Robotics: Science and Systems 2021
Held Virtually, July 12–16, 2021

PROMPT: Probabilistic Motion Primitives based Trajectory Planning

Tobias Löw[†][**] Tirthankar Bandyopadhyay [†] Jason Williams [†] Paulo V K Borges [†]

***Abstract*—We present a novel approach to motion planning for autonomous ground vehicles by formulating motion primitives as probabilistic distributions of trajectories (aka probabilistic motion primitives - ProMP) and performing stochastic optimisation on them for finding an optimal path. We show that compared to the traditional approach of using discrete motion primitives or direct stochastic optimisation of the whole path, incorporating ProMPs enables higher quality of paths by enabling constraint conditioning, combination and blending of probability distributions. We present two motion planners utilizing this approach: feasibility based trajectory sampling (PROMPT-S) and stochastic gradient-based trajectory optimisation (PROMPT-O). We show simulation results of our approach outperforming state of the art optimisation as well as discrete motion primitives based planners.**

We additionally illustrate the versatility of our approach by showing PROMPT's ability to handle significantly skewed motion primitives, e.g, as induced by steering failure in AGVs as well as composition of motion primitives to perform complex manoeuvres. Finally, we demonstrate the practicality of these planners by implementing them on a real self-driving vehicle navigating on structured and unstructured off-road terrains.

I. INTRODUCTION

The success of motion primitives[*] in motion planning has been due to their ability to identify relevant motion patterns associated with different tasks that aids in reducing the intractable search space as well as act as a building block of complex motion sequences. Motion primitives [10] have been used widely in robotics literature to represent movement patterns both in high dimensional robotic systems like manipulators [3] and human motions [13] as well as for lower dimensional robots (e.g., AGVs, UAVs) with motion constraints [15]. To do so various representations have been proposed for motion primitives. *Atomic motion primitives* that represent a singleton action have been proposed in the context of search-based planning for manipulation [2]. The motion primitive sets are generated from sampling the control space to ensure suitable coverage of the state space [14], referred to as *control sampling primitives*. Alternatively, the state itself could be sampled and the controls solved by applying a boundary value problem solver as in the case of *state lattice primitives* [18]. To handle high dimensionality in manipulator arms, adaptive motion primitives [3] were proposed where static primitives with variable dimensionality was augmented with on-the-fly motions generated by analytical inverse kinematics solvers.

In the domain of human movement modelling and learning from demonstration (LfD), *movement* primitives have been introduced as a way to model complex human motions performing specific tasks or behaviours involving very high degrees of freedom systems into tractable lower dimensional representation. These movement primitives usually model tasks like pick and place as compared to kinodynamically constrained motion primitives used for AGVs, like turning with allowable curvatures. A powerful probabilistic representation of such movement primitives, probabilistic movement primitives (ProMP) was introduced and developed in [16, 17]. Similar formulations are used in LfD problems which aim to enable non-experts teach task specific skills to robots. In the case of ProMP, the trajectory can also be adapted online with human intention awareness [9]. We use this representation for our work and conceptually use the terms movement and motion interchangeably.

In addition to analytical formulations of the motion primitives, data driven approaches have also been used to represent motion models of complex systems in lower dimensionality as motion primitives. Kinematic motion primitives were extracted from EMG signals on human subjects to describe the complexity of human motions with reduced dimensionality [13]. More generally, dynamic motion primitives have been used to model attractor behaviours of autonomous non-linear dynamical systems using statistical learning techniques [6].

Recent work on discrete motion primitives, [25], introduces path groups as a mechanism to couple a set of deterministic trajectories whose combined quality (in terms of likelihood of collision free path) enables robust trajectory planning for partially known environments. In essence our approach takes a probabilistic approach of representing such trajectory bundles as Gaussian radial basis functions (GRBFs). We show how such a probabilistic representation improves the performance over its discrete counterparts. While the PROMPT formulation of this paper does not encode the likelihood of collision free path into its weight distributions, such additional parameters can be incorporated into our formulation by encoding a suitable cost function.

In this work, we formulate our motion primitives in the probabilistic motion primitive (ProMP) framework as introduced in [16]. Representing the motion primitive as a distribution rather than an atomic action allows for more flexibility in trajectory

[†]Robotics and Autonomous Systems Group, CSIRO, Pullenvale, QLD 4069 (Tirtha.Bandy@csiro.au, Jason.Williams@data61.csiro.au, Paulo.Borges@csiro.au)

[**]Idiap Research Institute, Martigny, Switzerland (tobias.loew@idiap.ch)

[*]In this paper motion primitives and movement primitives represent the same concept. We use motion primitives to bring consistency of terminology across the paper.

generation as the distribution could be conditioned based on the sampled goal position or the via points that need to be transited through. This enables additional constraints both physical (e.g., avoiding obstacles) and logical (e.g following a road lane) to be seamlessly added to constrain the trajectory.

Representing the trajectory as a weighted sum of an Gaussian radial basis function (GBRF) set allows the trajectory distribution to be encoded by the underlying weight distribution with just a few parameters. Additionally, these parameters of the weight distribution could either be learnt on the fly from the vehicle environmental interactions [11] or from an expert human drivers [24].

Stochastic optimisation algorithms for motion planning, e.g STOMP [7, 12] have been widely successful in a variety of scenarios of high complexity and our approach follows the same optimisation framework. However STOMP [7] does not utilise motion primitives but performs the optimisation on sampled trajectories in the neighbourhood of the initial possibly infeasible trajectory. Recent extensions Guided STOMP [12] use dynamic motion primitives to primarily initialise the stochastic optimisation planning problem, enabling significant generality of scenarios and tasks over existing learning from demonstration (LfD) approaches, however do not incorporate the motion primitives into the optimisation iterations. In our work, we restrict our sampling and optimisation from a trajectory distribution representing the motion primitives to generate better trajectories faster.

In essence, by incorporating two fundamental techniques, (1) representing the motion primitives in their probabilistic form (ProMP) and (2) utilizing the general stochastic optimisation approach to trajectory optimisation, we present a novel and powerful motion planning framework that outperforms the state of the art planners and can be generalized to a wide range of scenarios and robot kinematics. The main contributions of this paper are:

- a unified framework of utilizing ProMP for stochastic optimisation for trajectory planning
- to present two motion planning algorithms (PROMPT-O, PROMPT-S) under this framework
- to show the versatility of computing suitable trajectories of severely skewed motion primitive distributions
- to show the ability to generate optimal trajectories by combining multiple motion primitives in sequence
- to show the approach working on a real system.

We present the background of STOMP and ProMP in section II followed by our formulation of the problem and the planners in Section III. We show the comparison with other approaches in simulation and implementation results in an autonomous robots in Section IV.

II. BACKGROUND

In our formulation, we unify the two concepts of representing motion primitives as ProMP and performing stochastic optimisation over the local planning horizon. We take advantage of ProMP's Gaussian nature to find a compact, and continuously deformable, representation of the robot's motion capabilities inside this local window. In the following a brief overview of ProMP and stochastic optimisation with a focus on STOMP is presented.

A. Probabilistic Motion Primitives (ProMP)

Let a trajectory $\boldsymbol{\xi}$ be defined as sequence of states $\boldsymbol{\zeta}(s)$ over a progress variable $s \in [0,1]$, where s is normalised to 1 to allow for time modulation. The state vector $\boldsymbol{\zeta} \in \mathbb{R}^{n_s}$, consists of 3D position and orientation as a quaternion ($n_s = 7$). Following ProMP framework [16], we compute the trajectory $\boldsymbol{\xi}$ by a linear combination of weighted basis functions $\boldsymbol{\Phi}$ which itself is a matrix of Gaussian radial basis functions (GRBF) [4] with K kernels:

$$\boldsymbol{\Phi} = [\boldsymbol{\phi}_1 \dots \boldsymbol{\phi}_K] \otimes \mathbf{I}_{n_s} \tag{1}$$

where $\otimes$ denotes the Kronecker product, and $\boldsymbol{\phi}_k = [\phi_k(0), \dots, \phi_k(1)]^T$ represents the k-th RBF evaluated at different times $s \in [0,1]$, with $\phi_k(s)$ representing the RBF with the k-th mean evaluated at s.

The weight vector uniquely identifies a trajectory sample for a given basis function set. In our formulation, the weight vector is assumed to have a normal distribution, $\tilde{\boldsymbol{\mu}} \sim \mathcal{N}\{\boldsymbol{\mu}, \boldsymbol{\Sigma}\}$, so that a sampled trajectory can be constructed from the sampled weight as $\tilde{\boldsymbol{\xi}} = \boldsymbol{\Phi}\tilde{\boldsymbol{\mu}}$. The distribution of the weight vectors induces the trajectory distribution that in turn represents a single probabilistic motion primitive. The trajectory distribution then can be represented as

$$\tilde{\boldsymbol{\xi}} \sim \mathcal{N}\left(\boldsymbol{\Phi}\boldsymbol{\mu}, \boldsymbol{\Phi}\boldsymbol{\Sigma}\boldsymbol{\Phi}^T\right) \tag{2}$$

B. Stochastic Trajectory Optimisation

We perform stochastic trajectory optimisation similar to STOMP to compute an optimal trajectory for the robot. STOMP considers trajectory planning by solving an optimisation such as:

$$\min_{\boldsymbol{\xi}} \mathbb{E}\left[Q(\tilde{\boldsymbol{\xi}}) + \frac{1}{2}\tilde{\boldsymbol{\xi}}^T \mathbf{R}\tilde{\boldsymbol{\xi}}\right] \tag{3}$$

where $\tilde{\boldsymbol{\xi}} \sim \mathcal{N}(\boldsymbol{\xi}, \boldsymbol{\Upsilon})$ is a sampled trajectory whose discretised states are sampled from a normal distribution with a mean trajectory $\boldsymbol{\xi}$ and covariance matrix Υ. $Q(\tilde{\boldsymbol{\xi}})$ is the path cost and $\mathbf{R}$ is a positive semi-definite matrix representing control costs.

While STOMP was inspired by work in path integrals, an alternative way to motivate the solution is to replace the objective (3) by one that encodes the path cost $Q(\tilde{\boldsymbol{\xi}})$ through a probability measure, seeking the mean trajectory that maximises the likelihood:

$$\max_{\boldsymbol{\xi}} J(\boldsymbol{\xi}); \quad J(\boldsymbol{\xi}) = \mathbb{E}\left[e^{-\frac{1}{\lambda}Q(\tilde{\boldsymbol{\xi}})}\right] \tag{4}$$

where λ is a parameter that regulates the cost sensitivity, which is selected to adapt to the dynamic range of the cost values. It can be readily shown that an approximate Newton step for this optimisation is given by:

$$\delta\boldsymbol{\xi} = \frac{\sum_m (\tilde{\boldsymbol{\xi}}_m - \boldsymbol{\xi}) e^{-\frac{1}{\lambda}Q(\tilde{\boldsymbol{\xi}}_m)}}{\sum_m e^{-\frac{1}{\lambda}Q(\tilde{\boldsymbol{\xi}}_m)}} \tag{5}$$

where $\{\tilde{\boldsymbol{\xi}}_m\}$ are independent and identically distributed samples as drawn from (2).

III. Motion Planning Using Probabilistic Motion Primitives

A key advantage of using ProMPs comes from using a Gaussian distribution to represent trajectories that allows us to condition the trajectory distribution on desired constraints. This enables the primitive distribution to be easily modulated to pass through a desired state $\boldsymbol{\zeta}^*(s)$ with a covariance parameter $\boldsymbol{\Sigma}^*$, defining the tolerance of the desired constraint.

The conditioning is introduced as a pseudo-measurement with measurement matrix $\mathbf{H}_s = [\phi_1(s) \dots \phi_K(s)] \otimes \mathbf{I}_{n_s}$. This results in a new conditional trajectory distribution:

$$p(\tilde{\boldsymbol{\xi}}|s, \boldsymbol{\zeta}^*(s), \boldsymbol{\Sigma}^*) = \mathcal{N}(\tilde{\boldsymbol{\xi}}; \boldsymbol{\Phi}\bar{\boldsymbol{\mu}}, \boldsymbol{\Phi}\bar{\boldsymbol{\Sigma}}\boldsymbol{\Phi}^T) \tag{6}$$

The conditional mean $\bar{\boldsymbol{\mu}}$ and covariance $\bar{\boldsymbol{\Sigma}}$ can be found using Equations (7) and (8), respectively.

$$\bar{\boldsymbol{\mu}} = \boldsymbol{\mu} + \boldsymbol{\Sigma}\mathbf{H}_s^T(\boldsymbol{\Sigma}^* + \mathbf{H}_s\boldsymbol{\Sigma}\mathbf{H}_s^T)^{-1}(\boldsymbol{\zeta}^*(s) - \mathbf{H}_s\boldsymbol{\mu}) \tag{7}$$

$$\bar{\boldsymbol{\Sigma}} = \boldsymbol{\Sigma} - \boldsymbol{\Sigma}\mathbf{H}_s^T(\boldsymbol{\Sigma}^* + \mathbf{H}_s\boldsymbol{\Sigma}\mathbf{H}_s^T)^{-1}\mathbf{H}_s\boldsymbol{\Sigma} \tag{8}$$

While the above distribution is conditioned on the goal, it does not account for obstacles and other traversability limitations that might be encountered along the way. Finally, the product of Gaussians can be used to combine multiple distributions. This makes it possible to either bootstrap from a previous solution or bias towards a path from a global planner.

Using the above stochastic optimisation approach, we now present two planning algorithms that use probabilistic motion primitives for local trajectory planning of an AGV.

A. Probabilistic Motion Primitive Trajectory Sampling (PROMPT-S)

The first approach, Probabilistic Motion Primitive Trajectory Sampling (PROMPT-S), is shown in Algorithm 1. The motion primitive distribution is first conditioned according to (6) with $s = 1$ and $\boldsymbol{\zeta}^*(s = 1)$ the current local goal. The local goal or via point selected in the local planning horizon is either given directly as input to the local planner or selected from the global path in a rolling window fashion.

Trajectories are sampled directly from the conditional distribution outlined in Equations (7, 8) and infeasibility checks are performed to reject invalid trajectories. Note that to remove temporal dependency between the states, the diagonal of the covariance could be used for sampling as in [8], but in this instance we wanted to retain the temporal dependency, thus the full covariance matrix is utilised. The remaining valid trajectory samples are used to computed the new conditional distribution parameters, e.g mean and covariance. The algorithms runs iteratively until the mean trajectory passes the feasibility check and is the mean trajectory is used as the output of the planner. An advantage of iterating until the mean trajectory is feasible, as compared to selecting one of the sampled feasible trajectory is that any future iterations would produce feasible trajectories from the new conditional distribution with very high probability.

Algorithm 1: PROMPT-S

```
Data: trajectory distribution
        $p(\tilde{\boldsymbol{\xi}}) = \mathcal{N}\left(\tilde{\boldsymbol{\xi}}; \boldsymbol{\Phi}\boldsymbol{\mu}, \boldsymbol{\Phi}\boldsymbol{\Sigma}\boldsymbol{\Phi}^T\right)$
        current robot state $\boldsymbol{\zeta}(s = 0)$
        goal state $\boldsymbol{\zeta}(s = 1)$
Result: trajectory $\boldsymbol{\xi}$
1  obtain conditional distribution $q(\tilde{\boldsymbol{\mu}}) = \mathcal{N}(\tilde{\boldsymbol{\mu}}; \bar{\boldsymbol{\mu}}, \bar{\boldsymbol{\Sigma}})$
2      use equations (7) and (8)
3  assign initial parameter vector $\bar{\boldsymbol{\mu}}^* \leftarrow \bar{\boldsymbol{\mu}}$
4  while $\boldsymbol{\xi} = \boldsymbol{\Phi}\bar{\boldsymbol{\mu}}^*$ is not feasible do
5      initialize empty list of feasible samples $L$
6      draw $M$ samples $\tilde{\boldsymbol{\mu}}_m$ from $q(\tilde{\boldsymbol{\mu}}) = \mathcal{N}(\tilde{\boldsymbol{\mu}}; \bar{\boldsymbol{\mu}}^*, \bar{\boldsymbol{\Sigma}})$
7      for each sample $\tilde{\boldsymbol{\mu}}_m$:
8          if sample $\tilde{\boldsymbol{\mu}}_m$ is feasible then
9              add $\tilde{\boldsymbol{\mu}}_m$ to the list of feasible samples $L$
10         else
11             reject $\tilde{\boldsymbol{\mu}}_m$
12     $\bar{\boldsymbol{\mu}}^* \leftarrow$ mean of all samples in $L$
13 obtain optimised trajectory $\boldsymbol{\xi} = \boldsymbol{\Phi}\bar{\boldsymbol{\mu}}^*$
```

Additionally, the mean trajectory is more robust to control errors during execution as effectively it is at the centre of a bundle of feasible trajectories.

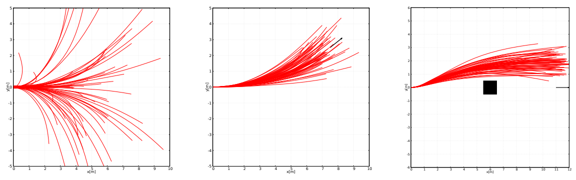

(a) (b) (c)

Figure 1. The effect of conditioning and distribution convergence in presence of obstacles. (a) shows samples drawn from a nominal motion primitive distribution, (b) shows samples drawn from a distribution conditioned on a via pose (8, 3, 45°), (c) shows samples drawn at 41 iterations from a distribution conditioned on the via pose (12, 0, 0°) and in the presence of an obstacle at (6,0)

Figure 1 shows how the probability distribution and the subsequent sampled trajectories are affected by the conditioning of a local goal or a via pose and the obstacles. Here the trajectories generated with the same time limit. The spatial discrepancy in the trajectory length come about as different velocities are being sampled as well and executed for the same time limit. In Figure 1(a) the samples are generated from an unconditioned motion primitive distribution. Using Equation (7,8) to condition the distribution on the goal or a via pose we can sample and evaluate suitable trajectories from a smaller subset of the whole trajectory space as seen in Figure 1(b) thereby increasing the effectiveness of planning. Similarly in Figure 1(c) as the sampled trajectories that collide with the obstacles are discarded, we find the trajectory distribution starting to converge on a set of solution paths that satisfy both the kinematic and goal constraints.

In reality, real solution distribution is multi-modal, and our gaussian modeling is inherently unimodal, the mean trajectory of the distribution might not be collision free. This however does not affect our approach negatively, as the gaussian parameters are only used for generating the sampled trajectories. Only those trajectories that are collision free, satisfy all the kinematic constraints and have the lowest cost are selected and passed to the robot during run time. As the algorithm iterates, the distribution *snaps* on to one of the homology class e.g the top section in Figure 1(c). A more general approach of maintaining multiple distributions across different homology classes as introduced in [22] for the TEB planner can be adopted to mitigate this effect.

B. Probabilistic Motion Primitive Trajectory Optimisation (PROMPT-O)

Algorithm 2: PROMPT-O

```
Data: trajectory distribution $p(\boldsymbol{\xi}) \sim \mathcal{N}\left(\boldsymbol{\Phi\mu}, \boldsymbol{\Phi}\Sigma\boldsymbol{\Phi}^T\right)$
      current robot state $\boldsymbol{\zeta}(s=0)$
      goal state $\boldsymbol{\zeta}(s=1)$
Result: optimised trajectory $\boldsymbol{\xi}$
1  while not converged do
2      obtain importance density $\rho(\tilde{\boldsymbol{\mu}}) = \mathcal{N}\{\tilde{\boldsymbol{\mu}}; \bar{\boldsymbol{\mu}}, \bar{\boldsymbol{\Sigma}}\} \propto \mathcal{N}\{\tilde{\boldsymbol{\mu}}; \boldsymbol{\mu}, \boldsymbol{\Sigma}\}\mathcal{N}\{\boldsymbol{\zeta}^*(s); \mathbf{H}_s\tilde{\boldsymbol{\mu}}, \boldsymbol{\Sigma}^*\}$
3          use equations (7) and (8)
4      draw $M$ samples $\tilde{\boldsymbol{\mu}}_m$ from $\rho(\tilde{\boldsymbol{\mu}})$
5      for each sample $\tilde{\boldsymbol{\mu}}_m$:
6          calculate trajectory cost $Q(\boldsymbol{\Phi}\tilde{\boldsymbol{\mu}}_m)$
7              use equation (10)
8          calculate importance weight and cost $\pi_m$
9              use equation (16)
10     calculate update $\delta\boldsymbol{\mu}$
11         use equation (18)
12     apply parameter update to $\boldsymbol{\mu}$
13         use equation (17)
14 obtain optimised trajectory $\boldsymbol{\xi} = \boldsymbol{\Phi\mu}$
```

We now present a stochastic trajectory optimisation variant of the previous planner, probabilistic motion primitive trajectory optimisation planner (PROMPT-O) in Algorithm 2. As in Equation (4), we formulate the motion planning problem as an optimisation that minimises costs e.g. smoothness, collisions and kinodynamic constraints, along the trajectory:

$$\boldsymbol{\mu}^* = \arg\max_{\boldsymbol{\mu}} J(\boldsymbol{\mu}); \quad J(\boldsymbol{\mu}) = \mathbb{E}\left[e^{-\frac{1}{\lambda}Q(\tilde{\boldsymbol{\xi}})}\right] \tag{9}$$

where $\tilde{\boldsymbol{\xi}}$ is a trajectory distributed according to Equation (2). $Q(\boldsymbol{\xi})$ denotes a general cost function for a single trajectory $\boldsymbol{\xi}$ that is usually expressed as a summation of state dependent cost values:

$$Q(\boldsymbol{\xi}) = \int_{s=0}^{s=1} q(\boldsymbol{\zeta}(s))ds \tag{10}$$

where for $\boldsymbol{\xi} = \boldsymbol{\Phi\mu}$, the state at progress s is $\boldsymbol{\zeta}(s) = \mathbf{H}_s\boldsymbol{\mu}$. Expanding the objective and taking the gradient and Hessian, we find:

$$J(\boldsymbol{\mu}) = \int \mathcal{N}\{\tilde{\boldsymbol{\mu}}; \boldsymbol{\mu}, \boldsymbol{\Sigma}\}e^{-\frac{1}{\lambda}Q(\boldsymbol{\Phi}\tilde{\boldsymbol{\mu}})}\mathrm{d}\tilde{\boldsymbol{\mu}} \tag{11}$$

$$\nabla_{\boldsymbol{\mu}} J(\boldsymbol{\mu}) = \boldsymbol{\Sigma}^{-1}\int (\tilde{\boldsymbol{\mu}} - \boldsymbol{\mu})\mathcal{N}\{\tilde{\boldsymbol{\mu}}; \boldsymbol{\mu}, \boldsymbol{\Sigma}\}e^{-\frac{1}{\lambda}Q(\Phi\tilde{\boldsymbol{\mu}})}\mathrm{d}\tilde{\boldsymbol{\mu}} \tag{12}$$

$$\nabla^2_{\boldsymbol{\mu}} J(\boldsymbol{\mu}) = \int \left[-\boldsymbol{\Sigma}^{-1} + \boldsymbol{\Sigma}^{-1}(\tilde{\boldsymbol{\mu}} - \boldsymbol{\mu})(\tilde{\boldsymbol{\mu}} - \boldsymbol{\mu})^T\boldsymbol{\Sigma}^{-1}\right]\times \mathcal{N}\{\tilde{\boldsymbol{\mu}}; \boldsymbol{\mu}, \boldsymbol{\Sigma}\}e^{-\frac{1}{\lambda}Q(\boldsymbol{\Phi}\tilde{\boldsymbol{\mu}})}\mathrm{d}\tilde{\boldsymbol{\mu}} \tag{13}$$

The challenge of estimating the gradient in Equation (12) is the inefficiency of sampling from $\mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma})$, since it represents the entire motion capability of the robot and therefore would take a long time to converge to a solution that is applicable to the current situation. Thus, in order to increase sampling efficiency we apply an importance sampling scheme where the sampling distribution incorporates the destination encoded in the cost $Q(\tilde{\boldsymbol{\xi}})$. Instead, we sample from an importance density which incorporates a Gaussian approximation of this destination. Specifically, let:

$$\begin{aligned} \rho(\tilde{\boldsymbol{\mu}}) &= \mathcal{N}\{\tilde{\boldsymbol{\mu}}; \bar{\boldsymbol{\mu}}, \bar{\boldsymbol{\Sigma}}\} \\ &\propto \mathcal{N}\{\tilde{\boldsymbol{\mu}}; \boldsymbol{\mu}, \boldsymbol{\Sigma}\}\mathcal{N}\{\boldsymbol{\zeta}^*(s); \mathbf{H}_s\tilde{\boldsymbol{\mu}}, \boldsymbol{\Sigma}^*\} \end{aligned} \tag{14}$$

where the conditional mean $\bar{\boldsymbol{\mu}}$ and covariance $\bar{\boldsymbol{\Sigma}}$ can be found using (7) and (8), respectively. Then taking M samples $\tilde{\boldsymbol{\mu}}_m \sim \rho(\tilde{\boldsymbol{\mu}})$, the gradient (12) can be estimated as:

$$\nabla_{\boldsymbol{\mu}} J(\boldsymbol{\mu}) \approx \frac{\boldsymbol{\Sigma}^{-1}}{M}\sum_{m=1}^{M}(\tilde{\boldsymbol{\mu}}_m - \boldsymbol{\mu})\pi_m \tag{15}$$

where π_m incorporates the importance weight and cost:

$$\pi_m = \frac{\mathcal{N}\{\tilde{\boldsymbol{\mu}}_m; \boldsymbol{\mu}, \boldsymbol{\Sigma}\}}{\mathcal{N}\{\tilde{\boldsymbol{\mu}}_m; \bar{\boldsymbol{\mu}}, \bar{\boldsymbol{\Sigma}}\}}e^{-\frac{1}{\lambda}Q(\boldsymbol{\Phi}\tilde{\boldsymbol{\mu}}_m)} \tag{16}$$

A key property for importance sampling to be applied is that the proposal density should have heavier tails than the desired density. On the surface, it might appear that this is not satisfied as the proposal distribution in the denominator of (16) is narrower than the numerator. However, if we view the trajectory cost as a part of the desired distribution, the encoding of the goal into $Q(\boldsymbol{\xi})$ and the goal factor in the proposal distribution can be selected to be in balance. Since we expect the covariance conditioned on the goal to be much smaller than the prior covariance $\boldsymbol{\Sigma}$, we can approximate the Hessian in (13) by dropping the second term in the sum. This provides an approximate Newton method through the update

$$\boldsymbol{\mu} \leftarrow \boldsymbol{\mu} + \alpha\delta\boldsymbol{\mu} \tag{17}$$

where α represents the step size, and the update $\delta\boldsymbol{\mu}$ is:

$$\delta\boldsymbol{\mu} = \frac{1}{c}\sum_{m=1}^{M}(\tilde{\boldsymbol{\mu}}_m - \boldsymbol{\mu})\pi_m, \quad c = \sum_{m=1}^{M}\pi_m \tag{18}$$

As before, unlike [8] we dont restrict our sampling to the diagonal of the covariance $\bar{\boldsymbol{\Sigma}}$ to preserve the temporal dependency, thus yielding samples that are likely to be kinematically feasible. Note that in Equation (17) as opposed to the update rule of STOMP, we do not require the gradient estimate to be smoothed with the scaled covariance matrix.

With PROMPT-S and PROMPT-O we presented two different approaches for exploiting Probabilistic Motion Primitives. It is important to note that PROMPT-S presents a naive implementation of using ProMP which is meant to provide a baseline behaviour that is shown to outperform state of the art algorithms like STOMP, clearly demonstrating the implicit advantage of using such a motion primitive formulation. PROMPT-O is a sophisticated algorithm that proposes the integrated approach for optimal trajectory generation on top of the ProMP formulation. Ideally, PROMPT-O is preferred to PROMPT-S under most conditions with well-defined cost structures.

C. Analysis and Discussion

Each iteration of PROMPT-S generates M samples, yielding complexity $O(MK{n_s}^2 n_t)$, where K is the number of GRBF kernels, $n_s = 7$ is the state space dimensionality, and n_t is the number of time steps evaluated along the trajectory for each kernel. Each iteration of PROMPT-O draws M samples, and evaluates the trajectory cost and the importance weight, yielding complexity $O(MK{n_s}^2 n_t + MK^3 n_s^3)$. In practice, we find that 40 samples are sufficient, and that convergence is generally achieved after 30 iterations for PROMPT-S and PROMPT-O. We limit the number of iterations to 200, and have not found this to be a problem in practice.

An important benefit in comparison to STOMP is the continuous-time nature of the motion primitive trajectories. It allows for a state to be queried for any arbitrary $s \in [0, 1]$. This makes it possible to have a non-uniform or adaptive discretization when evaluating the cost functions numerically. Both versions of PROMPT implicitly capture the robot kinematics with the ProMP representation of the trajectories. Therefore as opposed to STOMP the kinematics model does not need to be enforced from the outside via the cost function, it only needs to filter out the low probability samples which violate these constraints. Using ProMP and the forward kinematics with different velocities also allows for capturing time dependency resulting in smoother trajectories than STOMP. Furthermore, it provides an interface between classical trajectory planning approaches and modern machine learning techniques via the motion primitive distribution. This distribution can either be learned from demonstrations or explicitly modelled with the known kinematics. A combination is also possible, in which case the demonstrations can be thought of as a calibration of the specific robot or task. This is also where the fundamental difference to learning from demonstrations (LfD) lies in the use of ProMP. We use the ProMP distribution as a representation for the motion capabilities of the robot, whereas LfD encodes specific manipulation tasks and their variances into the distribution.

In the case of ground vehicles the benefit of applying LfD to the utilised ProMP distribution could lie in encoding specific driving styles; e.g., consider that every passenger has their personal preferences for driving a vehicle. By using LfD, those preferences could be taught and reproduced by the autonomous vehicle using our planning approach.

IV. EXPERIMENTS

To understand the nature of the PROMPT planning framework and its efficacy to current state-of-the-art planners, we compare our planners (PROMPT-O, PROMPT-S) with two classes of planners: search based discrete motion primitives planner FALCO([25]) and trajectory optimisation based planners STOMP([7]) and TEB([20]). While there are numerous variants to these planners, these three have been chosen due to their demonstrated improved performance over others. For comparison, publicly available code from the authors have been used without modification.

A. Experimental setup

We treat all the planners mentioned above in a local planning framework where a global path is provided as a prior, here computed by an A^* algorithm. The global planner is agnostic to the robot kinematics and only outputs via poses without orientations. The local planner converts this global path segment into executable local trajectories. Once the local planner generates a feasible solution, the trajectory is executed for 1s and the updated state is used to re-plan a new trajectory with updated via point in an online fashion. All the planners are given full obstacle information within the local planning horizon with a local goal or a via point selected from the global path to plan towards. All the planners are given the same time limit for planning. No additional smoothing is done on any of the trajectory outputs.

B. Motion Primitive Distribution

For the PROMPT planners, a data driven approach has been applied to generate the AGV's motion primitive distribution. Specifically for this paper the motion primitive distribution was generated using forward simulation of a simplified kinematics model of a car-like robot representing the AGV. In general other robot kinematics, can easily be incorporated into the framework.

To perform forward simulation, we discretize the input space (i.e., the control commands) and record the output (i.e, the trajectory) of the forward kinematics, which leaves us with a set of individual motion trajectories. By calculating the mean $\boldsymbol{\mu}$ and covariance $\boldsymbol{\Sigma}$ of the set, the motion primitive distribution can be obtained $p(\boldsymbol{\xi}) \sim \mathcal{N}(\Phi\boldsymbol{\mu}, \boldsymbol{\Phi}\boldsymbol{\Sigma}\boldsymbol{\Phi})$. Backward driving, and backward primitives can be easily incorporated by selective negative velocity commands, but for this paper we only use forward motion of AGVs to simplify comparison with other planners.

C. Cost Function

The cost given by Equation (10) is computed by discretizing $\boldsymbol{\xi}$ into N time steps between $s_0 = 0$ and $s_{N-1} = 1$ and adding the costs of each state $\boldsymbol{\zeta}(s_n)$. We use two classes of cost functions: state costs q_s (e.g., distance from obstacles) and transition costs q_t (e.g., kinematics). Expressing the resulting summation using $\boldsymbol{\zeta}(s_n) = \mathbf{H}_{s_n}\boldsymbol{\mu}$ yields:

$$Q(\boldsymbol{\Phi}\boldsymbol{\mu}) = \sum_{n=0}^{N} q_s(\mathbf{H}_{s_n}\boldsymbol{\mu}) + \sum_{n=1}^{N} q_t(\mathbf{H}_{s_{n-1}}\boldsymbol{\mu}, \mathbf{H}_{s_n}\boldsymbol{\mu}) \tag{19}$$

Here $\mathbf{H}_{s_n}$ is a measurement matrix as defined previously in (6). The basis functions have parameters $K = 6$ and a variance of 0.05 for each radial basis function. The obstacle cost is defined as the Euclidean distance transform of the boolean-valued occupancy grid map, analogously to other methods in the literature [19, 7]. As for the kinematics constraint, since we are simulating a non-holonomic robot, we follow the constraint derived in [21] defined by:

$$\left(\begin{bmatrix} \cos(\varphi_{t-1}) \\ \sin(\varphi_{t-1}) \\ 0 \end{bmatrix} + \begin{bmatrix} \cos(\varphi_t) \\ \sin(\varphi_t) \\ 0 \end{bmatrix} \right) \times \Delta\boldsymbol{\zeta} = 0 \tag{20}$$

The deviation from zero for each each pair of states is given as cost to planning algorithms. This ensures that the robot kinematic errors are minimized. For the experiments, $q_{kin,max} = 0.2$ empirically proved to result in kinematically feasible trajectories.

PROMPT is able to exploit additional cost functions during its optimisation step. E.g, the additional cost-to-go value for the local grids, which is a byproduct of the A^* planner can be provided along with the A^* path in the local planning horizon, enabling to converge to the optimal solutions faster. Note however that in many cases at runtime the local planner has to take into account obstacles which the global planner might not have knowledge of, e.g road obstruction, pedestrian movement etc. While the global path and the cost-to-go values aid in guiding the optimisation, a valid runtime evaluation and feasibility is performed ensuring that the trajectories generated at run time are truly feasible.

D. *Qualitative Comparison*

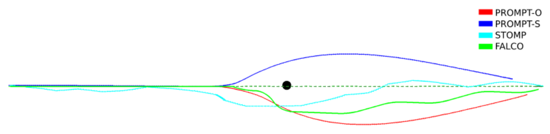


Figure 2. Behaviour of different planners when encountering a single obstacle. The whole environment is 20m long, the planners are given local planning horizon of 10m and are executed at 1Hz with a vehicle velocity of 1m/s. The trajectories displayed above have been computed online in a receding horizon approach.

a) Single Obstacle Scenario: In order to get an insight about the behaviour of the different planners we used for comparison we first simulated a single obstacle scenario which can be seen in Figure 2. To bring out the local adaptability of each planner in a priori unknown environments, we purposefully gave a path segment in collision to each planner.

We see from Figure 2 that although all the planners are able to avoid the unseen obstacle successfully, their output trajectories give an insight into the optimisation performance. As both PROMPT-O and PROMPT-S are sampled from a gaussian distribution, they are inherently smooth. Stochastic optimisation over non-gaussian distribution, as done by STOMP, generates a feasible solution but incurs higher kinematic cost as per Equation (20). It is possible that STOMP converges to a better solution with more time. FALCO effectively chooses suitable motion primitives to curve away from the obstacle at a close range ensuring fast collision avoidance. However, the effect of motion primitive discretization becomes apparent in the wake of the obstacle response as seen by the wavy nature of the trajectory where motion primitives of different curvatures are stitched together. This is the inherent difference between representing the motion primitives as discrete curvature trajectories compared to PROMPT's gaussian distribution representation which enables a seamless conditioning of the previous state generating a highly smooth trajectory.

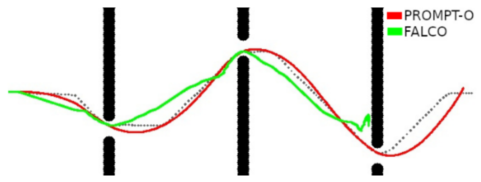


Figure 3. Behaviour of PROMPT-O compared to FALCO when passing multiple narrow passages. While FALCO computes the shortest path, PROMPT generates smoother paths. The whole environment is 70m long, with the walls spaced 20m from each other. The planners are given local horizon of 10m and are executed at 1Hz with a vehicle velocity of 1m/s. As before, the trajectories displayed above have been computed online in a receding horizon approach.

b) Narrow Passage Scenario: In more constrained scenarios, the limitations of discretization become apparent as can be seen in Figure 3. Here the FALCO and PROMPT-O are tasked with navigating multiple narrow passages. The grey dotted line denotes the A^* global path provided to both the planners. FALCO computes a much shorter path, but its local discrete primitive selection limits it from reaching the goal when it gets too close to the obstacle. This can be seen near the third barrier where the robot reaches a position and orientation from which FALCO is unable to execute a forward trajectory out. Note however that enabling backward motion primitive or a recovery mode would prevent a robot collision. However these manoeuvres are expensive and the planner should avoid such scenarios as much as possible. PROMPT-O on the other hand is able to generate a smooth path. This improved behaviour in more constrained environments is borne out by substantive quantitative experiments presented ahead.

E. *Quantitative Comparison*

Similar to [23], a randomised obstacle field (Figure 4), where the position and the radius of the circular obstacles is drawn from a uniform distribution, is used to compare the quantitative performance of all the planners. The size and number of obstacles is varied to generate cluttered environments of varying obstacle ratio (obstacle vs free space).

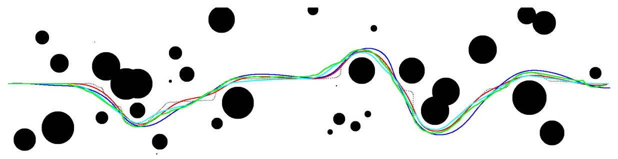


Figure 4. All planner's successful navigation through a randomised obstacle field. Figure shows the driven paths for PROMPT-O (red), PROMPT-S (blue), STOMP (light blue). TEB (yellow). FALCO (green).

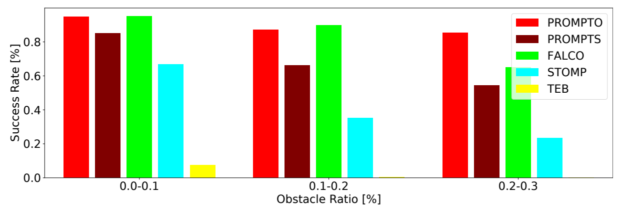
For the presented results the number of obstacles ranged between 20 and 100 (step size 5), their maximum radius between $3\,\mathrm{m}$ and $7\,\mathrm{m}$ (step size 0.5m). For each combination 15 different scenarios were generated. This equals a total of 1920 unique scenarios. The length of the entire obstacle field equals $200\,\mathrm{m}$ and its width is $50\,\mathrm{m}$.

The start and goal positions for each scenario always remain the same. Additionally, all the planners are given a precomputed successful global A^* path with the same robot footprint, which ensures the existence of a connected free space component containing the solution trajectory. All planners consider the same two cost functions to determine the feasibility of a trajectory: (1) the local perception of the obstacle field and (2) the kinematic constraints of the vehicle. As a consequence a planner fails as soon as it cannot find an executable trajectory that avoids all obstacles. The planners were constrained to maintain a minimum of 1.5m from obstacles and all the planners operated at $1\,\mathrm{Hz}$. Each planner is truncated if it fails to return a valid path under 1000 iterations. For stochastic optimisation planners like STOMP and PROMPT variants, the planned trajectory at the timeout is considered for evaluation, unless they hit a successful stopping criteria and return a solution earlier.

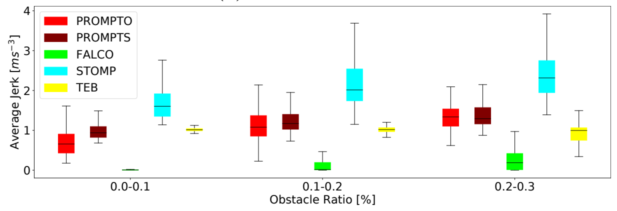
a) Performance metrics: To capture the smoothness of the path, we choose the average jerk and average curvature as performance metrics. Both jerk and curvature were summed along the trajectory and then divided by the number of points to get the average jerk and curvature, respectively. The average jerk value tells us the speed variation and the control effort required to maintain the speed of a vehicle to execute the trajectory. Lower jerk values pertain to temporally smooth motion. Similarly average curvature captures the steering behaviour of the vehicle. The higher the curvature, the higher the control effort in maintaining a smooth steering output. We also evaluate the average planning time taken per online local planning horizon to evaluate the rate at which the planners performed.

b) Results: These metrics are computed and plotted in Figure 5 against varying obstacle density ratio. As the obstacle density increases, the scenario becomes more cluttered making the planning more difficult.

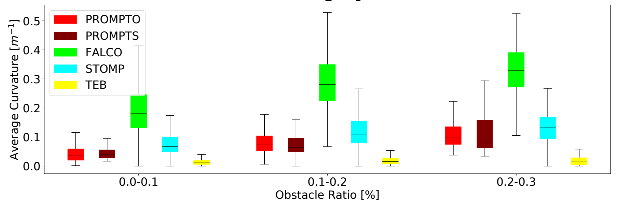
Figure 5(a) shows the success rates for each planner effectively decreases with increasing obstacle density as expected. TEB consistently under performs as compared to others, possibly due to the timeout constraints. Among the stochastic optimisation based planners, PROMPT variants outperform



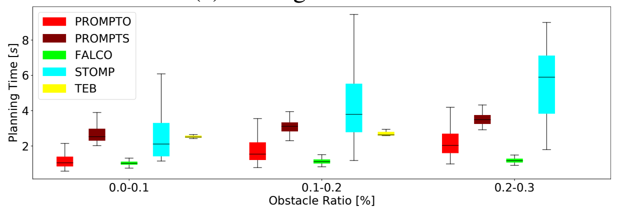
(a) Success rates.



(b) Average jerk



(c) Average curvature



(d) Total planning time for a successful run.

Figure 5. Quantitative results of the runs for the planners listed in the Figure 5 for randomized obstacle fields

STOMP because of more efficient sampling from a biased distribution. Without the knowledge of the vehicle kinematics, many of the STOMP trajectories have to be discarded due to high kinematic costs. With increased timeout values, the performance of STOMP could be improved. PROMPT-O outperforms both STOMP and PROMPT-S as incorporating importance sampling greatly improves the convergence over other stochastic optimisation approaches.

The discrete motion primitives based planner, FALCO, performs extremely well, matching the performance of PROMPT-O for lower and medium obstacle densities. This is primarily because there is sufficient free space for a sufficient set of precomputed motion primitives to be valid. From this set a suitable selection of motion primitives leads to successful navigation. However, with increasing obstacle density, this feasibility set gets depleted leading to lower success rates. PROMPT-O clearly outperforms FALCO at higher obstacle densities due to its continuous representation of motion primitives allowing for sufficient samples. The statistical result here confirms the insight from Figure 3.

From Figure 5(b-c) we see that the PROMPT planners have better curvature metrics than both STOMP and FALCO, and better jerk metric than STOMP. Precomputing discrete motion primitive with vehicle kinematics allows FALCO to have a much smoother speed variation as compared to PROMPT or STOMP.

In terms of the planning time as seen in Figure 5(d), we find STOMP takes a much longer planning time compared to other planners and FALCO is extremely fast due to its precomputed path groups. The PROMPT planners perform better than STOMP but worse than FALCO. The faster convergence due to importance sampling based optimisation of PROMPT-O as compared to sample rejection of PROMPT-S becomes evident here as well.

Overall in terms of the success rates, the quality of paths and planning time, PROMPT variants, especially PROMPT-O, matches the state of the art of both search based discrete motion primitives planners and stochastic optimisation planners in some metrics and outperforms in others.

F. Planning Under Degraded Actuation

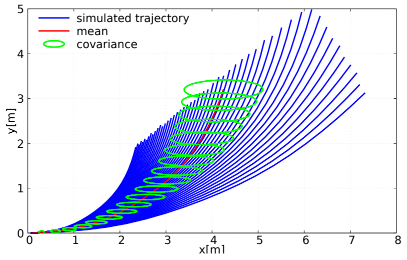


Figure 6. Skewed motion primitive distribution that is the result of e.g. a broken steering actuator.

A key advantage of representing the motion primitives as a probabilistic distribution is the ability to seamlessly adapt the parameters either by physical or functional constraints. So in a scenario where the robot's steering is broken, and only has partial angular actuation, we simply need to adjust the underlying motion primitive distribution accordingly for the planners. Figure 6 displays the individual trajectories that were used to form the distribution. In this example the steering angle was limited to $\varphi \in [0.1, 0.7]$ leading to a highly skewed mean trajectory. Figure 7 shows that, as expected, the robot drives in spirals if it is told to follow a global path. More importantly, when introduced to an obstacle field, as before the planner adapts its curvature conditioned on the obstacle while optimizing seemlessly for open spaces. This shows the versatility of PROMPT in incorporating a variety of motion primitive distributions for stochastic optimisation.

G. Hardware experiments

The field tests were conducted using a full-size AGV operating on a large outdoor site, with the vehicle performing localisation against a pre-built 3D map. For map generation and localisation, we use LiDAR-inertial mapping, employing a 3D SLAM algorithm [1, 5]. The AGV sets its reference path using an A*-based global planner, and as the vehicle moves the path is adapted locally using the proposed method.

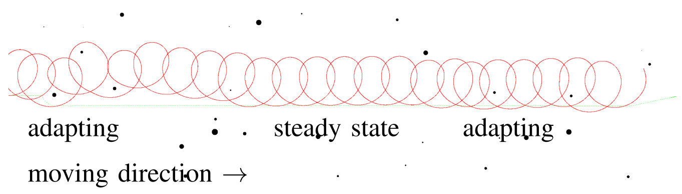


Figure 7. Broken robot moving in spirals through a randomised obstacle field. It can be seen that it adapts the curvature to avoid obstacles and settles to a steady state behaviour if there are no obstacles.

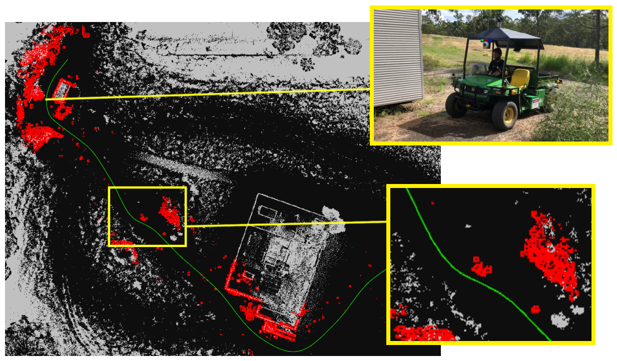


Figure 8. PROMPT-O was implemented on an autonomous AGV shown in the inset top right. The green path shows the trajectory executed by the vehicle, the grey points show occupancy grid obstacles and the red markers show local obstacles detected by the AGV's spinning laser. The inset at the bottom right shows the vehicle avoiding obstacles.

On average the trajectory planning was computed in under 200ms for the planning horizon of 10m. The path from a typical experimental run is plotted in Figure 8. The top right inset view of the vehicle shows a snapshot of the path where the vehicle negotiates a narrow passage between a bush on the left (which as not present in the original map) and the building wall on the right. The bottom right inset also shows the robot avoiding an obstacle (tall grass in the centre) on its path. This vegetation was not present in the original map and hence was not considered by the global planner, but was avoided by the proposed local planner. It can be seen that the planner is responding properly to obstacles like tree branches and grass patch as these appear as solid obstacles in the projected 2D plane of the planner at run time.

V. CONCLUSION

We presented a unified approach in incorporating motion primitives under the Probabilistic Motion Primitives framework and stochastic optimisation to compute an optimised trajectory. We have shown that using probabilistic movement primitives for motion planning of ground vehicles greatly improves not only the success rate of the algorithms but also the quality of plans. The versatility of the approach can be seen by introducing reduced mobility as induced by steering failure for an AGV and showing that our approach is able to handle such limitations seamlessly. The implementation results on a real autonomous vehicle shows the efficacy of the planners. The advantage of representing the motion primitive as a parameterized distribution allows us to integrate various learning paradigms seamlessly into our approach.

References

[1] Michael Bosse and Robert Zlot. Continuous 3d scan-matching with a spinning 2d laser. In *Robotics and Automation, 2009. ICRA'09. IEEE International Conference on*, pages 4312–4319. IEEE, 2009.

[2] B. J. Cohen, S. Chitta, and M. Likhachev. Search-based planning for manipulation with motion primitives. In *2010 IEEE International Conference on Robotics and Automation*, pages 2902–2908, May 2010. doi: 10.1109/ROBOT.2010.5509685.

[3] B. J. Cohen, G. Subramania, S. Chitta, and M. Likhachev. Planning for manipulation with adaptive motion primitives. In *2011 IEEE International Conference on Robotics and Automation*, pages 5478–5485, May 2011. doi: 10.1109/ICRA.2011.5980550.

[4] M D. Buhmann. Radial basis functions: Theory and implementations. *Radial Basis Functions*, 12, 01 2003. doi: 10.1017/CBO9780511543241.

[5] P. Egger, P. V. K. Borges, G. Catt, A. Pfrunder, R. Siegwart, and R. Dubé. Posemap: Lifelong, multi-environment 3d lidar localization. In *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 3430–3437, Oct 2018.

[6] A. J. Ijspeert, J. Nakanishi, H. Hoffmann, P. Pastor, and S. Schaal. Dynamical movement primitives: Learning attractor models for motor behaviors. *Neural Computation*, 25(2):328–373, Feb 2013. ISSN 0899-7667. doi: 10.1162/NECO_a_00393.

[7] Mrinal Kalakrishnan, Sachin Chitta, Evangelos Theodorou, Peter Pastor, and Stefan Schaal. STOMP: Stochastic trajectory optimization for motion planning. *Proceedings - IEEE International Conference on Robotics and Automation*, (May):4569–4574, 2011. ISSN 10504729. doi: 10.1109/ICRA.2011.5980280.

[8] Dorothea Koert, Guilherme Maeda, Rudolf Lioutikov, Gerhard Neumann, and Jan Peters. Demonstration based trajectory optimization for generalizable robot motions. *IEEE-RAS International Conference on Humanoid Robots*, pages 515–522, 2016. ISSN 21640580. doi: 10.1109/HUMANOIDS.2016.7803324.

[9] Dorothea Koert, Joni Pajarinen, Albert Schotschneider, Susanne Trick, Constantin Rothkopf, and Jan Peters. Learning Intention Aware Online Adaptation of Movement Primitives. *Robotics and Automation Letters*, pages 100–107, 2019. ISSN 2377-3766. doi: 10.1109/LRA.2019.2928760.

[10] Steven M. LaValle. *Planning Algorithms*. Cambridge University Press, USA, 2006. ISBN 0521862051.

[11] Tobias Löw, Tirthankar Bandyopadhyay, and Paulo Borges. Identification of effective motion primitives for ground vehicles. In *to appear on IEEE International Conference On Intelligent Robots and Systems*, 2020.

[12] Bence Magyar, Nikolaos Tsiogkas, Bruno Brito, Mayank Patel, David Lane, and Sen Wang. Guided Stochastic Optimization for Motion Planning. *Frontiers in Robotics and AI*, 6(November):1–13, 2019. doi: 10.3389/frobt.2019.00105.

[13] Federico Moro, Nikos Tsagarakis, and Darwin Caldwell. On the kinematic motion primitives (kmps) - theory and application. *Frontiers in Neurorobotics*, 6:10, 2012. ISSN 1662-5218. doi: 10.3389/fnbot.2012.00010.

[14] S. Pancanti, L. Pallottino, D. Salvadorini, and A. Bicchi. Motion planning through symbols and lattices. In *IEEE International Conference on Robotics and Automation, 2004. Proceedings. ICRA '04. 2004*, volume 4, pages 3914–3919 Vol.4, April 2004. doi: 10.1109/ROBOT.2004.1308878.

[15] Aditya A Paranjape, Kevin C Meier, Xichen Shi, Soon-Jo Chung, and Seth Hutchinson. Motion primitives and 3d path planning for fast flight through a forest. *The International Journal of Robotics Research*, 34(3):357–377, 2015.

[16] Alexandros Paraschos, Christian Daniel, Jan R Peters, and Gerhard Neumann. Probabilistic movement primitives. In *Advances in neural information processing systems*, pages 2616–2624, 2013.

[17] Alexandros Paraschos, Christian Daniel, Jan Peters, and Gerhard Neumann. Using probabilistic movement primitives in robotics. *Autonomous Robots*, 42(3):529–551, 2018.

[18] M. Pivtoraiko and A. Kelly. Kinodynamic motion planning with state lattice motion primitives. In *2011 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 2172–2179, Sep. 2011. doi: 10.1109/IROS.2011.6094900.

[19] Nathan Ratliff, Matt Zucker, J. Andrew Bagnell, and Siddhartha Srinivasa. CHOMP: Gradient optimization techniques for efficient motion planning. pages 489–494, 2009. doi: 10.1109/robot.2009.5152817.

[20] Christoph Rosmann, Wendelin Feiten, Thomas Wosch, Frank Hoffmann, and Torsten Bertram. Efficient trajectory optimization using a sparse model. *2013 European Conference on Mobile Robots, ECMR 2013 - Conference Proceedings*, pages 138–143, 2013. doi: 10.1109/ECMR.2013.6698833.

[21] Christoph Rosmann, Frank Hoffmann, and Torsten Bertram. Kinodynamic trajectory optimization and control for car-like robots. *IEEE International Conference on Intelligent Robots and Systems*, 2017-Septe:5681–5686, 2017. ISSN 21530866. doi: 10.1109/IROS.2017.8206458.

[22] C. Rösmann, F. Hoffmann, and T. Bertram. Planning of multiple robot trajectories in distinctive topologies. In *2015 European Conference on Mobile Robots (ECMR)*, pages 1–6, 2015. doi: 10.1109/ECMR.2015.7324179.

[23] Dave Ferguson Thomas M. Howard, Colin J. Green, Alonzo Kelly. State Space Sampling of Feasible Motions for High-Performance Mobile Robot Navigation in Complex Environments. *Journal of Field Robotics*, 25 (1):1–17, 2014. doi: 10.1002/rob.

[24] E. Velenis, P. Tsiotras, and J. Lu. Aggressive maneuvers on loose surfaces: Data analysis and input parametrization. In *2007 Mediterranean Conference on Control Automation*, pages 1–6, 2007.

[25] Ji Zhang, Chen Hu, Rushat Gupta Chadha, and Sanjiv Singh. Falco: Fast likelihood-based collision avoidance with extension to human-guided navigation. *Journal of Field Robotics*, n/a(n/a). doi: https://doi.org/10.1002/rob.21952. URL https://onlinelibrary.wiley.com/doi/abs/10.1002/rob.21952.