Motion Generation with Geodesic Paths on Learnt Skill Manifolds*

Ioannis Havoutis and Subramanian Ramamoorthy

Abstract. We present a framework for generating motions drawn from parametrized classes of motions and in response to goals chosen arbitrarily from a set. Our framework is based on learning a manifold representation of possible trajectories, from a set of example trajectories that are generated by a (computationally expensive) process of optimization. We show that these examples can be utilized to learn a manifold on which all feasible trajectories corresponding to a skill are the geodesics. This manifold is learned by inferring the local tangent spaces from data. Our main result is that this process allows us to define a flexible and computationally efficient motion generation procedure that comes close to the much more expensive computational optimization procedure in terms of accuracy while taking a small fraction of the time to perform a similar computation.

1 Introduction

Humanoid robots provide a flexible platform for a variety of tasks including rough terrain locomotion and dexterous manipulation. Typically, this flexibility also carries the burden of increased complexity that adversely impacts the practical usability of such systems. For example, if a humanoid robot were tasked with locomotion on an uneven terrain - requiring the ability to continually vary foot placement positions in response to external events, it is hard to define a suitable motion generation strategy for two reasons. Firstly, ensuring that the motion satisfies all requirements ranging

Ioannis Havoutis
Department of Advanced Robotics, Istituto Italiano di Tecnologia, Genova
e-mail: I.Havoutis@iit.it

Subramanian Ramamoorthy
Institute of Perception, Action and Behaviour, School of Informatics,
University of Edinburgh
e-mail: s.ramamoorthy@ed.ac.uk

* An extended version of this paper appears in [4].

K. Mombaur and K. Berns (Eds.): *Modeling, Simulation and Optimization*, COSMOS 18, pp. 43–51.
DOI: 10.1007/978-3-642-36368-9_4 © Springer-Verlag Berlin Heidelberg 2013

from high-level planning goals to intermediate stability constraints and lower level actuator of joint constraints is a hard computational problem. Even if we had the computational resources, it can be hard to actually specify all of these requirements in a well posed analytical formulation. One way to approach such problems is by learning from demonstration trajectories. Here, the problem is to infer the *continuum* of trajectories in the solution space corresponding to a specific skill from a sparse set of demonstrated solutions. Additionally, we must represent this set in a way that allows generation of new motions, directed to previously unseen goals, that are consistent with prior experience.

While there are a number of different state of the art techniques for learning by demonstration algorithms, they all share some weaknesses with respect to this specific goal. Many existing methods focus on reproduction of patterns of movement for an end effector [1], without direct consideration of joint-space motion either to address constraints or exploit additional flexibility, or they focus on tasks where it is acceptable to define independent joint-level trajectories that can be simultaneously used in parallel [5]. While these are good for, say, reproducing human motions, they may not be well suited to the needs of a flexible motion generator in an autonomous system that must be deployed in a continually changing world.

Our approach represents each skill as a manifold that is embedded in the robot's joint space. This manifold represents the set of all possible solutions to a skill and it is inferred from a few example solutions to corresponding optimization problems (or, if available, human demonstrations). When presented with a planning query we can generate a path that is within this set, generated by computing the geodesic path over the manifold.

In this paper, in order to illustrate the behaviour of the algorithm, we utilize example trajectories that are obtained from a computational method 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 properties of each task being considered, and a measure of algorithm performance against reasonable 'ground truth'.

2 Learning for Motion Synthesis

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 [3], LLE [7] and ISOMAP [8]. 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. Such a learnt manifold model can then be used to compute geodesic distances, to find projections of points on the manifold and to directly generate geodesic *paths* between points.

2.1 Learning the Manifold

Our nonlinear manifold learning algorithm is based on Locally Smooth Manifold Learning by Dollar et al. [2], which we have adapted with robot motion specific issues in mind. In particular we have replaced the neighbourhood graph creation process with a procedure that considers task space distances as well as ensuring that temporal neighbourhood relations along the demonstrated trajectories are respected, similar to the procedure used in ST-ISOMAP [6].

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

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

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

We then learn a model of the mapping with parametrization θ, i.e. $\mathscr{H}_\theta$, based on the generalized neighbourhood relations of the data, N, and the centred estimate of the directional derivative between two neighbours, $\Delta^i_{\cdot j}$. The model is trained by minimizing the error function:

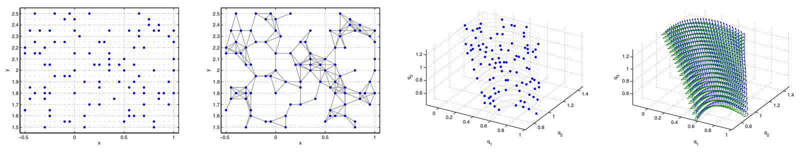
$$err(\theta) = \min_{\{\varepsilon^{ij}\}} \sum_{i,j\in N^i} \left\|\mathscr{H}_\theta(\bar{x}^{ij})\varepsilon^{ij} - \Delta^i_{\cdot j}\right\|_2^2,$$

where ε^{ij} is an unknown alignment factor and N^i is the set of neighbours of x^i.

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. Optimizing the model requires alternating between these two least squares problems, until a local minima has been reached. Typically more than one random restart is performed to avoid local minima [4].

2.2 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 *geodesic paths*; paths 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.



(a) Task space (b) NN graph (c) Joint space (d) Tangent space

Fig. 1 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 neighbourhood 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.

Our goal is to find the shortest path between two specified poses q_{start} and $q_{end} \in \mathbb{R}^D$, 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 satisfied. In practice, we discretize our path into a set of n via points, $\mathbf{q} = q_{start}, \ldots, q_{end}$, with q_{start} and q_{end} 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.

We first initialize a path by linearly interpolating between q_{start} and q_{end}, while following the geometry of the manifold, until the distance between consecutive points is acceptable. With the learnt tangent space we iteratively compute a minimum energy solution that makes q^is “stick” to the manifold and minimizes the length of the path without leaving the support of the manifold. The former is accomplished by following the orthonormal (to the manifold) component of the gradient of

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

and the latter by following the parallel (to the manifold) component of

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

while keeping the endpoints fixed.

The next sections present two examples of our method. The main thrust of our argument here is that the manifold representation provides a concise encoding of all motions corresponding to a skill. This encoding is equivalent to a computationally more expensive optimization process, but requires a fraction of the computational effort. We demonstrate this by first presenting a 3-dim motion problem, where the manifold can be easily visualized and the algorithm intuitively understood. Then, we show a more complex example involving a humanoid robot.

3 Experiments on a Robotic Arm

The planar 3-link 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 that is constrained to move on a 2 dimensional plane (task space).

The skill that we learn in this setting is the set of all solutions to a specific redundancy resolution scheme. 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, $\min \|\mathbf{q}-\mathbf{q_c}\|^2$, *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 points trace a smooth nonlinear manifold in joint space (Fig. 1(c)). Note that the manifold is not planar but lies on a convex strip that twists clockwise and tightens as we travel down the q_3 axis. Also, different redundancy resolution strategies would produce different optimality manifolds. In general, this kind of information may not be explicitly known (in the case of human demonstration) or visualizable for more complex problems.

We collect data (joint space points) from a grid in task space and subsample 100 points as our training set (Fig. 1(a)). We compute the neighbourhood graph from the task space distances and learn a model of $\mathscr{H}_\theta$ with 10 RBF's and 100 points, the blue points in Fig. 1(c). We can subsequently evaluate $\mathscr{H}_\theta$ at any point in our joint space. Fig. 1(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.

3.1 Evaluation

We evaluate the accuracy of the approximation that the learnt manifold provides in two generalization settings. One measures the interpolation ability, where we compare against ground truth data within the region of support of the training data, and the second demonstrates the extrapolation ability, where we compare what our model generates outside the region of support of the training data. We also record the time needed to produce the trajectories. In both cases we compare 50 trajectories with random start and end points that are produced with geodesic paths on the learnt manifold, against what the numerical optimization produces for the same goals. Samples of such paths for both generalization cases are depicted in Fig. 2(a) and (b) (grid points in light gray for comparison).

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. 2(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 $1.8935\times10^{-4}\pm3.6013\times10^{-5}$ (practically zero), while beyond the support of the data the average *RMSE* is $6.84\times10^{-2}\pm2.19\times10^{-2}$. In addition, computing the optimal geodesic paths takes less time on average (Fig. 2(d) in both cases).

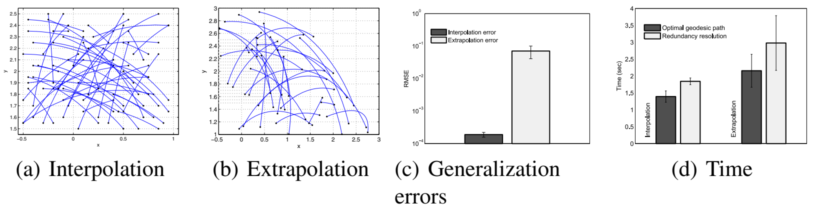

(a) Interpolation (b) Extrapolation (c) Generalization errors (d) Time

Fig. 2 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.

4 Experiments on a Humanoid Robot

To demonstrate the scalability of our approach we also present an example involving a humanoid robot platform. We use the *KHR-1HV* (Fig. 3(a)), that stands approximately 35cm tall and has 19 DoFs. We focus on the task of walking, with the aim of learning the manifold of quasi-static stepping trajectories for random

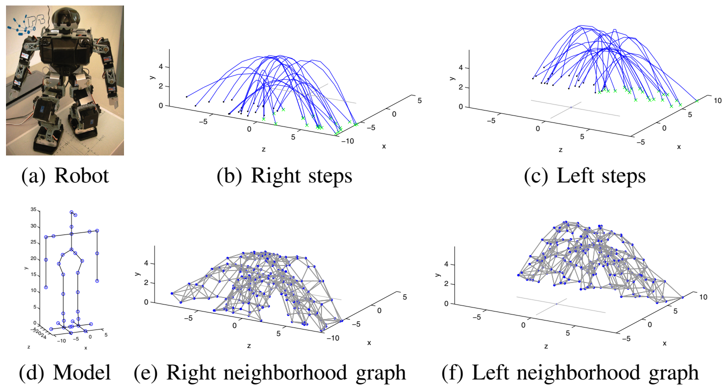
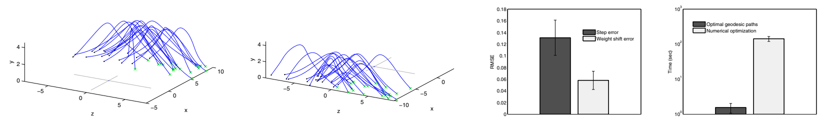

(a) Robot (b) Right steps (c) Left steps
(d) Model (e) Right neighborhood graph (f) Left neighborhood graph

Fig. 3 The *KHR-1HV* humanoid robot used, (d) skeleton model and (a) physical robot. *Task space* representation of the training data through forward kinematics. Random start and end point leg swing trajectories of the left (b) and right (c) legs. (e) and (f) the neighbourhood 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.



(a) Generated left steps (b) Generated right steps (c) Error (d) Absolute time

Fig. 4 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 log-scale) described in the text.

foothold placements, within a reasonably large step interval. We generate data with an unconstrained nonlinear optimization method that uses a hand-crafted cost function. Formally, the optimization problem is of the form $\min_q \mathscr{J}(\mathbf{q})$, *subject to* $f(\mathbf{q}) - \mathbf{x} = 0$, where $\mathscr{J}$ 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.

We collected 20 full body joint space trajectories where start and goal points of every step have been randomized within a reasonable reaching distance (Fig. 3(b) and 3(c)). 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 compute a neighbourhood graph (Fig 3(f) and 3(e)) and learn a manifold for each stepping phase. We set the dimensionality of the manifolds to be 3, being the simplest model that yields a low error.

4.1 Evaluation

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. 4(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. 4(a) and 4(b) for right and left swings accordingly.

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 required approximately *two minutes* on average, an approximately 98% increase in speed. This is a *significant* decrease in absolute planning time, which makes it possible to deploy this algorithm in realistic application

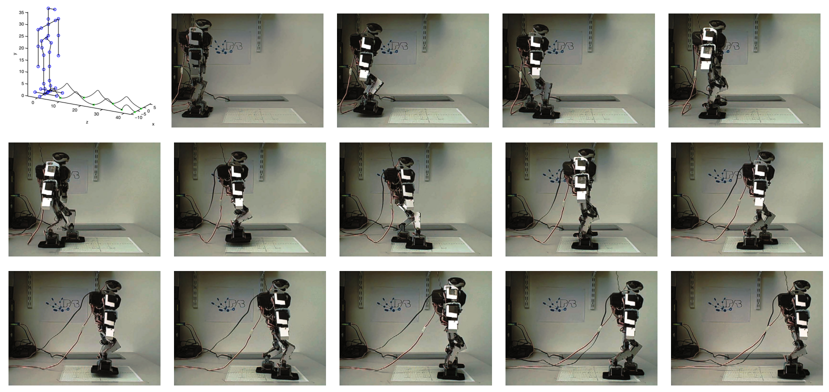


Fig. 5 Random walk generated by geodesic path optimization on the learnt manifolds for randomized task-space footholds and stills of the robot executing the planned motion

scenarios (e.g. RoboCup). A randomized walk sequence entirely generated with our method is depicted in Fig. 5, where foothold positions have been randomly generated and are previously unseen.

5 Conclusions and Future Work

We have demonstrated how a manifold representation can capture the flexibility required of a motion generation scheme operating in a continually changing environment. As used here, we have a computationally efficient procedure that can recover all of the solutions of a more expensive computational optimization procedure while also allowing for learning from data - where all requirements may not be easy to encode in an analytical framework. This work adds to the literature on learning by demonstration by addressing the cases where the task is more complex than simply reproducing specific task space trajectories and involves further kinodynamic requirements in the joint space, etc. We demonstrate this using a couple of robotics examples - a 3-link arm, where the results are easy to visualize, and a humanoid robot, where the stepping task is intuitively understood. Our long term goal is to utilize this procedure as part of a larger system that would be able to learn, plan and execute motions robustly and in real time.

References

[1] Calinon, S., D'halluin, F., Caldwell, D., Billard, A.: Handling of multiple constraints and motion alternatives in a robot programming by demonstration framework. In: IEEE-RAS International Conference on Humanoid Robots, Humanoids (2009)

[2] Dollár, P., Rabaud, V., Belongie, S.: Non-isometric manifold learning: Analysis and an algorithm. In: International Conference on Machine Learning, ICML (2007)
[3] Hastie, T., Tibshirani, R., Friedman, J.H.: The Elements of Statistical Learning. Springer (2001)
[4] Havoutis, I., Ramamoorthy, S.: Geodesic trajectory generation on learnt skill manifolds. In: Proceedings: IEEE International Conference on Robotics and Automation 2010, ICRA 2010 (2010)
[5] Ijspeert, A., Nakanishi, J., Schaal, S.: Trajectory formation for imitation with nonlinear dynamical systems. In: IEEE International Conference on Intelligent Robots and Systems (2001)
[6] Jenkins, O.C., Mataric, M.J.: A spatio-temporal extension to isomap nonlinear dimension reduction. In: International Conference on Machine Learning, ICML (2004)
[7] Roweis, S.T., Saul, L.K.: Nonlinear dimensionality reduction by locally linear embedding. Science 290(5500), 2323–2326 (2000)
[8] Tenenbaum, J.B., de Silva, V., Langford, J.C.: A global geometric framework for nonlinear dimensionality reduction. Science 290(5500), 2319–2323 (2000)