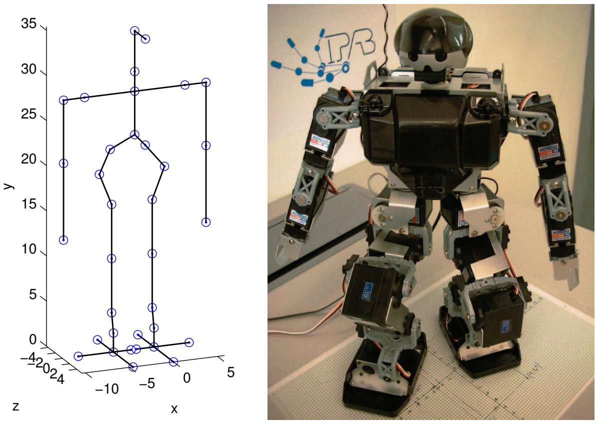
2010 IEEE International Conference on Robotics and Automation
Anchorage Convention District
May 3-8, 2010, Anchorage, Alaska, USA

Geodesic trajectory generation on learnt skill manifolds

Ioannis Havoutis Subramanian Ramamoorthy
Institute of Perception, Action and Behaviour
School of Informatics, University of Edinburgh, Edinburgh, EH8 9AB, UK
I.Havoutis@sms.ed.ac.uk, S.Ramamoorthy@ed.ac.uk

***Abstract*— Humanoid robots are appealing due to their inherent dexterity. However, these potential benefits may only be realized if the corresponding motion synthesis procedure is suitably flexible. This paper presents a flexible trajectory generation algorithm that utilizes a geometric representation of humanoid skills (e.g., walking) - in the form of skill manifolds. These manifolds are learnt from demonstration data that may be obtained from off-line optimization algorithms (or a human expert). We demonstrate that this model may be used to produce approximately optimal motion plans as geodesics over the manifold and that this allows us to effectively generalize from a limited training set. We demonstrate the effectiveness of our approach on a simulated 3-link planar arm, and then the more challenging example of a physical 19-DoF humanoid robot. We show that our algorithm produces a close approximation of the much more computationally intensive optimization procedure used to generate the data. This allows us to present experimental results for fast motion planning on a realistic – variable step length, width and height – walking task on a humanoid robot.**



(a) Model (b) Robot

Fig. 1. The *KHR-1HV* humanoid robot used, (a) skeleton model and (b) physical robot.

I. Introduction

In recent years, humanoid robot platforms have been receiving increasing attention due to their inherent dexterity and great flexibility. Correspondingly, this highlights the need for general purpose motion planners. Off the shelf solutions for humanoid robot behaviours are often restricted to a limited motion vocabulary that does not exploit the full capacity of the system. For instance, predesigned motions in many platforms are not parameterized in a flexible way (e.g., allowing full control over step length, width and height) and impose a limited discretization on the reachable space of the robot. There is a pressing need for efficient algorithms that can overcome these limitations and achieve a relatively rich set of within-skill variations in a realistic and practically implementable setting. Given such algorithms, one could then treat the skill as a component in a higher level discrete search [1]. Standard approaches that do allow for such flexibility tend to be computationally expensive, e.g., requiring high dimensional numerical optimization or c-space search. We need a more efficient alternative.

In realistic domains, e.g. RoboCup, where restrictions to variations on a skill would adversely impact higher level planning goals, one seeks a compact representation of the family of possible motions of a particular skill. This means that one would like to be able to learn and compactly represent the whole continuum of possible solutions for a particular task. In a machine learning setting, where one is acquiring a skill from demonstration, this raises the need for good generalization to solutions that possibly lie beyond the region of support of the original demonstration. Many existing data-driven approaches to humanoid motion synthesis are often limited in this respect - either they focus on interpolation within narrow regions near dense demonstrated samples or learning is posed as a problem of parameter tuning of an externally imposed path planning algorithm that may not naturally exploit the underlying structure of the space of solutions. We aim to make progress in this setting, by developing an algorithm that has better generalization properties and also a more natural and tighter integration between learning and planning.

In this setting, one way to obtain training data could be from demonstrated trajectories by an expert [2]. In this case notions such as optimality are intrinsic to the expert's demonstrations and can be based on a variety of (sometimes unmodelled) factors [3]. In order to have a better understanding of the behaviour of the algorithm, in this paper, we utilize demonstration data that is obtained from another computational solution which involves numerical optimization. These solutions are computationally expensive and not feasible for online operation. However, they can serve the same role as demonstration data. With this, we have a clear idea of the specific optimality properties of each task being considered, and a measure of algorithm performance against reasonable 'ground truth'.

As known from the study of biological behaviours, natural systems utilize synergies and coordination strategies that allow for efficient locomotion and fast planning. Biological strategies usually have a musculoskeletal basis that is inher-

978-1-4244-5040-4/10/$26.00 ©2010 IEEE

ent to the dynamics of the system, that restricts movement to a subset of all possible solutions. In a robotics context, system and (possibly artificial) task constraints can serve the same purpose. Robotics [4], [5] and graphics [6] researchers have utilized this fact to devise efficient motion synthesis strategies. Some recent works [7], [8], [9] also address this issue by considering how task space constraints, e.g., end-effector constraints, can be used to structure planning in configuration space with local Jacobian mappings. However the low-dimensional nature of the solutions may not always be taken into account explicitly.

The machine learning literature includes many examples of dimensionality reduction methods used to abstract and/or make problem spaces manageable. For example Chalodhorn et al. [10] use a low-dimensional sensory-motor mapping to optimize demonstrated motions over the robot's dynamics. Wang et al. [11] introduced the GPDM, a Gaussian processes based dimensionality reduction with a dynamical model of the evolution of the state, that can learn models of human kinematic trajectories. In the same spirit, Bitzer et al. [12] use a Gaussian Process-based nonlinear dimensionality reduction technique to arrive at an underlying model of demonstrated data, while using a parameterized path generation method over the learnt representation to generate novel movements.

Our goal is to learn a geometric structure, i.e., a skill manifold, that naturally and directly specifies both the low dimensional structure and dynamics on this subspace (which, in other works, one often externally and rather arbitrarily imposed).So, if one begins with a set of motion examples from a specific class, e.g., due to a path optimization or redundancy resolution principle or even a more complex kinodynamic constraint, then one seeks a representation that intrinsically captures both the restriction of states to a low-dimensional space and the evolution of the trajectories in that space. We achieve this by representing motions in terms of skill manifolds (learnt from data) where the tangent spaces are suitably defined so that geodesics correspond exactly to the execution of the desired motion.

II. MANIFOLD LEARNING

In this section we present the nonlinear manifold learning method that form the basis of our method. Our algorithm is a modification of Locally Smooth Manifold Learning by Dollar et al. [13], which we have adapted with robot motion-specific issues in mind. In particular we have replaced the neighborhood graph creation process with a procedure that considers task space distances as well as ensures that temporal neighborhood relations along the demonstrated trajectories are respected.

In the usual formulation, manifold learning is aimed at finding an embedding or 'unrolling' of a nonlinear manifold onto a lower dimensional space while preserving metric properties such as inter-point distances. Popular examples include MDS [14], LLE [15] and ISOMAP [16]. However, much of this work has been focused on summarization, visualization or analysis that explains some aspect of the observed data.

On the other hand, we are interested in preserving properties of trajectories in the data set.So, formally our goal is to learn a model of the tangent space of the low-dimensional nonlinear manifold, conditioned on the adjacency relations of the high dimensional data. The learnt manifold can be used to compute geodesic distances, to find projections of points on the manifold and to directly generate geodesic paths between points.

A. Learning the model

Given that our D-dimensional data lies on a locally smooth d-dimensional manifold in D-dimensional space, where $d < D$, there exists a continuous bijective mapping $\mathcal{M}$ that converts low dimensional points $y \in \mathbb{R}^d$ from the manifold, to points $x \in \mathbb{R}^D$ of the high dimensional space,

$$x = \mathcal{M}(y).$$

The goal is to learn a mapping from a point on the manifold to its tangent basis $\mathcal{H}(x)$,

$$\mathcal{H}:\ x \in \mathbb{R}^D \mapsto \left[\frac{\partial}{\partial y_1}\mathcal{M}(y)\cdots\frac{\partial}{\partial y_d}\mathcal{M}(y)\right] \in \mathbb{R}^{D\times d}$$

where each column of $\mathcal{H}(x)$ is a basis vector of the tangent space of the manifold at y, i.e. the partial derivative of $\mathcal{M}$ with respect to y.

Learning a model of the mapping with some parametrization θ, i.e. $\mathcal{H}_\theta$, is done as follows. Given two neighboring points on the manifold, x^i and x^j [1], the difference between these points, $\Delta^i_{.j}$, should be a linear combination of the tangent vectors at that point on the manifold, scaled by an unknown alignment factor. Taking $\Delta^i_{.j}$ to be the centered estimate of the directional derivative at $\bar{x}^{ij}$ and ϵ^{ij} to be the unknown alignment factor, we have

$$\mathcal{H}_\theta(\bar{x}^{ij})\epsilon^{ij} \approx \Delta^i_{.j},$$

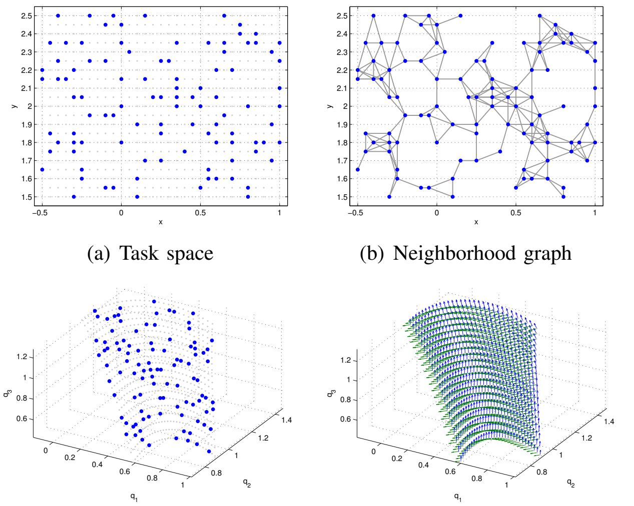
that holds given ϵ is small enough and the manifold can be locally approximated with a quadratic form. To learn $\mathcal{H}_\theta$ we define the error function:

$$err(\theta) = \min_{\{\epsilon^{ij}\}} \sum_{i,j\in N^i} \left\|\mathcal{H}_\theta(\bar{x}^{ij})\epsilon^{ij} - \Delta^i_{.j}\right\|_2^2,$$

where N^i is the set of neighbors of x^i. This minimization problem for θ is solved with a regularization term that ensures that the ϵ's do not get too large, that the tangents do not get too small and that neighboring tangent basis are aligned. For a precise model of the tangent space one would need to compute the tangent basis for each point, $\mathcal{H}_\theta(\bar{x}^{ij})$, which can be considered as a regression over the evidence (training data), and compute the alignment factors, ϵ^{ij}, for all neighboring points. Solving for the bases and their alignment simultaneously is complex, but if either one is kept constant, solving for the remaining variables becomes a tractable least squares problem.

Modeling $\mathcal{H}_\theta$ is done with a linear model of radial basis functions (RBF's) with features over the evidence [14], where

[1] Where superscript i and j are used for indexing.



(a) Task space (b) Neighborhood graph

(c) Joint space (d) Tangent space

Fig. 2. Learning the optimality manifold of a 3-link arm. (a) The planar task space of the arm and subsampled points (blue) used for leaning. (b) The neighborhood graph used for learning a manifold. (c) The optimality manifold that we wish to learn. Light gray points are not used for learning but are plotted to give a better estimate of the geometry of the manifold. Note that the manifold is not planar but twist and turns as we move down the q_3 axis. (d) The learnt tangent space model. Blue and green arrows are basis vectors evaluated at points that correspond to the original grid.

the number of basis functions, f, acts as parameter that can control the smoothness of the estimated mapping. More nonsmooth nonlinear manifolds with abrupt changes, would typically require more basis functions to ensure a tight local fit, though the generalization ability may be weakened. Optimizing the model requires alternating between the two least squares problems described above, until a local minima has been reached. Typically more than one random restart is performed to avoid local minima.

B. Optimal geodesic paths

By approximating the tangent space of the manifold, we gain access to a variety of geometric operations. Central to our robotics aims is the ability to compute paths through configuration space that lie on the low dimensional manifold. In this spirit, we now change our notation of points from x to q, to denote poses a robot can achieve in a configuration space.

Formally, our goal is to find the shortest path between two prespecified poses $q_1,\ q_n \in \mathbb{R}^D$, D being the dimensionality of the configuration space, that respects the geometry of the learnt manifold. In a robotics context, being on the manifold essentially means that the constraints (e.g., optimality w.r.t. a particular task-specific cost) inherent in the training data are respected. In practice we, discretize our path into a set of n via points, $\mathbf{q} = q_1, \ldots, q_n$, with the q_1 and q_n being fixed, and we follow a combination of gradient descent steps to minimize the length of the path while not leaving the support of the manifold.

The initial estimate of the shortest path is computed by interpolating between q_1 and q_n, while following the geometry of the manifold, until the distance between consecutive points is acceptable. Since we have learnt the tangent space of the manifold we can find a minimum energy solution that follows the orthonormal (to the manifold) component of the gradient of

$$err_{\mathcal{M}}(\mathbf{q}) = \min_{\{\epsilon^{ij}\}} \sum_{i,j \in N^i} \left\| \mathcal{H}_\theta(\bar{q}^{ij})\epsilon^{ij} - (q^i - q^j) \right\|_2^2,$$

that essentially makes the q^i's "stick" to the learnt manifold by iteratively moving them to points where neighboring (consecutive) bases are aligned. Next we apply another gradient descent optimization by following the parallel (to the manifold) component of

$$err_{length}(\mathbf{q}) = \sum_{i=2}^{n} \left\| q^i - q^{i-1} \right\|_2^2,$$

that iteratively minimizes the length of the path without leaving the support of the learnt manifold, while keeping the endpoints fixed.

The next sections present two examples of our method. The first example presents experiments on a simulated 3-link arm where both the manifold and the learnt model can be visualized and are representative of the core ideas behind this work. For the second example we use a physical humanoid robot, with which we demonstrate how our method scales to more complex systems and more challenging tasks.

III. Experiments on a robotic arm

Our first set of experiments were designed to elucidate the basic concepts underlying our approach. We have chosen a *3-link planar* arm where we can explicitly visualize both the configuration space and the optimization manifold. The arm is a series of three rigid links of unit length that are coupled with hinge joints, producing a redundant system with 3 degrees of freedom (DoFs) that is constrained to move on a 2 dimensional plane (task space).

A. Training data

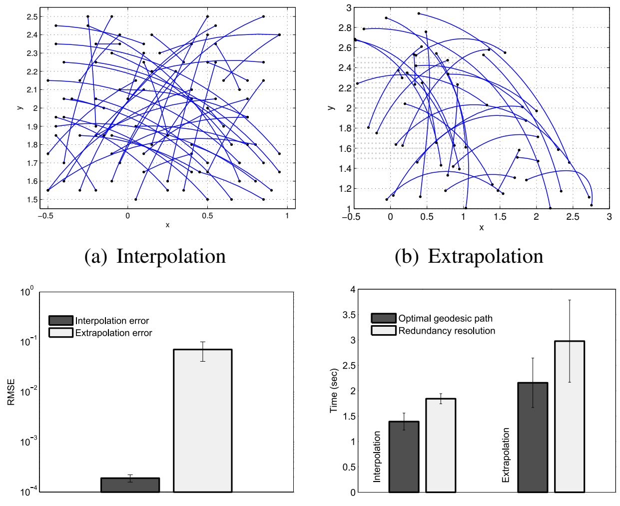
We start with a 21×31 grid in task space and compute the joint positions for each goal point with an iterative optimization procedure detailed below. We subsample 100 grid points to get a random permutation for learning, as in Fig. 2(a).

The system being redundant, we first have to choose a redundancy resolution strategy, which implicitly specifies the manifold that we will subsequently learn. Here, we choose the joint space configuration, $\mathbf{q}$, that minimizes the distance to a convenience (robot default or minimum strain) pose, $\mathbf{q}_c$. Formally,

$$\min \|\mathbf{q} - \mathbf{q_c}\|^2\,,\ \textit{subject to}\ f(\mathbf{q}) - \mathbf{x} = 0,$$

where f is the forward kinematics and $\mathbf{x}$ is the goal endpoint position on the plane.

The resulting $\mathbf{q}$'s trace a smooth nonlinear manifold in joint space, depicted in Fig. 2(c). We note that the manifold does not lie on a plane but on a convex strip that twists clockwise and tightens as we travel down the q_3 axis. Also different redundancy resolution strategies would produce



(a) Interpolation (b) Extrapolation

(c) Generalization errors (d) Time

Fig. 3. Results of the 3-link arm experiments. Novel task space trajectories produced with random start and end points where (a) demonstrates generalization within the region of support of the data, while (b) demonstrates generalization beyond the region of support of the training data. (c) RMSE error of generated trajectories against ground truth for the two cases. In the interpolation scenario the error is practically zero (y axis in log-scale). (d) Absolute planning time for the two cases. Note that in the interpolation case the length of the paths is consistently low.

different optimality manifolds. We note that, in general, this kind of information may not be explicitly known (in the case of human demonstration) or visualizable for more complex problems.

B. Implementation

The first step in data-driven learning of the desired manifold is to compute the neighborhood graph of the training data. We evaluate the task space distances to compute the neighborhood graph with the constraint that the graph contains a single connected component. In practice we gradually increase the neighborhood distance until all points are connected, as in Fig. 2(b).

The tangent space that we wish to learn is inherently two dimensional. We learn a model of $\mathcal{H}_\theta$ with 10 RBF's and 100 points, the blue points in Fig. 2(c). We can subsequently evaluate $\mathcal{H}_\theta$ at any point in our joint space. Fig. 2(d) shows the tangent bases evaluated at every point of the previously generated grid. Note that the basis vectors are aligned and vary smoothly, i.e. we obtain a good generalization within the region of support of the data.

C. Results

For measuring the goodness of our learnt manifold, we use two metrics. Central to our aims is the generalization ability of the model. Thus we quantitatively evaluate the error of planned motions against the poses that the original optimization procedure would produce. We distinguish between two scenarios for our motion planning. The first evaluates the model's interpolation ability, generating trajectories that in task space lie within the grid from which 100 points have been sampled for learning. The second case evaluates the extrapolation ability of the model by generating trajectories, the endpoints of which lie outside the original grid. In both cases start and endpoint positions in task space were random, while results are averaged over 10 trials for each scenario.

We create 50 optimal geodesic paths, with random start and end points for each case, with the method detailed in section II-B. Samples of such paths for both generalization cases are depicted in Fig. 3(a) and (b) (grid points in light gray for comparison).

We then collect all the intermediate points and compute the optimal solutions of their forward kinematics with the redundancy resolution algorithm detailed in section III-A, as ground truth. We compute the *RMSE*, for each trial and for each case, between ground truth and prediction of model, for a total of 10 trials.

The averaged errors are depicted in Fig. 3(c). Note that the *RMSE* axis is in log-scale while the difference of the two bars is of 2 orders of magnitude. To be precise the average *RMSE* for paths generated within the region of support of the data is $1.8935 \times 10^{-4} \pm 3.6013 \times 10^{-5}$(practically zero), while beyond the support of the data the average *RMSE* is $6.84 \times 10^{-2} \pm 2.19 \times 10^{-2}$. In addition, computing the optimal geodesic paths takes less time on average (Fig. 3(d) in both cases).

IV. Experiments on a humanoid robot

The three-link arm experiments are useful for demonstrating the working of the manifold learning and optimal geodesic path planning algorithm. We now move to a more complex system. In this setting, the skill manifold idea is more intuitively understood. We use the *KHR-1HV* (Fig. 1(b)), a "KidSized" humanoid robot[2] that stands approximately 35cm tall. It consists of 19 digital servo motors on brackets, in a bipedal-two-armed configuration, with a control board and a battery pack. The system is unstable as the center of mass is elevated.

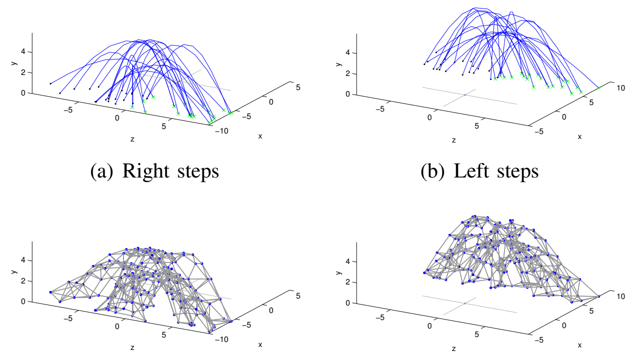
No analytical model of the dynamics of the system is available to us as. Obtaining such models is labor intensive. Moreover, even if we were to approximate such a model, it would have to account for varying model parameters, e.g. the change in the servos' behaviour as the battery gets depleted or the motor temperatures vary. These effects are hard to estimate, so we prefer to work directly from experimental data.

We focus on the task of walking, with the aim of generating a motion synthesis strategy that allows for full coverage of a reasonably large interval in step length. We begin with a redundancy resolution strategy that would yield training data and ground truth for our subsequent comparisons.

A. Training data

We frame the redundancy resolution strategy as an unconstrained nonlinear optimization problem. Algorithmically, we use a Quasi-Newton approach with a cubic line search procedure, based on the BFGS formula for iteratively updating

[2] According to the RoboCup Humanoid League size classification.



(a) Right steps (b) Left steps

(c) Right neighborhood graph (d) Left neighborhood graph

Fig. 4. *Task space* representation of the training data through forward kinematics. Random start and end point leg swing trajectories of the left (a) and right (b) legs. (c) and (d) the neighborhood graphs that result from the task space distances between demonstrated data (units in *cm*). This provides the task-specific distance metric for the high dimensional *joint-space*. Note that depicted here are only feet midpoint positions while the datasets consist of the joint space points that are 19-dimensional.

the estimate of the Hessian of the objective (cost) function [17]. Formally, the optimization problem is of the form

$$\min_{q} \mathcal{J}(\mathbf{q}), \text{ subject to } f(\mathbf{q}) - \mathbf{x} = 0,$$

where $\mathcal{J}$ is the cost function, f is the forward kinematics and $\mathbf{x}$ is a goal task space position. The cost function is a mixture of task constraints and stability constraints. The cost function evaluates:

- the distance of the midpoint of the swing foot to the desired goal
- the alignment of the swing foot with the x and y versors, to keep the foot flat
- the horizontal distance of the position of the pelvis to the desired pelvic position, to manipulate the center of mass of the humanoid
- the alignment of the waist of the robot with the z versor, to keep the humanoid, from the hips up, in an upright position

The optimization initialization pose is one where the humanoid stands upright with the knee joints slightly bent.

To generate a walking trajectory we start with the desired task space path of the swing leg and the position of the pelvis, and discretize to 20 waypoints. The swing foot trajectories are straight lines from start to goal points while the height of the foot is regulated with a sinusoid, scaled to a prespecified height. In practice we set the position of the pelvis to be over the support foot and perform a double support weight shift step once the swing leg has reached the goal position. Last we run the optimization procedure detailed earlier, and get the joint space trajectory of the leg swing and the weight swift phases for each complete task space step path.

The optimization results are approximately constant speed quasi-static trajectories, in the sense that inertial effects are negligible. We collected 20 full body joint space trajectories for stepping with the right leg and the same amount for stepping with the left leg. Start and goal points of every step have been randomized within a reasonable reaching distance. Figure 4(a) and 4(b) show the task space trajectories of each swing leg by running the datasets through the forward kinematics (the support foot is in light gray for comparison).

B. Implementation

Compared to our previous simpler example, this is higher dimensional space and sampling is necessarily somewhat sparse. Of the 19 DoFs of the robot we used the 12 DoFs of legs and hips and kept the remaining arm joints at a constant pose. Furthermore we separated each footstep to a swing phase and a weight shift phase. This way we divided the learning into two components, leg swing manifold and support weight shift manifold, as the measure of optimality is essentially different for each phase.

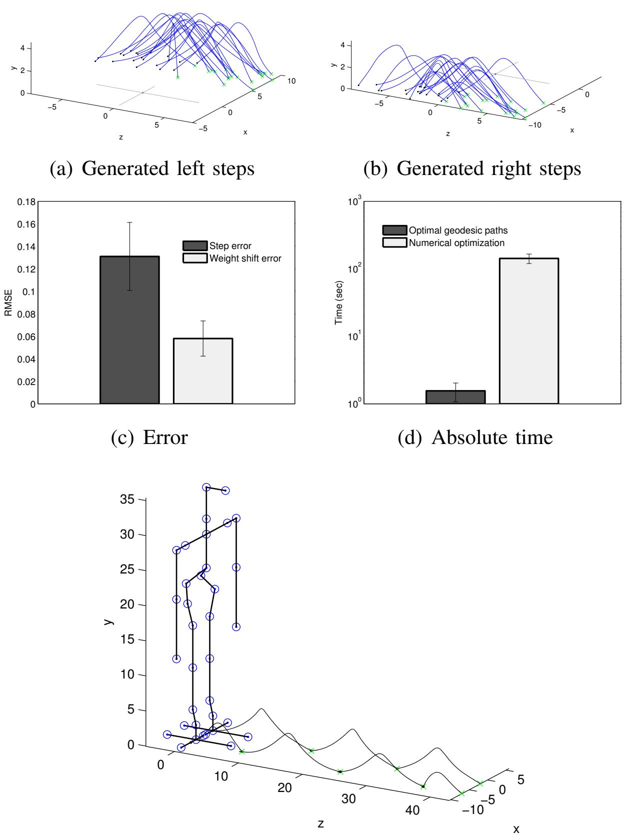
We begin with the same neighborhood graph computation procedure where we gradually increase our neighborhood distance until the graph is not disconnected (Fig 4(d) and 4(c)). We set the dimensionality of the manifolds to be 3, corresponding to the natural task space of the robot (see section V). In all learnt manifolds we used models with 20 RBF's and 400 data points that belong to 20 random task space trajectories as described in the previous section.

C. Results

The learnt manifolds are able to produce smooth walking trajectories that satisfy the optimization criteria used to produce the training data. Specifically, the average *RMSE* (*degrees*) of the leg swing manifold for the ground truth was as low as 0.12 while the average *RMSE* of the weight shift manifold ranged on average near 0.06 (Fig. 5(c)). This implies that the geometry of the step manifold is more complex and some of its features might be smoothed over by the RBF model. Nonetheless the procedure was able to produce stable walking in the continuum of the reaching space of the robot as depicted in Fig. 5(a) and 5(b) for right and left swings accordingly.

One point to note is that the shape of the trajectories in task space is qualitatively different than the training data. This suggests that the learnt manifold indeed traces the true underlying geometry that the optimization procedure sculpts in the robot's joint space. In contrast the training data has been generated on a point by point basis, while the shape of the trajectories in the task space (sinusoid) has been artificially imposed, regardless of the intrinsic structure of the optimality surface. The geodesic paths that are generated are optimal with respect to the manifold's geometry and traverse the configuration space smoothly.

The absolute time needed to generate an optimal geodesic path on the pair of manifolds (swing leg and weight shift) from random start to random end points was approximately 1.5552 ± 0.4785 seconds (in a standard, not particularly fine-tuned, numerical implementation of the algorithm) whereas generating a trajectory with the optimization procedure, described in section IV-A required approximately *two minutes* on average. This is a *significant* decrease in absolute planning



(a) Generated left steps

(b) Generated right steps

(c) Error

(d) Absolute time

(e) Generated random walk

Fig. 5. Experimental results with the humanoid robot. Random start and end point trajectories for left (b) and right (b) leg swings that have been generated from our learnt manifold, via geodesic path optimization (units in *cm*). (c) *RMSE* (*degrees*) of generated data against ground truth. (d) absolute time needed for planning and optimization with our method and the nonlinear optimization method (y axis in logscale) described in the text (section IV-A). (e) Random walk generated by geodesic path optimization on the learnt manifolds for randomized task-space goals. Snapshots of the robot executing the motion in Fig. 6, see also accompanying video.

time, which makes it possible to deploy this algorithm in realistic application scenarios (e.g. RoboCup).

A randomized walk sequence entirely generated with our method is depicted in Fig. 5(e). Notice that the step lengths are varying and the step points are variable as well with respect to the x axis. Snapshots of this walk executed by the robot are shown in Fig. 6. Also see the video clip accompanying this paper.

V. Discussion

We have demonstrated how a machine learning technique for approximating a low-dimensional skill manifold may be tightly integrated with the process of trajectory generation. One of the important differences between the manifold learning algorithm as used here, and other versions of such algorithms coming out of domains such as vision, is that we utilize task space metrics to shape the geodesic computations on the (configuration space) manifold, and focus on preserving properties of the trajectories, and not just a point set.

In both examples presented, we have chosen d to have the dimensionality of the system's task space. The reasoning behind this choice is that there might be configurations that are close in joint space but far away in task space. Since our aim is to learn skill-specific manifolds, this seems natural.We could have used any $d < D$, but simpler models are preferred. Choosing the appropriate dimensionality falls under the bias-variance trade off, as discussed below.

We now make a few observations regarding limitations (hence, directions for future improvement) of the algorithm in its current form. In this work, we do assume that the skills may be represented by a subspace that is a single connected component. This is clearly not an issue for the 3-link arm example. However, in general, this may well be insufficient as the dimensionality of the system grows. The place where this plays a role is the neighborhood graph computation where by connecting two points that should not be connected we would obtain a skewed model. In practice, suitably dense sampling, or better still incremental sampling in appropriate regions, and a bit of algorithmic book keeping, would suffice to ensure that this aspect of the manifold structure is properly reflected.

Also, one must keep in mind that the manifold learning step is performed with an iterative algorithm, much like Expectation Maximization, that is randomly initialized and does not always guarantee a global minimum. So, learnt models may not be unique solutions. This may call for better model selection procedures - a topic for future development.

The number of RBF basis in our experiments was chosen empirically, thus is open to further improvement. A high number of RBF's would allow the model to capture more intricate local geometric structure of the manifold, but would impair its generalization ability. On the other hand a low number of RBF's may oversmooth the solution and lose much of the geometric variation present in the training data.

This is a bias-variance trade-off and could be handled with a cross-validation procedure. Such choices would need to be closely related to the geometric complexity of the manifold that one would like to learn. Also the use of the centered estimate of the directional derivatives implies that the expressive ability of the model would not be able to handle manifolds that cannot be locally approximated with a quadratic form. In practice highly nonlinear manifolds that vary wildly or have sudden cutoffs may not be suitable for learning, without additional treatment.

Finally, we assume that start and end points of each trajectory are known. For this we have used the redundancy resolution strategy used in generating the demonstrated data. There is no implicit mapping of task space goals to configuration space poses on the manifold per se, but in principle once the manifold is learned one can easily search for points that satisfy task space goals.

VI. Conclusions and Future Work

We have demonstrated how a manifold learning algorithm can capture the geometric properties of a low dimensional skill manifold that underlies a high dimensional dataset.

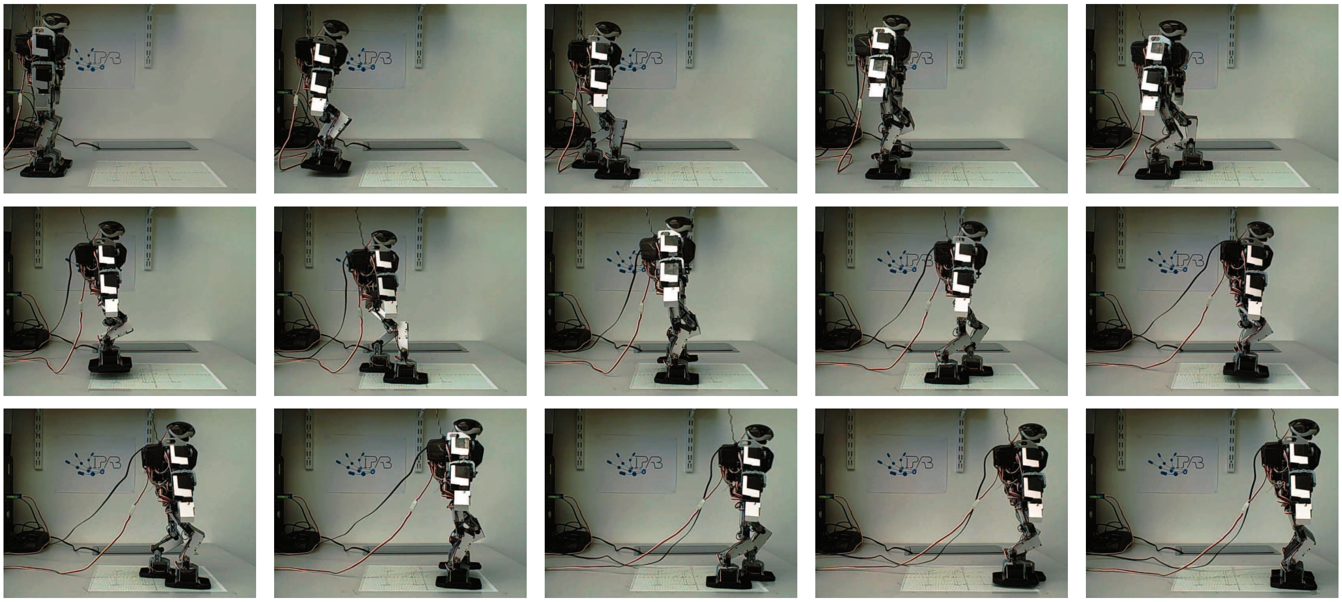


Fig. 6. Stills of the robot executing the planned motion depicted in Fig. 5(e).

We have also shown how this model can be naturally used to generate joint space trajectories, and how the generated trajectories reflect the optimality and constraints inherent in the training data.

We started with an example of a simulated robotic arm that is suitable for demonstrating the core concepts of our work and then demonstrated a similar result on a more interesting humanoid robot behaviour. We have demonstrated how manifolds of complex numerical optimization solutions can be learnt from sparse data and how the geometric structure generalizes within and beyond the support of the data. Finally, we have shown how such learnt manifolds can be used to produce novel approximately optimal solutions to continuous path planning queries in a very efficient and fast manner.

In future we aim to further extend our method for planning in the presence of kinodynamic constraints. Also we would like to add sensory feedback to the planning step as well as incorporate higher order terms, e.g. velocities and accelerations, in the state space. Our long term goal is to utilize the manifold learning and planning method as the the core of a larger system that would be able to learn, plan and execute motions robustly and in real time.

REFERENCES

[1] J. Chestnutt, M. Lau, K. M. Cheung, J. Kuffner, J. K. Hodgins, and T. Kanade, "Footstep planning for the honda asimo humanoid," in *Proceedings of the IEEE International Conference on Robotics and Automation*, April 2005.

[2] S. Schaal, A. Ijspeert, and A. Billard, "Computational approaches to motor learning by imitation," *Philosophical Transactions: Biological Sciences*, vol. 358, no. 1431, pp. 537–547, 2003.

[3] A. Coates, P. Abbeel, and A. Y. Ng, "Learning for control from multiple demonstrations," in *ICML '08: Proceedings of the 25th international conference on Machine learning*. New York, NY, USA: ACM, 2008, pp. 144–151.

[4] S. Ramamoorthy and B. J. Kuipers, "Trajectory generation for dynamic bipedal walking through qualitative model based manifold learning," *IEEE International Conference on Robotics and Automation (ICRA)*, pp. 359–366, May 2008.

[5] P. Isto and M. Saha, "A slicing connection strategy for constructing prms in high-dimensional cspaces," *Robotics and Automation, 2006. ICRA 2006. Proceedings 2006 IEEE International Conference on*, pp. 1249–1254, May 2006.

[6] A. Safonova, J. K. Hodgins, and N. S. Pollard, "Synthesizing physically realistic human motion in low-dimensional, behavior-specific spaces," *ACM Trans. Graph.*, vol. 23, no. 3, pp. 514–521, 2004.

[7] D. Berenson, S. Srinivasa, D. Ferguson, and J. Kuffner, "Manipulation planning on constraint manifolds," in *IEEE International Conference on Robotics and Automation (ICRA '09)*, May 2009.

[8] M. Stilman, "Task constrained motion planning in robot joint space," *Intelligent Robots and Systems, 2007. IROS 2007. IEEE/RSJ International Conference on*, pp. 3074–3081, 29 2007-Nov. 2 2007.

[9] T. Bretl, S. Lall, J.-C. Latombe, and S. Rock, "Multi-step motion planning for free-climbing robots," in *in WAFR*, 2004, pp. 1–16.

[10] R. Chalodhorn, D. Grimes, G. Maganis, R. Rao, and M. Asada, "Learning humanoid motion dynamics through sensory-motor mapping in reduced dimensional spaces," in *Robotics and Automation, 2006. ICRA 2006. Proceedings 2006 IEEE International Conference on*, May 2006, pp. 3693–3698.

[11] J. M. Wang, D. J. Fleet, and A. Hertzmann, "Gaussian process dynamical models for human motion," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 30, no. 2, pp. 283–298, 2008.

[12] S. Bitzer, I. Havoutis, and S. Vijayakumar, "Synthesising novel movements through latent space modulation of scalable control policies," in *Lecture Notes in Computer Science*. Springer Berlin / Heidelberg, 2008, pp. 199–209.

[13] P. Dollár, V. Rabaud, and S. Belongie, "Non-isometric manifold learning: Analysis and an algorithm," in *ICML*, June 2007.

[14] T. Hastie, R. Tibshirani, and J. H. Friedman, *The Elements of Statistical Learning*. Springer, August 2001.

[15] S. T. Roweis and L. K. Saul, "Nonlinear dimensionality reduction by locally linear embedding." *Science*, vol. 290, no. 5500, pp. 2323–2326, Dec 2000.

[16] J. B. Tenenbaum, V. de Silva, and J. C. Langford, "A global geometric framework for nonlinear dimensionality reduction." *Science*, vol. 290, no. 5500, pp. 2319–2323, Dec 2000.

[17] J. Nocedal and S. J. Wright, *Numerical Optimization*, 2nd ed. Springer, 2006.