Improving Local Trajectory Optimisation using Probabilistic Movement Primitives

RB Ashith Shyam[1], Peter Lightbody[1], Gautham Das[1],
Pengcheng Liu[2], Sebastian Gomez-Gonzalez[3,4] and Gerhard Neumann[1]

***Abstract*— Local trajectory optimisation techniques are a powerful tool for motion planning. However, they often get stuck in local optima depending on the quality of the initial solution and consequently, often do not find a valid (i.e. collision free) trajectory. Moreover, they often require fine tuning of a cost function to obtain the desired motions. In this paper, we address both problems by combining local trajectory optimisation with learning from demonstrations. The human expert demonstrates how to reach different target end-effector locations in different ways. From these demonstrations, we estimate a trajectory distribution, represented by a Probabilistic Movement Primitive (ProMP). For a new target location, we sample different trajectories from the ProMP and use these trajectories as initial solutions for the local optimisation. As the ProMP generates versatile initial solutions for the optimisation, the chance of finding poor local minima is significantly reduced. Moreover, the learned trajectory distribution is used to specify the smoothness costs for the optimisation, resulting in solutions of similar shape than the demonstrations. We demonstrate the effectiveness of our approach in several complex obstacle avoidance scenarios.**

Keywords: motion planning, obstacle avoidance, gradient optimisation, probabilistic movement primitives, robot manipulation, learning from demonstrations.

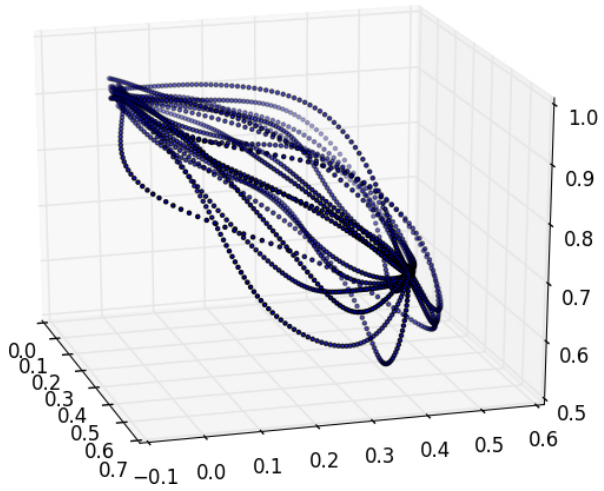


Fig. 1. End-effector trajectories in task space after conditioning from the learned distribution

I. INTRODUCTION

Trajectory planning is a classic problem within robotics research which addresses the issue of traversing a robot through a potentially complex obstacle-cluttered environment from a start state to a goal state.

For humans, trajectory planning is relatively straight forward. One of the main goals of this work is to replicate that natural or human-like motion. This type of motion can be characterised as one which not only minimises energy consumption but also avoids motor or actuator damage by producing a smooth and jerk-free trajectory. Another aspect of this work is to provide a more general solution to local planning problems as compared to the work [1] as well as providing a solution which less frequently gets stuck at local minima. Utilising learning from kinesthetic demonstrations provides the opportunity to understand human motion, while combining this with optimisation-based obstacle avoidance results in trajectories which can be both reliably generated and natural looking.

In this work we use Probabilistic Movement Primitives (ProMPs) to learn the trajectory distribution from which any number of trajectories could be sampled. The reason for choosing a probabilistic approach is due to its properties like time modulation, encoding the variability, conditioning to desired state etc. For motion planning the conditioned trajectories can be optimised by minimising the weighted sum of the smoothness and obstacle costs which produces a collision-free trajectory. The smoothness cost incorporates the dynamics of the trajectories by penalising random jerky motions whereas the force required to push the robot away from obstacles is encapsulated within the obstacle cost. We compare the performance of our new planner, based on ProMPs, against other local planners like CHOMP and STOMP in respect of planning time, ability to successfully plan a trajectory and total cost of the final trajectory. To illustrate that local planners produce a smoother trajectory, the smoothness cost of the ProMP planner is compared with a sampling based planner, RRT.

To avoid getting stuck at local minima, initialisation of the ProMP planner can be achieved with several feasible trajectories sampled from the conditioned distribution. Unlike CHOMP, where optimisation is carried out in trajectory space, our study optimises the cost function in parameter space. Thus the number of parameters to optimise is considerably reduced and the computational time consequently much less than is required for CHOMP.

[1] Lincoln Centre for Autonomous System, University of Lincoln, UK
[2] Cardiff School of Technologies, Cardiff Metropolitan University, UK
[3] Max Plank Institute for Intelligent Systems, Täbingen, Germany
[4] Computer Science Department, TU Darmstadt, Darmstadt, Germany

Manuscript 1128 submitted to 2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS). Received March 1, 2019.

II. RELATED WORK

The evolution of movement primitives can be traced back to the late 1990s where the works [2], [3] demonstrated that several human kinematic movements can be expressed as a linear combination of a small number of components. This is known as Principal Component Analysis and states that 80% of the variance in a human grasp can be captured using only two principal components [4]. Following this, the pioneering work by Schaal called the Dynamic Movement Primitives (DMPs) [5] formalised the theory for trajectory planning and control for dynamic non-linear systems without causing instability. However, DMPs also have many weaknesses including their inability to adapt to unforeseen situations and changing intermediate points. In addition to these shortcomings, works by [6] have also demonstrated that DMPs often produce sub-optimal solutions.

To encode variability in movements, a probabilistic framework for motion planning was proposed by Calinon et. al in [7]. Paraschos et. al. then extended this concept to be what is known as Probabilistic Movement Primitives [6]. Compared to DMPs, ProMPs is a more general framework which encodes the distribution of possible trajectories which can be conditioned to the desired start and goal state to achieve a motion plan.

Motion planning can generally be categorised into three sets: sampling-based [8], optimisation based [9] and learning from demonstrations [1], each of which have advantages and drawbacks. Sampling-based planners like Rapidly-exploring Random Trees (RRT) [10] and Probabilistic Road Maps (PRM) [11], along with its variants, are particularly well suited for mobile robots and have been used successfully in computational biology [12], computer graphics [13] and animations [14]. However, sampling-based planners can often suffer from a high computational cost as discussed in detail in [15], [16] & [17]. In addition, the random nature of these algorithms can lead to inconsistent behaviour with a large amount of variance even when all other constraints are the same. This level of variation is prone to producing trajectories which a human would likely deem as unnatural like motion. This can be further emphasised with the introduction of obstacles. In order to avoid obstacles within the scene, sampling-based methods utilise a validation condition which rejects sample positions which would collide with an object, resulting in trajectories with a potentially low smoothness cost.

Optimisation planners, on the other hand, were originally developed by Quinlan and Khatib [18] and requires an objective function to be minimised which forces the manipulator to go from a start position to a goal state and can be used to push the robot away from obstacles. Such algorithms are often capable of handling constraints imposed by the platform such as the robots joint limits. However, one of the main concerns with optimisation based algorithms is that they depend entirely on the minimising function which can often become stuck at local minima. In other words, formulating a convex functional for optimisation is the key to guaranteeing a solution which in certain task-specific cases, like a robot disentangling a rope, is extremely difficult [1]. More traditional approaches to robot motion planning are also typically derived, by an expert, from mathematically-based policies. This approach not only requires a specialist to develop it, but is also heavily impacted by inaccuracies of the world model [19]. This is where learning from demonstrations (LfD) can be useful. Here, human demonstrations of a desired motion (which is capable of being taught to a robot) are provided from which a policy can be produced in order to replicate the behaviour similar to the actions which were taught. As a result, complex behaviours can be adopted without having to be specifically encoded. An example of using learning from demonstration to produce motion plans can be found in work by Lauretti et. al. [20] but it is important to note, however, that where demonstrations are carried out when there are no obstacles, post-processing of the trajectories is essential to make the resulting motion plan obstacle free.

As far as the optimisation is concerned, the result should be an obstacle-free, smooth trajectory. There are two significant costs involved in optimisation namely the smoothness cost of the trajectory and the cost of being near an obstacle. Various researchers have incorporated smoothness as the sum of the squared velocity or acceleration or jerk [9], [15] in trajectory space and hence the number of variables to optimise is large. To avoid colliding with obstacles, a force is needed to push the robot away from obstacles. This was modelled as elastic bands in the work [18].

III. THE PROMP ALGORITHM

A. Learning from demonstrations

In this work, we consider τ to be the trajectory which maps time to robot configurations (y). This can be written as $\tau : [0, 1] \longrightarrow y$ where $y \subset \mathbb{R}^d$ where 'd' corresponds to the number of joints. The linear basis function (ϕ) model representation of the robot configuration, y_t at any time instant, t, and the probability of observing a trajectory given the weight vector w can be written as in [6] as

$$y_t = \begin{bmatrix} q_t \\ \dot{q}_t \end{bmatrix} = \begin{bmatrix} \phi_t \\ \dot{\phi}_t \end{bmatrix} w + \epsilon_y \tag{1}$$

$$p(\tau|w) = \prod_t \mathcal{N}(y_t|\Phi_t w, \Sigma_y) \tag{2}$$

where $\epsilon_y \sim \mathcal{N}(0,\ \Sigma_y)$ represents the Gaussian noise associated with each observation. The choice of basis function depends on the type of task to be carried out. For example, Gaussian or Von-Mises basis functions can be used respectively to learn stroke-based or rhythmic movements. The mean and variance of the weight vector w can be captured by introducing another parameter θ and the trajectory distribution can finally be obtained as

$$p(\tau; \theta) = \int p(\tau|w)p(w; \theta)dw \tag{3}$$

Manuscript 1128 submitted to 2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS). Received March 1, 2019.

For estimating the weights, a ridge regression could be carried out as given in eq. (4)

$$w_i = (\Phi^T \Phi + \lambda I)^{-1}\ \Phi^T Y_i \tag{4}$$

where Y_i is the concatenation of all joint positions from the demonstrated data. The mean and variance of the parameter vector, w, are estimated as in eq. (5)

$$\begin{aligned} \mu_w &= \frac{1}{N}\sum_{i=1}^{N} w_i \\ \Sigma_w &= \frac{1}{N}\sum_{i=1}^{N}(w_i - \mu_w)(w_i - \mu_w)^T \end{aligned} \tag{5}$$

For a motion planning problem, we need to take the robot from a start configuration to a goal configuration. This implies that $\tau(0)$ and $\tau(1)$ are start and goal configuration respectively and are known in advance either in task space or configuration space.

B. Conditioning

One of the most important properties of probabilistic distributions which is especially useful for motion planning is conditioning. A probabilistic trajectory distribution can be conditioned to follow not only the desired start and goal state but also follow through the via-points [6]. For example if our trajectory has to pass through a desired state y_t^* at any time point t, it is incorporated by adding a desired observation to the probabilistic model and using Bayes theorem.

$$x_t^* = \{y_t^*, \Sigma_y*\} \tag{6}$$

It can be shown that for Gaussian trajectory distributions, the new mean and variance of the conditioned trajectory can be written as

$$\begin{aligned} \mu_w^{[new]} &= \mu_w + L(y_t^* - \Phi_t^T \mu_w) \\ \Sigma_w^{[new]} &= \Sigma_w - L\Phi_t^T \Sigma_w \end{aligned} \tag{7}$$

where L is

$$L = \Sigma_w \Phi_t (\Sigma_y^* + \Phi_t^T \Sigma_w \Phi_t)^{-1} \tag{8}$$

A typical plot of the end-effector trajectory in task space after conditioning the learned distribution can be seen as shown in Fig 1.

C. Trajectory optimisation

For the optimisation part, without loss of generality, it is assumed that the start and goal configurations are collision-free. We assume at this point the trajectories are already conditioned and the only objective is obstacle avoidance by perturbing the conditioned distributions by a small amount without affecting the conditioned states.

1) Smoothness cost: The time evolution of the trajectory can be written as

$$\tau = \Phi w + \epsilon_y \tag{9}$$

where Φ is the concatenation of the basis function matrix ϕ. The smoothness cost of the trajectory can be described in the parameter space as

$$S = \frac{1}{2}(w - \mu_w)^T \Sigma_w^{-1}(w - \mu_w) + \frac{1}{2}\alpha\ w^T w \tag{10}$$

where μ_w and Σ_w are the mean and covariance of the learned trajectory distribution and α is a regularisation coefficient. The term $\frac{1}{2}(w-\mu_w)^T\Sigma_w^{-1}(w-\mu_w)$ penalises those trajectories which go far away from the mean μ_w whereas the term $\frac{1}{2}\alpha\ w^T w$ prevents the parameter w from having very high values. The gradient of the smoothness cost with respect to the parameter w is

$$\nabla S = \Sigma_w^{-1}(w - \mu_w) + \alpha * w \tag{11}$$

2) Obstacle cost: As discussed in CHOMP [15], for the formulation of obstacle cost, we consider that each link of the manipulator is made up of several geometric primitive shapes whose dimensions and spacing are good enough to specify the physical dimensions of the robot. In a similar manner, we can consider the obstacles in the planning environment to be made up of several small geometric primitive shapes. For simplicity, we consider the robot body and the obstacles are made of several spheres. From the Denavit-Hartenberg [21] parameters, it is possible to find the forward kinematics and jacobian of each of the robot body points given the joint values, $y_t \subset \mathbb{R}^d$. As mentioned earlier, a trajectory is a time series of joint values and for one particular trajectory it is possible to calculate the signed minimum distance between the robot body spheres and the obstacle spheres. The value of the signed distance is an indication as to whether the robot is away from the obstacle (positive), touching the obstacle (zero) or hitting the obstacle (negative). There are various ways we could construct such a signed distance field and for the purpose of our study we used the following

$$c = \begin{cases} 0 & \text{if } d > \epsilon \\ k(d-\epsilon)^2 & \text{if } d \leq \epsilon \end{cases} \tag{12}$$

where d, ϵ and k are respectively the Euclidean distance between the sphere surfaces, the threshold value up to which the robot can reach near the obstacle and a gain parameter. In order to make sure that the obstacle cost is invariant to the choice of parameter value, we multiply the cost, c, with the velocity of the body sphere and the obstacle cost functional becomes

$$O[w] = \int_0^1 \int_u c(x_u(w,t)) \left\| \frac{d}{dt} x_u(w,t) \right\| du\ dt \tag{13}$$

where u and x_u represent the body spheres and position of the body sphere in task space given by the parameter w and time respectively. The gradient of the obstacle functional with respect to the parameter w is

$$\nabla O[w] = \Phi^T \int_u J^T \left[\|\dot{x}_u\| \left((I - \hat{\dot{x}}_u \hat{\dot{x}}_u^T) \nabla c - c\kappa \right) \right] \quad (14)$$

where $\dot{x}_u$, $\hat{\dot{x}}_u$, J and κ are respectively the velocity of the robot body point u, the unit velocity vector, the jacobian of each robot body point and the curvature of the trajectory defined by

$$\kappa = \frac{1}{\|\dot{x}_u\|}^2 \left((I - \hat{\dot{x}}_u \hat{\dot{x}}_u^T) \ddot{x}_u \right) \quad (15)$$

where $\ddot{x}_u$ is the acceleration.

3) Total objective cost and gradient: From eq. (10) and eq. (13), the total cost function can be written as

$$TC = \lambda_{smooth} \times S + \lambda_{obs} \times O \quad (16)$$

and the gradient becomes

$$TG = \lambda_{smooth} \times \nabla S + \lambda_{obs} \times \nabla O \quad (17)$$

where λ_{smooth} and λ_{obs} are scalar weight parameters. It is to be noted that the λ's need tuning.

The algorithm can be summarised as given below

- **Given:**
  - Start, $\tau(0)$ and goal, $\tau(1)$ configurations either in joint or task space
  - Kinesthetic demonstration data (joint parameters)
  - DH table of the robot
  - A weight dependent cost function, $C(w)$
- **Precompute:**
  - Compute the basis function, Φ
  - Compute the mean, μ_w and covariance, Σ_w of the parameter vector
  - Compute IK if $\tau(0)$ or $\tau(1)$ is given in task space.
  - Compute μ_w^{new} and Σ_w^{new}
- **Repeat Until Convergence:**
  - Sample weight vector from the learned distribution
  - Calculate the total cost and gradient
  - Repeat for several iterations if the sampled weight vector takes the cost to the tolerance:

    1) if cost greater than tolerance,

    $$w_1 = w - \alpha \times \frac{C(w)}{C'(w)}$$

    2) Calculate trajectory cost, $C(w)$

IV. EXPERIMENT

A. Methodology

For our experiments, we used Franka Emika's 7-DOF panda robot [22]. For our study, the robot was tasked with moving under the table from an upright starting position, shown in Fig 2, to a goal state beneath the table identified by a fiducial marker [23]. This involved the robot moving through the gaps between the table legs in various ways. Hundreds of kinesthetic demonstrations by a human expert were carried out and the joint parameters recorded. This data was used to find the mean and co-variance of the parameter vector w. In most real-world applications, the task space end-effector goal state is specified but our learning is carried out in joint space. The transformation from task space to joint space can easily be carried out by another optimisation-based inverse kinematic approach. Since the panda robot has a redundant degree of freedom we were able to get several inverse kinematic solutions (IKs) but we penalise those IKs which deviate too greatly from the mean of the demonstrated data. The learning can be carried out as described in section III-A. A plot of the learned trajectories by maximum-a-posteriori sampling is shown in Fig 3. The trajectories are then conditioned to the required start and goal state as per eq. (7). This new mean and variance of the parameter vector can be used to sample several trajectories with all the trajectories having the same end-effector pose. A plot of the mean and variance of the conditioned trajectories for joint 1 is shown in Fig 4.

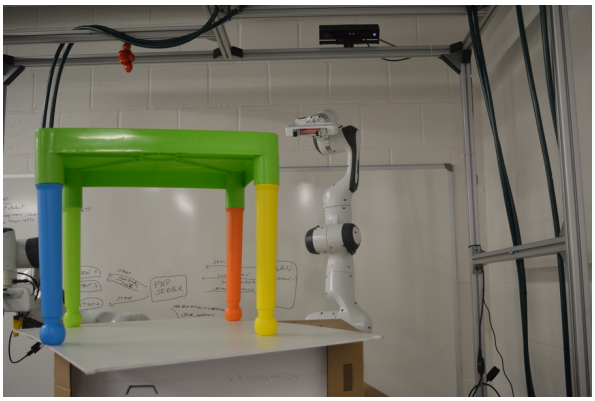


Fig. 2. Planning Scene

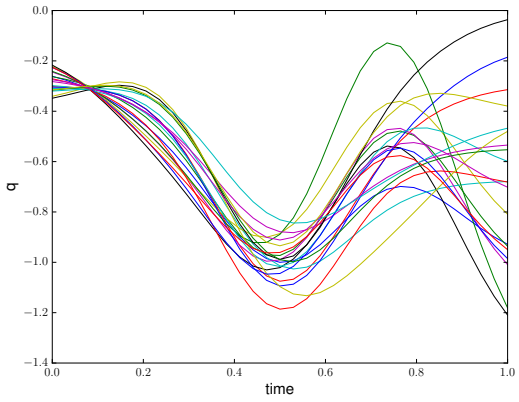


Fig. 3. MAP sampling of learned trajectory for joint 1

For the optimisation, the solution generally depends on the initial estimate provided to the algorithm. The trajectories obtained after conditioning are all possible candidates as

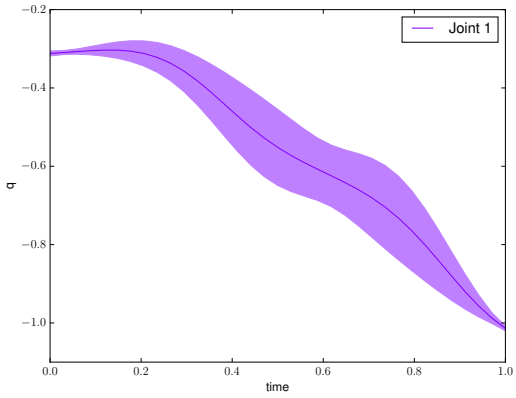
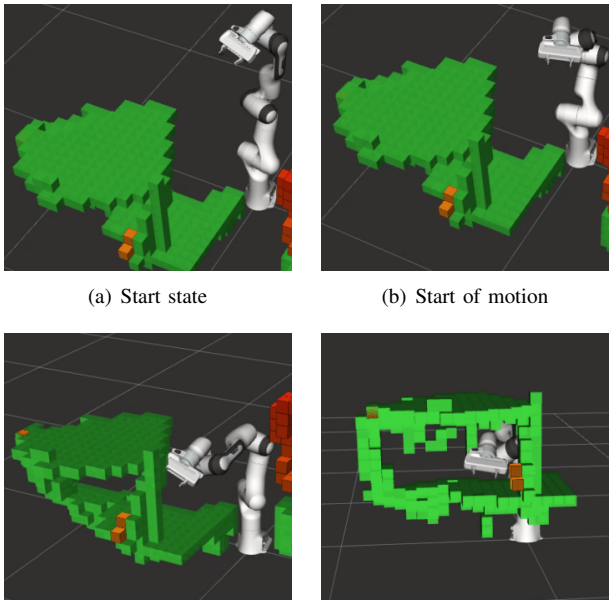


Fig. 4. Mean and variance of the conditioned trajectory for joint 1

an initial estimate to the optimisation, which eliminates the possibility of getting stuck at the local minima. In general, it is safe to assume that the demonstrations are more or less smooth since they were carried out by a human expert and hence the weights to multiply the smoothness cost are set to small, whereas the cost of being near an obstacle is penalised much higher.



(a) Start state (b) Start of motion

(c) Going under the table (d) Goal state

Fig. 5. Evolution of the trajectory planning

We used a Microsoft Kinect camera and integrated it with Robot Operating System (ROS) [24] for our vision system and is calibrated to the robot base. The obstacles are represented as octomaps [25] in the planning scene as shown in Fig 5. It also shows the evolution of the motion plan to reach a pose under the table. The comparison of our algorithm against two existing local planners, viz., CHOMP [15] and STOMP [9] is carried out and is shown in Table I. For CHOMP, a linearly discretised straight line trajectory from the start to the goal state is provided as the initial guess. For STOMP, as described in the work, the initialisation is carried out by perturbing the straight line trajectory by $\epsilon_k = \mathcal{N}(0, R^{-1})$ where R can be derived from the finite difference matrix.

V. EVALUATION & RESULTS

A. Experiments in Simulation

For our simulation study, a real scene with a table as an obstacle was recorded with Kinect as point-clouds and was imported to RViz, as shown in Fig 5. The position the robot end-effector has to reach is carefully chosen to be under the table (and the table can be moved in the work space as long as there are enough demonstrations to learn effectively). ProMP, CHOMP and STOMP are implemented in Python whereas RRT is used directly from the Open Motion Planning Library (OMPL) [26]. The purpose of this is to show that the plan generated by local planners is smoother than sampling-based planners. Quantitatively, the ProMP generated a plan which on average is 52% and 79% smoother than CHOMP/STOMP and RRT respectively. The success rate for planning was found to be 87%, 21% and 71% for ProMP, CHOMP and STOMP respectively. Average planning time to success for local planners was relatively high. This could be attributed to implementation in Python and can be significantly reduced by opting for other programming languages like C/C++. However, from the results obtained, it was clearly demonstrated that for ProMP the average time for successful plan is 17% and 10% better than CHOMP and STOMP respectively.

B. Experiments With a Real Robot

In addition to the tests in simulation, our algorithm has also been measured using a real 7 DOF Frank Emika's Panda robot. The video attached shows how the planner goes to a location under the table in two different ways

VI. CONCLUSION

From the results presented here, it is clear that local trajectory optimisation by learning from demonstrations results in a much smoother trajectory. Additionally, when compared to other optimisation-based planners, the likelihood of getting stuck at local minima is considerably reduced, as seen from the success rate given in Table I. There is also the potential for more complicated motion plans to be achieved with the help of expert human demonstrators.

Potential future work could include the creation of a more robust planner, achieved by modelling using Gaussian mixtures. It would be worthwhile to both investigate those scenarios where the planner fails to find a solution and to incorporate parameters for sequential learning.

ACKNOWLEDGEMENT

The work was funded by Innovate UK grant Automato Robot Tomato Harvester, IUK 102642, and supported by Xihelm, Ltd.

TABLE I

RESULTS OF MOTION PLANNING IN SIMULATION

Scenario	ProMP	Chomp (Straight line initialisation)	STOMP (stochastic initialisation)	RRT
Number of successful plans	87/100	21/100	71/100	92/100
Average planning time to success (s)	58.15 $\pm$ 11.56	68.19 $\pm$ 13.42	63.99 $\pm$ 12.65	0.15 $\pm$ 0.06
Average smoothness cost	0.079 $\pm$ 0.0082	0.167 $\pm$ 0.03	0.161 $\pm$ 0.03	0.377 $\pm$ 0.06
Average obstacle cost	0.038 $\pm$ 0.034	0.0165 $\pm$ 0.014	0.022 $\pm$ 0.015	0.063 $\pm$ 0.022

REFERENCES

[1] T. Osa, A. M. G. Esfahani, R. Stolkin, R. Lioutikov, J. Peters, and G. Neumann, "Guiding trajectory optimization by demonstrated distributions," *IEEE Robotics and Automation Letters*, vol. 2, no. 2, pp. 819–826, 2017.

[2] J. F. Soechting, Marco Santello, "Matching object size by controlling finger span and hand shape," *Somatosensory & motor research*, vol. 14, no. 3, pp. 203–212, 1997.

[3] M. Santello, M. Flanders, and J. F. Soechting, "Postural hand synergies for tool use," *Journal of Neuroscience*, vol. 18, no. 23, pp. 10 105–10 115, 1998.

[4] T. D. SANGER, "Human arm movements described by a low-dimensional superposition of principal components," *J. Neurosci.*, vol. 20, pp. 1066–1072, 2000.

[5] S. Schaal, "Dynamic movement primitives-a framework for motor control in humans and humanoid robotics," in *Adaptive motion of animals and machines*. Springer, 2006, pp. 261–280.

[6] A. Paraschos, C. Daniel, J. Peters, and G. Neumann, "Using probabilistic movement primitives in robotics," *Autonomous Robots*, vol. 42, no. 3, pp. 529–551, 2018.

[7] S. Calinon, F. D'halluin, E. L. Sauser, D. G. Caldwell, and A. G. Billard, "Learning and reproduction of gestures by imitation," *IEEE Robotics & Automation Magazine*, vol. 17, no. 2, pp. 44–54, 2010.

[8] S. Karaman and E. Frazzoli, "Sampling-based algorithms for optimal motion planning," *The international journal of robotics research*, vol. 30, no. 7, pp. 846–894, 2011.

[9] M. Kalakrishnan, S. Chitta, E. Theodorou, P. Pastor, and S. Schaal, "Stomp: Stochastic trajectory optimization for motion planning," in *2011 IEEE international conference on robotics and automation*. IEEE, 2011, pp. 4569–4574.

[10] S. M. Lavalle, J. J. Kuffner, and Jr., "Rapidly-exploring random trees: Progress and prospects," in *Algorithmic and Computational Robotics: New Directions*, 2000, pp. 293–308.

[11] L. E. Kavraki, M. N. Kolountzakis, and J. Latombe, "Analysis of probabilistic roadmaps for path planning," *IEEE Trans. Robotics and Automation*, vol. 14, no. 1, pp. 166–171, 1998. [Online]. Available: https://doi.org/10.1109/70.660866

[12] Y. Koga, K. Kondo, J. Kuffner, and J.-C. Latombe, "Planning motions with intentions," in *Proceedings of the 21st annual conference on Computer graphics and interactive techniques*. ACM, 1994, pp. 395–408.

[13] J. Pettré, J.-P. Laumond, and T. Siméon, "A 2-stages locomotion planner for digital actors." in *Symposium on Computer Animation*, 2003, pp. 258–264.

[14] N. M. Amato and G. Song, "Using motion planning to study protein folding pathways," *Journal of Computational Biology*, vol. 9, no. 2, pp. 149–168, 2002.

[15] M. Zucker, N. Ratliff, A. D. Dragan, M. Pivtoraiko, M. Klingensmith, C. M. Dellin, J. A. Bagnell, and S. S. Srinivasa, "Chomp: Covariant hamiltonian optimization for motion planning," *The International Journal of Robotics Research*, vol. 32, no. 9-10, pp. 1164–1193, 2013.

[16] J. Schulman, Y. Duan, J. Ho, A. Lee, I. Awwal, H. Bradlow, J. Pan, S. Patil, K. Goldberg, and P. Abbeel, "Motion planning with sequential convex optimization and convex collision checking," *The International Journal of Robotics Research*, vol. 33, no. 9, pp. 1251–1270, 2014.

[17] S. R. Lindemann and S. M. LaValle, "Current issues in sampling-based motion planning," in *Robotics Research. The Eleventh International Symposium*. Springer, 2005, pp. 36–54.

[18] S. Quinlan and O. Khatib, "Elastic bands: Connecting path planning and control," in *[1993] Proceedings IEEE International Conference on Robotics and Automation*. IEEE, 1993, pp. 802–807.

[19] B. D. Argall, S. Chernova, M. Veloso, and B. Browning, "A survey of robot learning from demonstration," *Robotics and autonomous systems*, vol. 57, no. 5, pp. 469–483, 2009.

[20] C. Lauretti, F. Cordella, A. L. Ciancio, E. Trigili, J. M. Catalan, F. J. Badesa, S. Crea, S. M. Pagliara, S. Sterzi, N. Vitiello, *et al.*, "Learning by demonstration for motion planning of upper-limb exoskeletons," *Frontiers in neurorobotics*, vol. 12, p. 5, 2018.

[21] R. S. Hartenberg and J. Denavit, "A kinematic notation for lower pair mechanisms based on matrices," *Journal of applied mechanics*, vol. 77, no. 2, pp. 215–221, 1955.

[22] Franka emika. [Online]. Available: https://www.franka.de

[23] P. Lightbody, T. Krajník, and M. Hanheide, "An efficient visual fiducial localisation system," *ACM SIGAPP Applied Computing Review*, vol. 17, no. 3, pp. 28–37, 2017.

[24] T. Wiedemeyer, "IAI Kinect2," https://github.com/code-iai/iai_kinect2, Institute for Artificial Intelligence, University Bremen, 2014 – 2015, accessed June 12, 2015.

[25] A. Hornung, K. M. Wurm, M. Bennewitz, C. Stachniss, and W. Burgard, "OctoMap: An efficient probabilistic 3D mapping framework based on octrees," *Autonomous Robots*, 2013, software available at http://octomap.github.com. [Online]. Available: http://octomap.github.com

[26] I. A. Şucan, M. Moll, and L. E. Kavraki, "The Open Motion Planning Library," *IEEE Robotics & Automation Magazine*, vol. 19, no. 4, pp. 72–82, December 2012, http://ompl.kavrakilab.org.

Manuscript 1128 submitted to 2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS). Received March 1, 2019.