exactly the same value for the global constraint, but in practice they differ slightly due to the particular training set and model used. The pseudoinverse method, being local, can only find one of these trajectories (fig. 5–6, green). In our method, the dynamic programming search considers all these trajectories and selects the one with globally minimal constraint value. However, if (say)

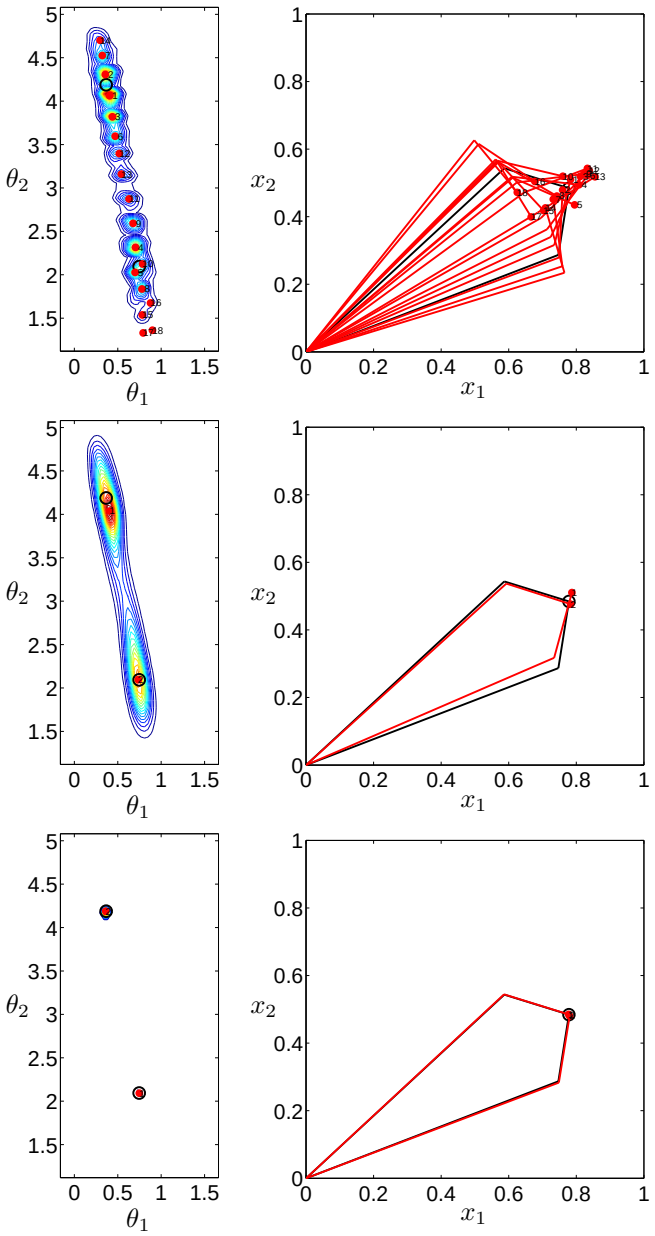

Fig. 4. Sample conditional densities $p(\boldsymbol{\theta}|\mathbf{x} = (0.78, 0.48))$ for 3 models: coarse GTM (top, 18 modes), fine GTM (middle, 2 modes) and MDN (bottom, 2 modes). *Left*: contours of the conditional density in $\boldsymbol{\theta}$-space, its modes (red dots) and the true inverses (black circles). *Right*: robot arm configurations for the modes (red) and true inverses (black).

the region $[1, 1.5] \times [1.5, 2]$ of $\boldsymbol{\theta}$-space were not allowed (e.g. because of mechanical constraints) then the trajectory found by the pseudoinverse method would be invalid; a local method has to decide which inverse branch to take at the singularity near $\boldsymbol{\theta} = (0.3, 3)$ and does not benefit from the information about the forbidden $\boldsymbol{\theta}$–rectangle that lies in the future (assuming the trajectory starts near $\boldsymbol{\theta} = (0.3, 1.6)$). Our method does benefit from it by learning (through the training set) only those regions and branches that are actually feasible and succeeds in reconstructing the correct trajectory.

Table I gives the errors in $\boldsymbol{\theta}$ and $\mathbf{x}$ wrt the true trajectory (true = any of fig. 7). For $\mathbf{x}$ they are of around 2% of

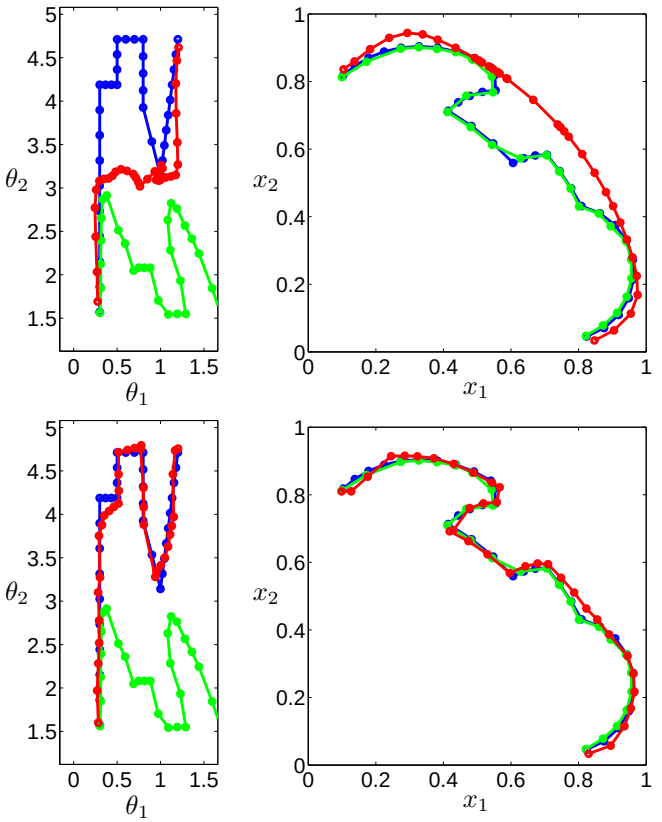

Fig. 5. True (blue) and reconstructed trajectories with the fine GTM model (red) and the pseudoinverse (green). *Left*: $\boldsymbol{\theta}$-space, *right*: $\mathbf{x}$-space. *Top*: using the conditional mean, *bottom*: using the modes and the continuity constraint $\mathcal{C}$. The pseudoinverse solution is one of the trajectories of fig. 7.

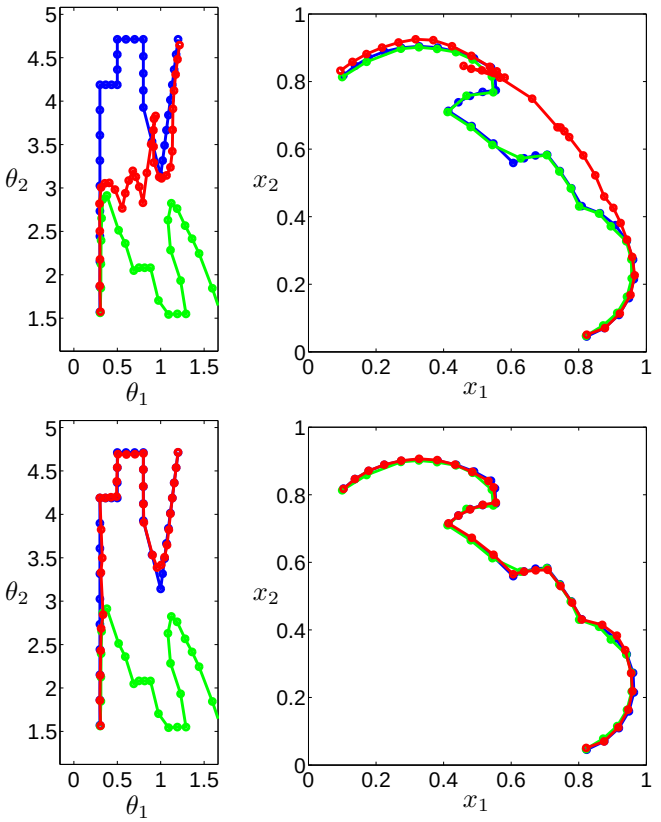

Fig. 6. As fig. 5 but for the MDN model (red).

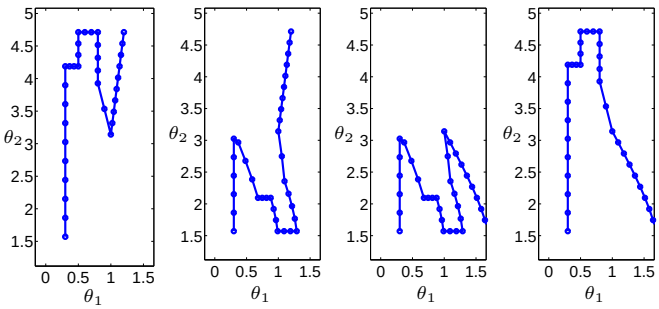

Fig. 7. Four trajectories in $\boldsymbol{\theta}$-space that produce the same $\mathbf{x}$-trajectory of fig. 2 (blue).

the length of the fully stretched arm ($l_1 + l_2 = 1$) for the fine and coarse GTM models, and of 0.5% for the MDN model. These errors are very close to the "oracle" column, which gives the error achieved if the closest modes to the true solution were selected. We could refine the trajectory and reduce the error as much as desired in a postprocessing step by initialising an analysis-by-synthesis search at each point in the trajectory. We find that the continuity constraint alone is enough to find the correct trajectory with the MDN and the fine GTM model, but not with the coarse GTM model, because of the spurious modes it has (which provide shortcuts that the continuity constraint favours). However, adding the forward constraint $\mathcal{F}$ as $\mathcal{C}+\lambda\mathcal{F}$ (over a wide range of $\lambda > 0$) yields the correct trajectory for all methods. The smoothness constraint $\mathcal{S}$ performs as well as the continuity constraint $\mathcal{C}$. The errors when using the mean of the density are considerably larger, but only the figures show how truly bad its solutions are.

Both the pseudoinverse and our method can achieve low reconstruction error, depending on the chosen number of iterations and of GM components. Besides its ability to ensure globally feasible trajectories, our method has the advantage of being less sensitive to singularities. Near singularities, the pseudoinverse method is numerically unstable and takes many iterations to converge (not so our method).

TABLE I

RECONSTRUCTION ERRORS FOR THE 2D ROBOT ARM

Angle reconstruction error $\frac{1}{N}\sum_{n=1}^{N}\|\boldsymbol{\theta}_n - \hat{\boldsymbol{\theta}}_n\|$ (rad)

Model	mean	oracle	$\mathcal{C}$	$\mathcal{S}$	$\mathcal{C}+\lambda\mathcal{F}$	$\mathcal{S}+\lambda\mathcal{F}$
coarse GTM	0.783	0.083	0.628	0.704	0.118	0.122
fine GTM	0.798	0.114	0.114	0.127	0.114	0.127
MDN	0.668	0.037	0.037	0.037	0.037	0.037
pseudoinv	0.06					

Workspace reconstruction error $\frac{1}{N}\sum_{n=1}^{N}\|\mathbf{x}_n - \mathbf{f}(\hat{\boldsymbol{\theta}}_n)\|$

Model	mean	oracle	$\mathcal{C}$	$\mathcal{S}$	$\mathcal{C}+\lambda\mathcal{F}$	$\mathcal{S}+\lambda\mathcal{F}$
coarse GTM	0.084	0.024	0.094	0.097	0.022	0.028
fine GTM	0.084	0.021	0.021	0.021	0.021	0.021
MDN	0.072	0.005	0.005	0.005	0.005	0.005
pseudoinv	0.016					

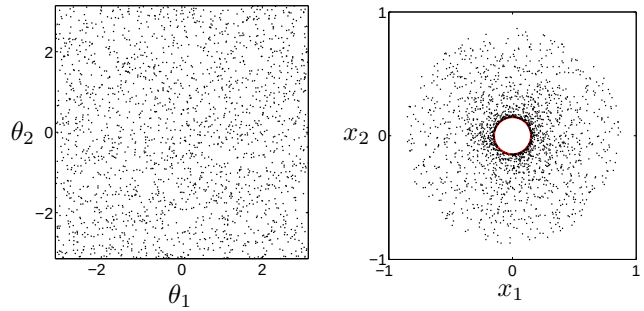

Fig. 8. Training set for the PUMA 560 robot arm of sec. III-B (top views, left: $\boldsymbol{\theta}$-space, right: $\mathbf{x}$-space). The workspace contains an unreachable region shaped like a vertical cylinder passing through the robot foot.

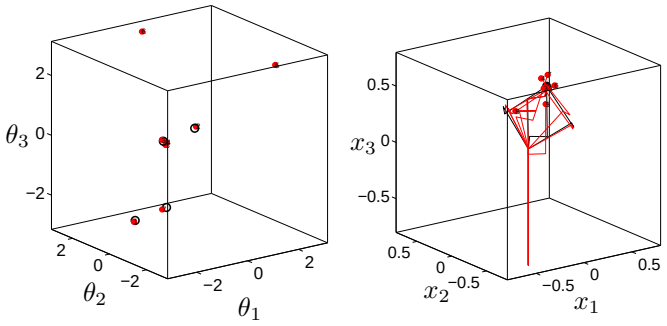

Fig. 9. *Left*: modes (red dots) for the conditional density $p(\boldsymbol{\theta}|\mathbf{x})$ for the MDN model and the PUMA 560 robot arm. There are 4 true inverses (black circles), which are well represented by the modes, but there are also two spurious modes (which are removed by the forward constraint $\mathcal{F}$, since they map far from the desired $\mathbf{x}$). *Right*: modes and true inverses in workspace, represented as schematic arms.

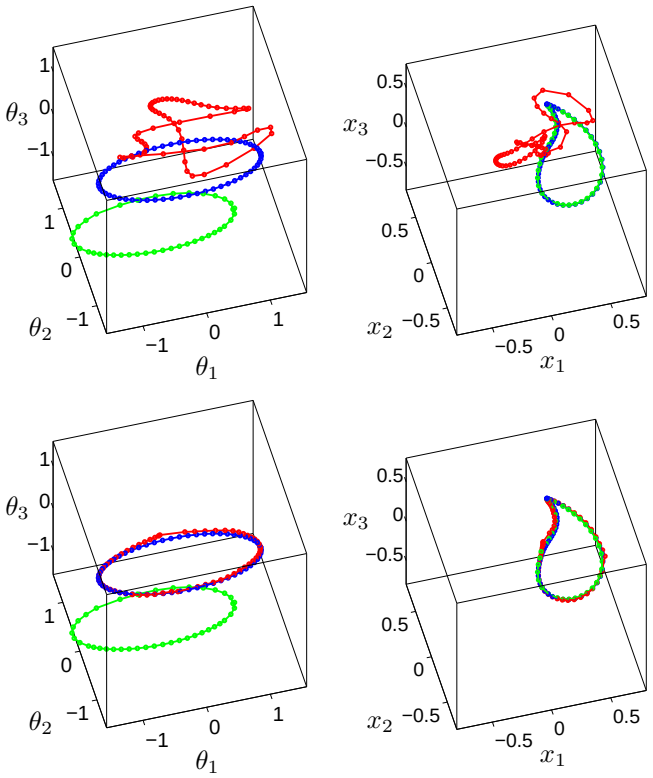

Fig. 10. Reconstruction of an elliptical trajectory (blue) for the PUMA 560 arm: MDN (red); pseudoinverse (green). *Left*: $\boldsymbol{\theta}$-space, *right*: $\mathbf{x}$-space. *Top*: mean of the density, *bottom*: modes and continuity constraint $\mathcal{C}$.

B. PUMA 560 robot arm with 6 DOF

Figs. 10–11 and table II show similar experiments for a PUMA 560 robot arm with 3 dof for position $\boldsymbol{\theta} = (\theta_1,\ \theta_2,\ \theta_3)$, 3 dof for orientation (which we ignore), and a 3D workspace $\mathbf{x} \in \mathbb{R}^3$. The (point) IK can be solved analytically for this robot [9] and yields 4 solution branches (two combinations of elbow up/down; fig. 9); we use the implementation of the Matlab Robotics Toolbox [22]. As before, we limit the angle domain in order to complicate the topology of the inverse mapping, and generate a training set of $5\,000$ pairs $(\boldsymbol{\theta}, \mathbf{x})$ (shown in fig. 8). The GTM (full density model) that we trained (results not shown) failed to produce a good reconstruction because of the existence of multiple spurious modes. The reasons for this are the higher dimensionality of the space, but also the fact that GTM is practically limited to an intrinsic dimensionality of at most 2, while in this case the intrinsic dimensionality is 3. We also trained a MDN (conditional density model) with $M = 12$ components (and neural nets with 300 hidden units), which did succeed in reconstructing various trajectories, with errors of similar magnitude as with the planar arm of sec. III-A; we show a sample of results, for 3 trajectories (an elliptical closed loop, a figure–8 closed loop, and an open trajectory; figs. 10–11, table II). Again, the symmetry of the problem results in several equivalent global solutions; the pseudoinverse and our method choose different ones. The larger errors occur

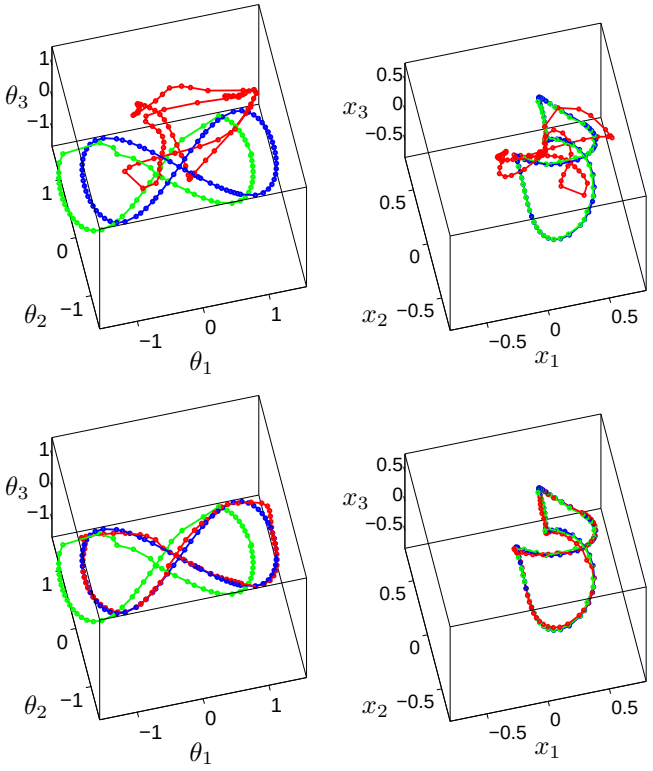

Fig. 11. As fig. 10 but for a figure–8 trajectory.

TABLE II

RECONSTRUCTION ERRORS FOR THE 3D ROBOT ARM (PUMA 560)

Angle reconstruction error $\frac{1}{N}\sum_{n=1}^{N}\|\boldsymbol{\theta}_n - \hat{\boldsymbol{\theta}}_n\|$ (rad)

Traj.	pseudoinv	mean	$\mathcal{C}$	$\mathcal{S}$	$\mathcal{C}+\lambda\mathcal{F}$	$\mathcal{S}+\lambda\mathcal{F}$
Ellipse	0.072	2.110	0.076	0.069	0.071	0.069
Figure–8	0.076	1.990	0.082	0.081	0.081	0.080
Open	0.042	2.140	0.173	0.778	0.173	0.176

Workspace reconstruction error $\frac{1}{N}\sum_{n=1}^{N}\|\mathbf{x}_n - \mathbf{f}(\hat{\boldsymbol{\theta}}_n)\|$

Traj.	pseudoinv	mean	$\mathcal{C}$	$\mathcal{S}$	$\mathcal{C}+\lambda\mathcal{F}$	$\mathcal{S}+\lambda\mathcal{F}$
Ellipse	0.025	0.819	0.030	0.029	0.029	0.029
Figure–8	0.019	0.750	0.028	0.027	0.027	0.027
Open	0.007	0.665	0.055	0.080	0.055	0.055

for points near a cylindrical hole at centre of the workspace which is not reachable by the robot, because of boundary effects of the density model. They could be reduced by increasing the number of components in the GM, or more efficiently by refining the trajectory with a local method. The "oracle" (best achievable) error (not shown) was very similar to that of $\mathcal{C}+\lambda\mathcal{F}$.

C. Redundant planar 3-link robot arm

When $\dim\boldsymbol{\theta} > \dim\mathbf{x}$ (redundant manipulator), an infinite number of inverses $\boldsymbol{\theta}$ exist for a given $\mathbf{x}$. The corresponding density $p(\boldsymbol{\theta}|\mathbf{x})$ would ideally be uniform over this set of inverses. Instead, because we use a Gaussian mixture, this uniform density becomes approximate and has multiple modes distributed over the set of inverses. Thus, these modes act as a quantised representation of the inverse set, and are available for use by the global constraint (which could also incorporate terms suggested by arguments of movement economy, such as integrated jerk or torque). We show this with a 3-link redundant manipulator with 3 dof for $\boldsymbol{\theta}$ (link lengths: 3, 2.5, 2; foot at $\mathbf{x}=\mathbf{0}$) and a 2D workspace $\mathbf{x}\in\mathbb{R}^2$. We generate a training set in a subset of $[0,2\pi]^3$ and train a MDN ($M=36$ components, neural nets with 300 hidden units). Figs. 12–13 and table III show experiments for three trajectories in $\mathbf{x}$–space (a circle, a loopy trajectory with self-intersections and a figure–8). The larger errors occur when the robot arm is close to fully-stretched configurations (corresponding to singularities). Both the pseudoinverse and our method are able to retrieve continuous (but different) trajectories in $\boldsymbol{\theta}$–space. As before, near singularities the pseudoinverse method is unstable and takes many iterations to converge.

TABLE III

RECONSTRUCTION ERRORS FOR THE REDUNDANT MANIPULATOR

Workspace reconstruction error $\frac{1}{N}\sum_{n=1}^{N}\|\mathbf{x}_n - \mathbf{f}(\hat{\boldsymbol{\theta}}_n)\|$

Model	pseudoinv	mean	$\mathcal{C}$	$\mathcal{S}$	$\mathcal{C}+\lambda\mathcal{F}$	$\mathcal{S}+\lambda\mathcal{F}$
Circle	0.060	4.610	0.173	0.185	0.140	0.161
Loopy	0.106	3.970	0.231	0.040	0.040	0.040
Figure–8	0.135	3.930	0.069	0.069	0.069	0.069

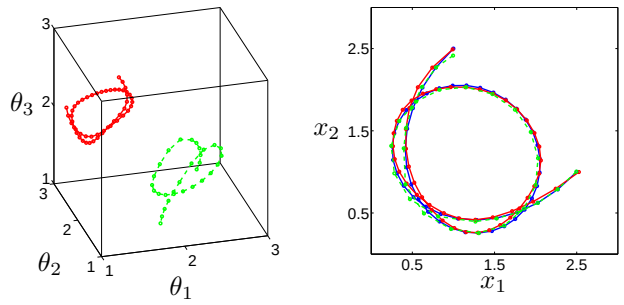

Fig. 12. Reconstruction of the loopy trajectory for the redundant arm: MDN with constraint $\mathcal{C}+\lambda\mathcal{F}$ (red); pseudoinverse (green).

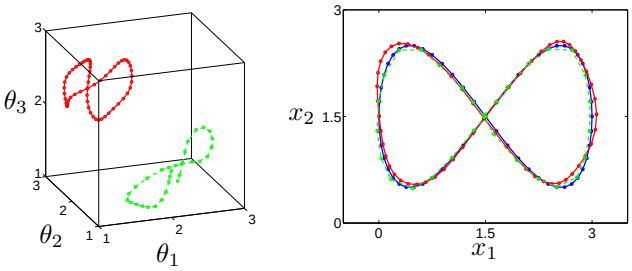

Fig. 13. As fig. 12 but for a figure–8 trajectory.

IV. DISCUSSION

Our method, by directly representing multivalued mappings and using a global constraint, is able to achieve feasible, globally correct solutions to trajectory IK even in the presence of (1) singularities of the Jacobian, where the forward mapping has multiple local inverses, and (2) complicated angle domains, which are captured through the training set. The power of the density model is its flexibility: in principle, it represents implicitly (through its modes) all the feasible solution branches once and for all, even when their topology can be very complex (e.g. with a number of branches that depends on $\mathbf{x}$) because of the nonlinearity of the forward mapping, or because of mechanical constraints. The disadvantage is that the mappings are implicit, and must be made explicit at run time by mode finding. We discuss several aspects of the method next.

Data collection: In common with other machine learning methods, we need a training set of pairs $(\boldsymbol{\theta},\mathbf{x})$. These can be collected by sampling the $\boldsymbol{\theta}$-space and computing $\mathbf{x}=\mathbf{f}(\boldsymbol{\theta})$, if the forward mapping $\mathbf{f}$ is known, or by recording $(\boldsymbol{\theta},\mathbf{x})$ while the robot is performing a task (perhaps imitating a human). This has the advantage of yielding valid pairs (by definition) and sampling only those areas of $\boldsymbol{\theta}$-space that correspond to *typical* motion, rather than feasible but atypical motions. Besides, typical behaviour may result in correlations between joints that reduce the intrinsic dimensionality of the $\boldsymbol{\theta}$-space. This idea is being exploited in motion-capture systems and has wide applicability in IK in graphics [2] and articulated pose tracking in computer vision [23], [24]. The density model need not be overly accurate; it suffices to yield modes near the true solution, and spurious modes (if there are not too many of them) may be filtered out by the forward constraint. Being data-driven, a limitation of

our method is that the estimated global trajectories are not perfectly accurate, however they are very close to the true trajectory and may be refined a posteriori if desired by a local method (e.g. RMRC or analysis-by-synthesis).

Run time: In practice, the run time is dominated by the mode-finding step, which takes $\mathcal{O}(kM^2)$ where M is the number of components in the GM and k the average number of iterations per component (≈ 50). When using a full density model $p(\mathbf{x}, \boldsymbol{\theta})$, M is very large, which prevents use in real time. But with a conditional density model $p(\boldsymbol{\theta}|\mathbf{x})$ (e.g. a MDN), which besides is more accurate, we can limit M to a number slightly larger than the (estimated) maximum number of solution branches for all $\mathbf{x}$, which is far smaller. The mode-finding algorithms, e.g. Gaussian mean-shift, can also be significantly accelerated [18], again noting that there is no need to converge with large accuracy. Our method does not use the Jacobian and needs no matrix inversions. In our unoptimised Matlab implementation for the PUMA arm, our method took $50/10/4$ ms per point (worst/average/best), while the pseudoinverse method took $200/30/10$ ms.

In summary, our method can obtain very accurate solutions if a GM with a large enough number of components is used. However, its main strength is in being able to find a globally feasible solution without suffering from singularities of the Jacobian (since it does not use the Jacobian or possibly even a closed-form forward mapping), and dealing in a natural way (through the training set) with complex angle domains that are very difficult to express in analytical form.

V. Conclusion

We have introduced a machine learning method for trajectory IK that can deal with trajectories containing singularities, where the inverse mapping changes topology, and with complicated angle domains caused by mechanical constraints (e.g. to prevent self-intersection of body limbs in a humanoid robot)—a hard problem for local methods (e.g. Jacobian pseudoinverse). Given a training set $(\boldsymbol{\theta}, \mathbf{x})$, the method learns a conditional density $p(\boldsymbol{\theta}|\mathbf{x})$ (using a mixture density network, MDN) that implicitly represents the branches of the inverse mapping $\boldsymbol{\theta} = \mathbf{f}^{-1}(\mathbf{x})$; the mappings are obtained by finding the modes of the conditional density using a Gaussian mean-shift algorithm, and the final $\boldsymbol{\theta}$-trajectory is obtained by minimising a global, trajectory-wide constraint over the set of modes. We have demonstrated the method with trajectory IK for simple robot arms (e.g. PUMA 560) with known forward and inverse mappings. Future work will apply it to trajectory IK in other domains (such as animation in computer graphics, articulated pose tracking in computer vision or articulatory inversion in speech), where neither the inverse nor possibly the forward mappings are known, and having complex mechanical constraints that are best captured by data-driven approaches. Another advantage of the method is its probabilistic nature: it can model noise in the measured $\boldsymbol{\theta}$, $\mathbf{x}$ and estimate the uncertainty in the reconstructed trajectory (error bars); it is also applicable when some of the $\mathbf{x}$ variables are missing or unspecified (e.g. for a humanoid robot we might not care about the hand position when walking).

VI. Acknowledgements

MACP thanks Sethu Vijayakumar and Marcelo Kallmann for valuable discussions. Work funded by NSF CAREER award IIS–0754089.

References

[1] J. Craig, *Introduction to Robotics. Mechanics and Control*, Addison-Wesley, Reading, MA, 1989.

[2] K. Grochow, S. Martin, A. Hertzmann, and Z. Popović, "Style-based inverse kinematics," *ACM Trans. Graphics*, vol. 23, no. 3, pp. 522–531, Aug. 2004.

[3] J. Schroeter and M. Sondhi, "Techniques for estimating vocal-tract shapes from the speech signal," *IEEE Trans. Speech and Audio Process.*, 1994.

[4] C. Qin and M. Á. Carreira-Perpiñán, "An empirical investigation of the nonuniqueness in the acoustic-to-articulatory mapping," in *Proc. Interspeech*, 2007.

[5] R. Kolodny, L. Guibas, M. Levitt, and P. Koehl, "Inverse kinematics in biology: The protein loop closure problem," *Int. J. of Robotics Research*, vol. 24, no. 2–3, 2005.

[6] C. A. Klein and C. H. Huang, "Review of pseudoinverse control for use with kinematically redundant manipulators," *IEEE Trans. Systems, Man, and Cybernetics*, vol. SMC–13, no. 2, , 1983.

[7] D. DeMers and K. Kreutz-Delgado, "Canonical parameterization of excess motor degrees of freedom with self-organizing maps," *IEEE Trans. Neural Networks*, vol. 7, no. 1, 1996.

[8] A. D'Souza, S. Vijayakumar, and S. Schaal, "Learning inverse kinematics," in *Int. Conf. Intelligent Robots and Systems (IROS'01)*.

[9] R. P. Paul and H. Zhang, "Computationally efficient kinematics for manipulators with spherical wrists based on the homogeneous transformation representation," *Int. J. of Robotics Research*, vol. 5, no. 2, 1986.

[10] D. E. Whitney, "Resolved motion rate control of manipulators and human prostheses," *IEEE Trans. Man-Machine Systems*, vol. 10, no. 2, 1969.

[11] A. Liegeois, "Automatic supervisory control of configuration and behavior of multibody mechanisms," *IEEE Trans. Systems, Man, and Cybernetics*, vol. 7, no. 12, 1977.

[12] Y. Nakamura and H. Hanafusa, "Optimal redundancy control of robot manipulators," *Int. J. of Robotics Research*, vol. 6, no. 1, 1987.

[13] D. P. Martin, J. Baillieul, and J. M. Hollerbach, "Resolution of kinematic redundancy using optimization techniques," *IEEE Trans. Robotics & Automation*, vol. 5, no. 4, 1989.

[14] M. Jordan and D. Rumelhart, "Forward models: Supervised learning with a distal teacher," *Cognitive Science*, vol. 16, no. 3, 1992.

[15] D. DeMers and K. Kreutz-Delgado, "Learning global properties of nonredundant kinematic mappings," *Int. J. of Robotics Research*, vol. 17, no. 5, 1998.

[16] M. Á. Carreira-Perpiñán, "Reconstruction of sequential data with probabilistic models and continuity constraints," in *Advances in Neural Information Processing Systems (NIPS)*, 2000.

[17] M. Á. Carreira-Perpiñán, "Mode-finding for mixtures of Gaussian distributions," *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 22, no. 11, 2000.

[18] M. Á. Carreira-Perpiñán, "Acceleration strategies for Gaussian mean-shift image segmentation," in *Proc. of the 2006 IEEE Computer Society Conf. Computer Vision and Pattern Recognition (CVPR'06)*, 2006.

[19] C. M. Bishop, M. Svensén, and C. K.Williams, "GTM: The generative topographic mapping," *Neural Computation*, vol. 10, no. 1, 1998.

[20] C. M. Bishop, *Neural Networks for Pattern Recognition*, Oxford University Press, 1995.

[21] R. A. Jacobs, M. I. Jordan, S. J. Nowlan, and G. E. Hinton, "Adaptive mixtures of local experts," *Neural Computation*, vol. 3, no. 1, 1991.

[22] P. I. Corke, "A robotics toolbox for MATLAB," *IEEE Robotics & Automation Mag*, vol. 3, no. 1, 1996.

[23] R. Urtasun, D. Fleet, A. Hertzmann, and P. Fua, "Priors for people tracking from small training sets," in *Proc. 10th Int. Conf. Computer Vision (ICCV'05)*, 2005.

[24] M. Á. Carreira-Perpiñán and Z. Lu, "The Laplacian Eigenmaps Latent Variable Model," in *Proc. of the 11th Int. Workshop on Artificial Intelligence and Statistics (AISTATS 2007)*, 2007.