Motion Synthesis through Randomized Exploration on Submanifolds of Configuration Space

Ioannis Havoutis and 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. Motion synthesis for humanoid robot behaviours is made difficult by the combination of task space, joint space and kinodynamic constraints that define realisability. Solving these problems by general purpose methods such as sampling based motion planning has involved significant computational complexity, and has also required specialised heuristics to handle constraints. In this paper we propose an approach to incorporate specifications and constraints as a bias in the exploration process of such planning algorithms. We present a general approach to solving this problem wherein a subspace, of the configuration space and consisting of poses involved in a specific task, is identified in the form of a nonlinear manifold, which is in turn used to focus the exploration of a sampling based motion planning algorithm. This allows us to solve the motion planning problem so that we synthesize previously unseen paths for novel goals in a way that is strongly biased by known good or feasible paths, e.g., from human demonstration. We demonstrate this result with a simulated humanoid robot performing a number of bipedal tasks.

1 Introduction

One of the most significant recent trends in robotics is the push towards robust autonomy with complex robots such as humanoids. In principle, humanoid robots and other related architectures are highly versatile and capable of performing an unprecedented variety of tasks in applications ranging from service at home to rescue in rugged terrains. However, due to the inherent complexity of these systems, robotics researchers have struggled to realise this promise of robust and flexible operation in a multitude of environments. From the point of view of motion synthesis, i.e., the generation of feasible trajectories for all the joints of a robot given a *family* of task level goals such as, say, foot placement points, one of the big difficulties has been that of reconciling the need for efficient exploration of all possible ways to perform a family of tasks with the need for understanding of the intrinsic constraints that define realisability of the

J. Baltes et al. (Eds.): RoboCup 2009, LNAI 5949, pp. 92–103, 2010.

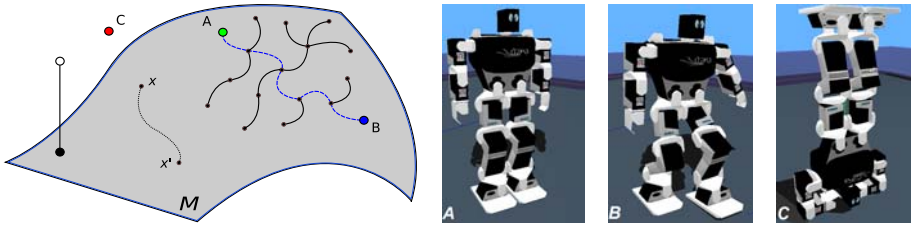

Fig. 1. *Left:* Schematic representation of a low-dimensional manifold as used in our algorithm. Combining an RRT planner with manifold learning focuses random sampling to a task relevant subspace. An RRT is grown by searching this manifold M, a nonlinear subspace of the configuration space. *A, B and C:* Taking a step with the humanoid is a process of following a path connecting configuration A (*green*) with configuration B (*blue*). The point C (*red*) represents a configuration that is not reached in any realisation of the task. Furthermore task specific geodesic distances can be computed, e.g. distance (*dotted line*) between x and x' and samples (*white*) lying close to the manifold can be projected onto it (*black*). The geodesic distance is used to calculate nearest neighbours and step sizes while projection guides sampling onto the task-relevant subspace.

task. Machine learning methods are efficient in capturing intrinsic task-specific constrains within restricted domains, i.e., focussing on properly interpolating between observed examples, while sampling-based motion planning methods are more focussed towards large-scale exploration of the global structure of configuration spaces. In the absence of specialised knowledge of task constraints, this can involve significant computational complexity.

In fact there are a number of tasks, e.g., locomotion in *RoboCup* domain where it is possible to get some human demonstration data but it is hard to explicitly characterize the implicit constraints that define the task. With this in mind, in this paper, we present an approach to motion synthesis that brings together two related but distinct algorithmic threads: sampling-based motion planning and manifold learning. We begin with a small set of example trajectories that are representative of the intrinsic constraints that define a task, e.g., bipedal walking. These trajectories are really just samples drawn from a set of possible trajectories that define a sub-manifold in the configuration space of the robot - indirectly defined by task space, joint space and kinodynamic constraints. We use a manifold learning algorithm to approximate this sub-manifold. In particular, our construction enables us to specify projections onto the manifold and also to compute geodesics. Then, as the robot is presented with different goals that appear in the course of its operation, we use a sampling based motion planning algorithm – Rapidly-exploring Random Trees (RRT) [1] – to synthesise novel trajectories that are restricted to lie on this sub-manifold.

A primary benefit of focussing exploration in this way is that it enables us to bring into the planning process constraints that are only known in terms of observed data from known good behaviours (i.e., not explicitly modelled). This makes our approach a data–driven one, wherein the constraint is inferred from

observed data and used in the planning process in the form of a sub–manifold onto which exploration is restricted. The learned sub-manifold provides a basis for higher level deliberation in a layered architecture. So, in addition to computational savings arising from focussed search, the learned model serves as an abstraction that succinctly encodes the variety of ways in which the underlying task may be performed - enabling different motion synthesis strategies.

The main contribution of this paper is the manifold-RRT algorithm, a novel extension to the RRT, which incorporates the focussed sampling idea mentioned above through a data-driven manifold learning algorithm. This enables us to synthesise high quality trajectories for bipedal robotic tasks such that the exploration is focussed to the neighbourhood of demonstrated behaviours. We first provide an overview of the motion planning and manifold learning algorithms as they relate to this work. Then, we describe the mRRT algorithm which combines the benefits of these two approaches. We demonstrate the applicability of this idea through experimental results with a simulated version of the KHR-2HV humanoid robot. Finally, we conclude with a brief discussion of how this specific result may be applied in more general settings involving humanoid and other robot behaviours.

2 Related Work

In the context of biological behaviours, it has been argued [2] and observed [2,3] that the curse of dimensionality is best overcome by utilising synergies and coordination strategies that enforce a restriction of the synthesised motions to low-dimensional spaces. Robotics [4,5,6] and graphics [7] researchers have utilised this fact to devise efficient motion synthesis strategies. Our interest is in incorporating this feature directly into sampling based motion planning. Some recent work [8,9,10] comes close to this issue by considering how task space constraints, e.g., end-effector constraints, can be used to structure search in configuration space with local Jacobian mappings. In other related work, e.g. references [11,12], the goal is to edit a statically stable trajectory, discovered by a sampling based motion planner, in a post-processing step to make the resulting trajectory dynamically realisable. However, the low-dimensional structure of the task is not directly leveraged in on-line planning. Computer animation researchers have arrived at closely related insights in developing structures such as motion-motif graphs [13] which try to abstract families of related trajectories into symbolic nodes so that on-line search is made efficient. However, in that work, the issue of task constraints is not given as much importance as in robotics and the focus is really on efficiently compressing a motion capture database.

3 Background

3.1 Rapidly Exploring Random Trees

Sampling-based motion planning algorithms are based on the idea of approximating the *free* portion of the configuration space by a suitable random structure

that enables efficient computation and fast exploration. The RRT [14] is a remarkably simple yet effective algorithm for planning a path between two points in configuration space.

In the algorithm, one adopts a simple set characterisation of the configuration space, which is the union of the free space, Q_{free} and the obstacle space Q_{obs}. Sampled configurations, q, are drawn from Q_{full}, $Q_{full} = Q_{free} \cup Q_{obs}$. Q_{full} can be the configuration space or the phase space for the system, or even just any composition of state variables within $q \in \mathbb{R}^D$, D being the dimensionality of the problem space.

We root a tree, T, at the given starting point, q_{init} and grow it by iterating the following process. Pick a random point $q_{rand} \in Q_{full}$ and calculate its distance from each point already in T. Select the closest point, q_{near}, from T and grow the tree toward q_{rand} by a step size Δx. Then evaluate if the resulting configuration $q_{new} = q_{near} + \Delta x_{q_{rand}}$ belongs to Q_{free} or Q_{obs}. If the former is true q_{new} is added to T, else the sample is discarded. The procedure is repeated until the goal configuration q_{goal} is reached, within some tolerance or number of iterations. The shortest path is then computed on T using a tree search algorithm. Algorithm 2 includes these core RRT (cRRT[1]) steps and is augmented with the LSML procedure, to be described.

RRTs quickly branch into unexplored regions of the space and when such regions become small the algorithm begins to fill in gaps with increasing resolution. This ensures that the planner is probabilistically complete, thus it will find a path if one exists as the number of samples grows to infinity. However, when considering complex problems involving humanoids, many finer points need consideration, including convergence to the goal, stability and realisability constrains, space coverage and resolution. For example, as a rule of thumb, in spaces with $D \geq 8$ convergence is typically slow. It has been shown that including a bias favouring the goal greatly increases the convergence speed as it steers the exploration [1].

In general, success of RRTs depends on the metric that is defined over the space to be explored. Traditionally a metric of the form:

$$d(q, q') = \sum_{i=1}^{n} w_i \left\| q_i - q'_i \right\|,$$

is used where the weights w_i denote the importance of each Degree of Freedom (DoF). These weights are often empirically chosen based on trial and error but as the dimensionality grows, and in nonlinear systems, this becomes difficult from intuition alone, so, there is a need for other ways to arrive such metrics. We argue that learning such a metric in a data-driven fashion is a desirable and scalable approach.

The second, related, issue that determines success of RRT-based planning is coverage. Random sampling in high dimensions can be excessively wasteful when the underlying task has special structure. The key issue is that sampling a high

[1] We term cRRT the classic RRT algorithm as described in [1].

Algorithm 1. Learn Manifold

1: **Lsml**(*tr_data*, *d*)
2: INPUT: kinematic task-relevant data *tr_data*, dimensionality of manifold *d*
3: OUTPUT: manifold *M*
4: $NN \leftarrow$ NN_GRAPH(*tr_data*)
5: $\theta \leftarrow$ OPTIMISE_PARAMETERS(*tr_data*, NN) {Model Parameters}
6: $M \leftarrow$ MINIMISE_MODEL_ERROR(θ) {Fit the manifold}
7: RETURN M

dimensional space densely enough is computationally infeasible. Knowing that many interesting robotic behaviours are restricted to low-dimensional subspaces [15,16,2,4,17], due to a variety of reasons including stability and energy constraints, joint limits and self-collision constraints, it is desirable to leverage this to achieve better coverage where it matters.

3.2 Manifold Learning

The machine learning literature includes many examples of dimensionality reduction methods used to abstract and/or make problem spaces manageable [18,19,16]. One of the big benefits of these methods is that they are data-driven and can be used in a scalable way in novel domains.

In the usual formulation, manifold learning is about 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 [20], LLE [21] and ISOMAP [22]. However, much of this work has been focused on summarisation, visualisation or analysis that explains some aspect of the observed data. Instead, we are more interested in methods that provide a direct representation of a nonlinear subspace in a way that enables standard geometric operations needed in motion planning. Such methods should work with demonstrated motions and provide good interpolation and extrapolation on the learnt manifold. For this, we choose a recently developed method – Locally Smooth Manifold Learning [23,24]. LSML explicitly focuses on generalising to unseen portions of the manifold, which is crucial for use with an exploration algorithm. The learnt manifold can be used to compute geodesic distances, to find projections of points on the manifold and to generate novel sample points. A detailed description of LSML, from [23], follows.

LSML. Given that our D-dimensional data lies on a locally smooth d-dimensional manifold, where $d < D$, there exists a continuous bijective mapping M that converts low dimensional points, $y \in \mathbb{R}^d$, to points, $x \in \mathbb{R}^D$, in the original high dimensional space. The goal is to learn a warping function W that can take a point on the manifold and compute its neighbouring points on the manifold, capturing the modes of variation of the data. Thus we can approximate W by M locally by defining $W(x, \epsilon) = M(y + \epsilon)$ where $y = M^{-1}(x)$ and $\epsilon \in \mathbb{R}^d$. The first order approximation of the above is $W(x, \epsilon) \approx x + \mathcal{H}(x)\epsilon$ where

each column $\mathcal{H}_{.k}$ of $\mathcal{H}(x)$ is the partial derivative of M with respect to y_k, i.e. $\mathcal{H}_{.k}(x) = \partial/\partial y_k M(y)$, valid given ϵ is small enough.

The objective then is to learn the unknown parameterised function $\mathcal{H}_\theta : \mathbb{R}^D \rightarrow \mathbb{R}^{D\times d}$, parameterised by a variable θ (e.g. parameters of an RBF-linear model). For that we first calculate the set of nearest neighbours N^i, for each point x^i of the training data. This way, if x^j is a neighbour of x^i, then there exists an unknown ϵ^{ij} such that $W(x^i, \epsilon^{ij}) = x^j$, or to a good approximation $\mathcal{H}_\theta(\bar{x}^{ij})\epsilon^{ij} \approx \Delta^i_{.j}$, where $\Delta^i_{.j}$ can be regarded as the centred estimate of the directional derivative at $\bar{x}^{ij}$.

To solve for $\mathcal{H}_\theta$ we define the error:

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

and minimise for θ with the addition of a regularisation term:

$$\lambda \sum \left\| \epsilon^{ij} \right\|_2^2 + \lambda \sum \left\| \mathcal{H}_\theta(\bar{x}^{ij}) - \mathcal{H}_\theta(\bar{x}^{ij'}) \right\|_F^2,$$

where $\bar{x}^{ij}$ and $\bar{x}^{ij'}$ are two neighboring locations, ϵ^{ij} and λ are regularisation terms that enforce the smoothness of the mapping. To solve this, a radial basis function(RBF)-based linear parametrisation is used, along with an alternating minimisation procedure (with random restarts to avoid local minima). Pseudocode for the method is available in Algorithm 1.

Projection. The projection of a point x on a learnt manifold M cannot be computed in closed form. Instead a gradient descent approach is utilised in finding a new point x' on M that minimises the distance $\|x - x'\|_2^2$. Since $\mathcal{H}_\theta$ is defined over the whole $\mathbb{R}^D$ we calculate the orthonormalised tangent space at x', $H' \equiv orth(\mathcal{H}_\theta(x'))$, and $H'H'^T$ the corresponding projection matrix. We follow the gradient to the local minima on the manifold, using the update rule for x': $x' \leftarrow x' + \alpha H'H'^T(x - x')$, with α being the step size. To find the closest projection we initially set x' to be the nearest point in the training data.

Geodesic distance. To compute the geodesic distance between two points, x and x', on a manifold we use an active contour model, also known as a *snake* [25]. A *snake* defines a discretised path between x and x' and its length is being minimised by gradient descent. The error reflecting the length of the path is given by: $err_{len}(\chi) = \sum_{i=2}^{m} \left\| \chi^i - \chi^{i-1} \right\|_2^2$, where the χ's are the linearly interpolated -manifold respecting- points between the fixed start and end points. The update rule for each χ^i is very similar to the update rule used for projection.

4 The Manifold-RRT Algorithm

Our algorithm is a variant of the conventional RRT, augmented with the manifold learning operation (Algorithm 1). This hybrid procedure is described in Algorithm 2. We use the learnt manifold, M, to compute distances between points

Algorithm 2. Manifold Path Planning

```
1: mRrt(q_init, q_goal, M)
2: INPUT: start point q_init, goal point q_goal, learnt manifold M
3: OUTPUT: path in configuration space p
4: T.add(q_init) {Initialize tree T}
5: for i = 0 to k do
6:    q_rand ← RANDOM_POINT
7:    q_proj ← PROJECT(q_rand, M)
8:    q_near ← GEODESIC_D(q_proj, T, M)
9:    q_new ← STEP(q_near, q_proj, dx) {Construct Snake}
10:   valid ← EVALUATE(q_new)
11:   if valid == true then
12:      T.add(q_new)
13:      dist ← GEODESIC_D(q_new, q_goal, M)
14:      if dist ≤ tolerance then
15:         break
16:      end if
17:   end if
18: end for
19: p ← SHORTEST_PATH(T.first,T.last)
20: RETURN p
```

in configuration space. The metric is the geodesic distance directly learnt from the training data. We utilise the geodesic distance to evaluate nearest neighbour relations and find q_{near}. This is used to decide which node of the tree will be subsequently grown. Moreover we use the learnt manifold to project uniform random samples in configuration space, q_{rand}, onto the manifold - focusing the planner to explore a task-relevant subspace.

Growing the tree T involves this projection, q_{proj}, of the random sample and the computation of a *snake* (Section 3.2) from the nearest neighbour on the graph to the new point, q_{new}. The interpolated points on the manifold that compose the *snake* are then examined and the geodesic distance, $dist$, from the starting point is computed. When the geodesic distance reaches the desired step size dx we set the via-point as the end of the step and evaluate the resulting path in simulation.

Next, the geodesic distance from the new vertex to the goal-point is computed. If the distance is lower than a tolerance threshold, the exploration stops. A shortest path from the start-point to the last vertex added is computed using a standard tree search algorithm. The resulting path, p, is the motion plan.

5 Experimental Setup

We present experiments with a simulated humanoid robot, KHR-2HV (Figure 2). This involves no explicit analytic model of the humanoid robot dynamics. Instead, we treat the simulated robot as an incrementally evaluable black-box.

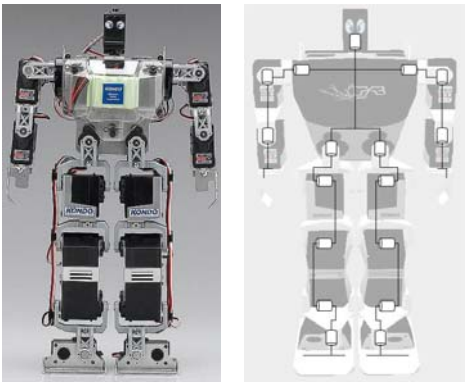

Fig. 2. The KHR-2HV Humanoid robot and the corresponding 17 degrees of freedom

So, although we present the results from simulation, the procedure can be identically applied to a physical robot as well. In particular, even though we search a region in configuration space, the intrinsic dynamics of the nonlinear high dimensional system are taken into consideration implicitly (of course, subject to the restriction of what is expressible in the configuration space).

Our simulation is in Webots [26], a commercial physically realistic modelling and simulation ODE-based environment. In Webots we use an accurate model of the Kondo KHR-2HV humanoid robot, where motion is performed using P-controllers that closely simulate the characteristics of the real robot servos. A controller has been implemented in C that handles the communication between Webots and Matlab and exposes the full functionality of the robot model. Both cRRT and mRRT algorithms are implemented in Matlab and communicate directly with the simulator for the evaluation of configurations. LSML is implemented in Matlab, using Piotr Dollar's LSML code [2].

5.1 Task

We have experimented with a number of bipedal tasks. However, we include nothing in our experiment that is specific to these particular tasks, thus the same procedure is applicable to other bipedal tasks as well. We compare the classical RRT and mRRT on the same tasks of forward and backward stepping and kicking. For the purposes of planning, we consider all the leg and hip DoFs of the humanoid, resulting in a 10-dimensional configuration space.

We begin with a single example – a hand-crafted trajectory, from stance to double support for stepping and to midair reach for kicking. We sample the training data in simulation from the KHR-2HV humanoid and we use a 5 millisecond sampling rate that is equal to the physical simulation time step. The resulting motions are 116, 98 and 96 points long for step, kick and backstep

[2] Available at: http://vision.ucsd.edu/∼pdollar/

accordingly. The start and end points are chosen equally as the initial and goal points of both compared algorithms. In order to add variability to the learning step of mRRT we further use 2 perturbed instances of the previous motions. These trajectories are generated by random normal sampling in the vicinity of the training data and subsequent stability evaluation in simulation. Furthermore the sequential nature of the training data ensures that the learnt sub–space is a single manifold and that is not disconnected.

5.2 Evaluation of Samples

In both cases, new samples are evaluated in simulation. It is worth noting that we do not have an explicit model of the robot's kinematics or dynamics. So, samples are evaluated in a dynamic fashion that ensures their suitability. Both in cRRT and mRRT, the nearest neighbour of every new sample is computed on each iteration of the exploration cycle. The humanoid's servos are set at the appropriate positions and its global position and rotation is set accordingly. The robot is then commanded to perform the motion that reaches the new configuration point according to the servos' P-controllers. We utilise feedback from the gyroscope and the accelerometers to evaluate the stability and stance. Furthermore we employ two foot force sensors to distinguish between single and double support configurations.

5.3 Other Parameters

We have used the average geodesic distance between data points in the training set to set the step length in both algorithms. Such a choice is well suited to the task at hand and was made for comparability in evaluation. Note that this choice greatly *favours* cRRT as the metric used is now '*informed*' in a systematic manner, in contrast to the often ad-hoc RRT setting. We have set a bias of 0.1 towards the goal point in order to boost convergence. The default LSML parameterisation has been used with no effort at special optimisation as errors have been adequately small. The actual time for learning a manifold depends on the amount of training data and for our experiments required less than a minute in all cases.

6 Results

We compare the performance of cRRT and mRRT with a number of different metrics. Each trial has been repeated 10 times for both algorithms and all reported results are averaged over the number of trials. Examples of a resulting paths discovered by mRRT are depicted in Figure 3.

The evaluation metrics are quite intuitive. In particular, we note the following. Average path length corresponds to the number of points that are traversed from the initial configuration in order to reach the goal configuration. Number of samples denote the total explorative samples needed until the goal is reached.

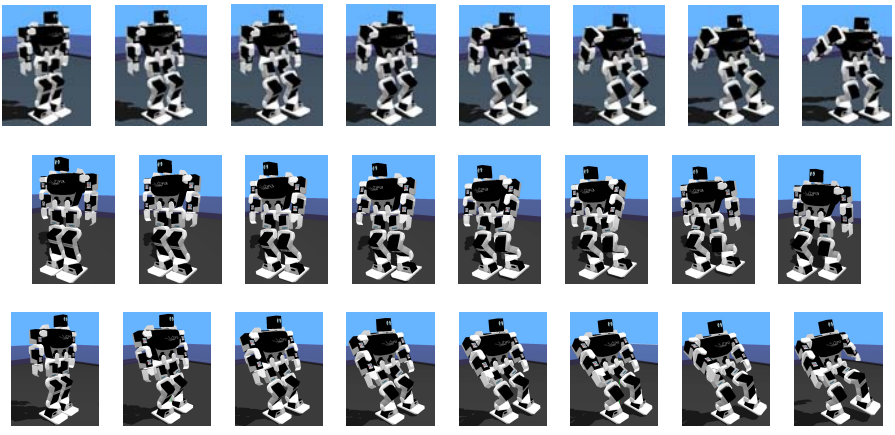

Fig. 3. Paths discovered by mRRT for left step forward *(top row)*, left step backward *(middle row)* and left leg kick *(bottom row)*

Table 1. Results averaged over 10 trials for step forward, kick and step backward

Task	*step*		*kick*		*backstep*	
	cRRT	***mRRT***	***cRRT***	***mRRT***	***cRRT***	***mRRT***
Average path length	40.9	38	52.5	49.4	47.2	37.5
Average number of samples	268.63	199.2	291	249.3	293.4	189.8
Average tree size	127.7	127	140.3	137.7	120.7	108.4
Average number of invalid samples	140.6	74.4	150.7	111.6	172.7	81.4
Smoothness {*nRMSE*}	0.0051	0.0049	0.0055	0.0041	0.0046	0.0043

Tree size is the number of vertices that the resulting tree consists of. Invalid samples are evaluated points that do not satisfy dynamic stability or collision constraints. Smoothness is defined as the average normalised Root Mean Square Error (nRMSE) with respect to a fitted cubic polynomial at each joint motion for each resulting plan path.

Our experimental results show that mRRT, in all trials, discovers a solution with *much fewer* invalid samples. These are the random configurations that fail in the evaluation step. On average mRRT explores only half as many 'bad' samples as cRRT (57.6%). This translates to an average decrease of 25.2% in overall planning steps for the specific tasks. More importantly, as tasks become more complex in terms of dynamical constraints, this ensures that exploration is accordingly useful.

On average, mRRT discovers shorter paths than cRRT and requires much fewer samples. For all tasks, the average size of trees is approximately equal and both algorithms find smooth paths. The results are summarised in Table 1. In general, we expect that the above mentioned differences would be more pronounced as the tasks are more spatio-temporally extended and dynamical realisability constraints become more severe.

7 Conclusion

We have demonstrated an approach to sampling based motion planning that utilizes demonstrated examples to glean information regarding task-specific constraints. This data could come from motion capture or perhaps even just a few hand crafted partial solutions. Our approach is to augment a sampling-based motion planning algorithm with a manifold learning procedure to provide task-specific metrics and a way to synthesize motion as geodesics. We have shown that this yields a marked improvement in exploration efficiency with respect to a standard RRT based planner, due to the fact that the learnt metric focusses exploration better than other forms of random sampling in a larger space. In addition, and very importantly, we note that this procedure yields an efficient encoding of the many different ways to perform a particular task (e.g., quantitatively different kicking trajectories), which is crucial for the construction of multi-level motion synthesis strategies.

We have shown examples of bipedal tasks in simulation and our current work involves porting the algorithm to a physical robot platform. Also, in future work, we would like to augment the space with velocity and acceleration information which will be required to encode many challenging dynamic behaviours including jumping and running. Finally, our long term goal is to use this procedure as a way to seed the learning of a layered architecture for control, planning and reasoning.

References

1. LaValle, S.M.: Planning Algorithms. Cambridge University Press, Cambridge (2006)
2. Full, R., Koditschek, D.: Templates and anchors: neuromechanical hypotheses of legged locomotion on land. J. Exp. Biol. 202(23), 3325–3332 (1999)
3. Gutfreund, Y., Flash, T., Yarom, Y., Fiorito, G., Segev, I., Hochner, B.: Organization of Octopus Arm Movements: A Model System for Studying the Control of Flexible Arms. J. Neurosci. 16(22), 7297–7307 (1996)
4. Ramamoorthy, S., Kuipers, B.J.: Qualitative hybrid control of dynamic bipedal walking. In: Proceedings of Robotics: Science and Systems, Philadelphia, USA (August 2006)
5. Ramamoorthy, S., Kuipers, B.J.: Trajectory generation for dynamic bipedal walking through qualitative model based manifold learning. In: IEEE International Conference on Robotics and Automation (ICRA), May 2008, pp. 359–366 (2008)
6. Isto, P., Saha, M.: A slicing connection strategy for constructing prms in high-dimensional cspaces. In: Proceedings 2006 IEEE International Conference on Robotics and Automation, ICRA 2006, May 2006, pp. 1249–1254 (2006)
7. Safonova, A., Hodgins, J.K., Pollard, N.S.: Synthesizing physically realistic human motion in low-dimensional, behavior-specific spaces. ACM Trans. Graph. 23(3), 514–521 (2004)
8. Stilman, M.: Task constrained motion planning in robot joint space. In: IEEE/RSJ International Conference on Intelligent Robots and Systems, IROS 2007, 29 November 2 (2007)

9. Yao, Z., Gupta, K.: Path planning with general end-effector constraints: using task space to guide configuration space search. In: 2005 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2005), August 2005, pp. 1875–1880 (2005)
10. Bretl, T., Lall, S., Latombe, J.C., Rock, S.: Multi-step motion planning for free-climbing robots. In: WAFR, pp. 1–16 (2004)
11. James, J., Kuffner, J., Kagami, S., Nishiwaki, K., Inaba, M., Inoue, H.: Dynamically-stable motion planning for humanoid robots. Auton. Robots 12(1), 105–118 (2002)
12. Nakamura, Y., Yamane, K.: Interactive motion generation of humanoid robots via dynamics filter. In: Proc. of First IEEE-RAS Int. Conf. on Humanoid Robots (2000)
13. Beaudoin, P., van de Panne, M., Poulin, P., Coros, S.: Motion-motif graphs. In: Symposium on Computer Animation 2008 (July 2008)
14. LaValle, S.M., Kuffner, J.J.: Randomized kinodynamic planning. The International Journal of Robotics Research 20(5), 378–400 (2001)
15. Vijayakumar, S., D'souza, A., Schaal, S.: Incremental online learning in high dimensions. Neural Comput. 17(12), 2602–2634 (2005)
16. Bitzer, S., Havoutis, I., Vijayakumar, S.: Synthesising novel movements through latent space modulation of scalable control policies. In: Asada, M., Hallam, J.C.T., Meyer, J.-A., Tani, J. (eds.) SAB 2008. LNCS (LNAI), vol. 5040, pp. 199–209. Springer, Heidelberg (2008)
17. Jenkins, O.C., Mataric, M.J.: A spatio-temporal extension to isomap nonlinear dimension reduction. In: International Conference on Machine Learning (ICML), pp. 441–448 (2004)
18. Wang, J.M., Fleet, D.J., Hertzmann, A.: Gaussian process dynamical models for human motion. IEEE Trans. Pattern Anal. Mach. Intell. 30(2), 283–298 (2008)
19. Urtasun, R., Fleet, D.J., Geiger, A., Popovic, J., Darrell, T.J., Lawrence, N.D.: Topologically-constrained latent variable models. In: Proceedings of the 25th international Conference on Machine Learning, Helsinki, Finland, July 5-9, pp. 1080–1087. ACM, New York (2008)
20. Hastie, T., Tibshirani, R., Friedman, J.H.: The Elements of Statistical Learning. Springer, Heidelberg (2001)
21. Roweis, S.T., Saul, L.K.: Nonlinear dimensionality reduction by locally linear embedding. Science 290(5500), 2323–2326 (2000)
22. Tenenbaum, J.B., de Silva, V., Langford, J.C.: A global geometric framework for nonlinear dimensionality reduction. Science 290(5500), 2319–2323 (2000)
23. Dollár, P., Rabaud, V., Belongie, S.: Non-isometric manifold learning: Analysis and an algorithm. In: ICML (June 2007)
24. Dollár, P., Rabaud, V., Belongie, S.: Learning to traverse image manifolds. In: NIPS (December 2006)
25. Blake, A., Isard, M.: Active Contours. Springer, Heidelberg (1998)
26. Michel, O.: Webots: Professional mobile robot simulation. Journal of Advanced Robotics Systems 1(1), 39–42 (2004)