Abstract:
We present a novel approach to path planning for robotic manipulators, in which paths are produced via iterative optimisation in the latent space of a generative model of robot poses. Constraints are incorporated through the use of constraint satisfaction classifiers operating on the same space. Optimisation leverages gradients through our learned models that provide a simple way to combine goal reaching objectives with constraint satisfaction, even in the presence of otherwise non-differentiable constraints. Our models are trained in a task-agnostic manner on randomly sampled robot poses. In baseline comparisons against a number of widely used planners, we achieve commensurate performance in terms of task success, planning time and path length, performing successful path planning with multi-obstacle avoidance on a real 7-DoF robot arm.

Keywords: generative modelling, path planning, manipulation, optimisation

1 INTRODUCTION
Path planning is a cornerstone of robotics. For a robotic manipulator, this generally consists of producing a sequence of joint states the robot needs to follow in order to move from a start to a goal configuration. This requires that the poses along the sequence are kinematically feasible while at the same time avoiding unwanted contact either by the manipulator with itself or with potential objects in the robot’s workspace. Due to its importance, path planning is a richly explored area in robotics [e.g 1, 2, 3, 4, 5]. However, traditional approaches are often marred by a number of issues. As the state-space dimensionality increases and constraints become more constrictive, the decreasing efficiency of traditional planning methods makes reactive behaviour computationally challenging.

While existing sampling and optimisation-based approaches to the planning problem can find solutions, they scale super-linearly with a robot’s degrees of freedom, and those that have optimality guarantees on resulting paths are guaranteed to achieve this only asymptotically, after infinite time [3]. Increasing system and task complexity also requires consideration of multiple objectives (e.g. performing a certain task while adhering to pose constraints). Yet, enforcing constraints on the planned motion can be difficult. Optimisation-based planners can struggle to incorporate constraints that cannot be expressed directly in joint space. Sampling-based planners, on the other hand, struggle to find solutions in scenarios where constraints render only a small volume of configuration space feasible or where narrow passages exist [6].

The advent of deep learning has shown that learning-based approaches can offer some relief in overcoming robotic planning and control challenges. While a considerable body of work examines the direct learning of control policies, [e.g 7], attempts have been made to apply deep learning to robotic path planning. Learnt heuristics and neural network collision detectors have been used as drop-in replacements to stages of traditional methods [e.g 8, 9, 10]. A number of works explore the use of structured latent spaces to effect planning and control [e.g 11, 12, 13, 14]. However, existing works typically require training for a particular task on carefully curated data. In contrast, applications of variational autoencoders (VAEs) in the space of affordance-learning [15] and quadruped locomotion [16] have highlighted the potential of viewing planning as run-time optimisation in pre-trained statistical models of state-space to achieve feasible spatial paths under environmental constraints.
Figure 1: A VAE is trained to produce a latent representation $z$ of the joint states and corresponding end-effector positions and an obstacle collision predictor learns the probability of collision. Once trained, gradients through the VAE decoder and collision predictor enable optimisation in the latent space to bring the decoded end-effector position closer to the target position. Performing this optimisation iteratively with a learning rate produces a series of latent values $\{z_t\}_{t=1}^T$ that describe a joint-space path to the target that satisfies the collision constraint.

Inspired by [15] and [16], in this work we explore an alternative, entirely data-driven approach to both joint-space planning and constraint satisfaction in a robot manipulation setting. In particular, our approach leverages iterative, gradient-based optimisation to produce a sequence of joint configurations by traversing the latent space of a VAE. Training data for this model is trivially obtained as it need not be in any way task-oriented but can come from random motor-babbling on a real platform, or simply sampling valid states in simulation. In addition, *performance predictors* operating on the latent space and potentially other observational data (for example, the positions of obstacles) are trained in a supervised fashion to output probabilities of certain performance-related metrics, such as whether the manipulator is in collision with an obstacle. These networks are frozen after training and are subsequently used in this gradient-based optimisation approach to planning through *activation maximisation* [17], which is the process of using backpropagation through network weights to find a permutation to the network inputs that would act to bring about a desired change in the network’s outputs.

Taking this view of path planning overcomes many of the obstacles that make robotic path planning a non-trivial task: (a) As our plans consist of states drawn from a deep generative model fit to a large dataset of feasible robot poses, and are thus approximately drawn from this data distribution, there is a very high likelihood that every state in the planned path is valid in terms of self-collisions and kinematic feasibility. (b) By modelling joint states and end-effector positions jointly, we avoid the need to explicitly calculate inverse or forward kinematics at any stage during planning, even when the goal configuration is given in $\mathbb{R}^3$ Cartesian space. (c) By leveraging activation maximisation (AM) via gradients through performance predictors, we can enforce arbitrarily complex, potentially non-differentiable constraints that would be hard to express in direct optimisation-based planners, and might be intractably restrictive for sampling-based planners. (d) By taking a pre-trained, data-driven approach to collision avoidance, we do not need any geometric analysis or accurate 3D models at planning time, nor indeed do we need to perform any kind of explicit collision checking, which is generally the main computational bottleneck in sampling-based planners [18].

In addition to the advantages in path planning that this method offers, we introduce an additional loss on the run-time AM optimisation process which encourages the planning process to remain in areas of high likelihood under the generative model. In our experiments we find that this contribution is critical in enabling successful planning that stays in feasible state-space.
2 RELATED WORK

Planning for Robot Manipulators Successful path planning for a robotic manipulator generally consists of producing a kinematic sequence of joint states through which the robot can actuate in order to move from a start to a goal configuration, while moving only through viable configurations. While goal positions may be specified in the same joint space as the plan, in a manipulator context it is more common for the goal position to be specified in \( \mathbb{R}^3 \) Cartesian end-effector space, or \( \mathbb{R}^6 \) if the SO(3) rotation group is included as well. Viable configurations are the intersection of feasible states for the robot - i.e. those that are within joint limits and do not result in self-collision - and collision-free states with respect to obstacles in the wider environment. The intersection of these defines the configuration space for the robot in a given environment.

As analytically describing the valid configuration space is generally intractable, sampling-based methods for planning provide the ability to quickly find connected paths through valid space, by checking for the validity of individual sampled nodes. Variants of the Probabilistic Roadmap (PRM) and Rapidly-exploring Random Tree (RRT) sampling-based algorithms are widely used [1, 2], and provably asymptotically optimal variants exist in PRM*, RRT* [3]. These methods suffer from a trade-off between runtime and optimality: while often relatively quick to find a feasible collision-free path, they tend to employ a second, slower, stage of path optimisation to shorten the path through the application of heuristics. In the presence of restrictive constraints, both sampling- and optimisation-based planners can be very slow to find an initial feasible path [6].

Optimisation-based planners start from an initial path or trajectory guess and then refine it until certain costs, such as path length, are minimised, and differentiable constraints satisfied. Techniques such as CHOMP [4] bridge the gap between planning and optimal control theory by enabling planning over path and dynamics. Stochastic Trajectory Optimization for Motion Planning (STOMP) is notable in the context of this work as it is able to produce plans while optimising for non-differentiable constraints [5], which the current work enables with gradients through trained performance predictors.

Planning with Deep Neural Networks A recent line of work has explored the use of deep neural networks to augment some or all components of conventional planning methods. Qureshi et al. [9, 10] train a pair of neural networks to embed point-cloud environment representations and perform single timestep planning. Iterative application of the planning network produces a path plan. Ichter and Pavone [8] learn an embedding space of observations, and use RRT in this space with a learnt collision checker to produce path plans, but need data to learn a forward dynamics model in order to roll out the plan.

Another family of learning-based approaches to planning learn embedding spaces from high dimensional data, and learn forward dynamics models that operate on this learnt latent space. Universal Planning Networks [11] learn deterministic representations of high-dimensional data such that update steps by gradient descent correspond to the unrolling of a learned forward model. The Embed-to-Control works [12, 19] employ variational inference in deep generative models in which latent-space dynamics is locally linear, a property that enables locally optimal control in these spaces. DVBFs [20] improve on these models by relaxing the assumption that the observation space is Markovian. PlaNet [21] uses a latent-space dynamics model for planning in model-based RL. However, planning in all these models tends to consist of rolling trajectories out in time, finding a promising trajectory, and executing the given actions. As such, these techniques tend to become intractable for longer time horizons, and cannot be thought of as path planning frameworks.

Learning Inverse Kinematics In this work, by learning a joint embedding of joint angles \( q \) and end-effector positions \( e \), we are able to optimise for achieving an end-effector target \( e_{\text{target}} \), while planning state sequences in joint space. Leveraging this learnt statistical model of kinematics means we do not need to solve inverse kinematics (IK) at any point. Prior work has sought to learn solutions to IK that can cope with its ill-posed one-to-many nature for redundant manipulators [e.g. 22, 23], and overcome the problems with analytic and numerical approaches [e.g. 24, 25, 26]. Ren et al. [23] train a generative adversarial network to generate joint angles from end-effector positions, with the discriminator acting on the concatenation of both the input position and generated joints. This method implicitly maximises \( p(q|e) \), but does not address the multimodality of the true \( p(q|e) \) IK solutions. Bocsi et al. [22] employed structured output learning to learn a generative model for the joint distribution between joint angles and end-effector positions. By modelling the joint instead of conditional distributions, i.e. \( p(q|e) \) rather than \( p(q|e) \), they can learn models that capture the multimodal nature of IK, as one set of IK solutions \( (e_1, q_1) \) can be learnt without compromising the
learning of another set \( (e_1, q_2) \). These works do not extend to path planning, but are relevant to this work in the way in which they use a learnt statistical model to capture the relationship between \( q \) and \( e \).

3 PATH PLANNING AS OPTIMISATION IN LATENT-SPACE

Our approach to path planning first learns a latent representation of the robot state by observing random (feasible) arm configurations. We then learn high-level performance predictors acting on this latent space as well as environment information to guide optimisation in latent space.

3.1 Problem Formulation

Suppose we have a dataset of joint angles and end-effector positions \( x = \{ (q_i, e_i) \}_{i=1}^m \). We use a VAE to learn a generative latent-variable model of \( x \). When sampled, we expect the generative model to produce data that conform to the forward kinematics (FK) relationship. While we do not leverage the FK information at runtime, we use it during training to evaluate the sample consistency of the generative model, i.e. how well the samples of joint angles and corresponding Cartesian end-effector positions match the actual system. We opt to encode \( (q_i, e_i) \) jointly as the information is readily available from routine robot operation and it avoids the ambiguity usually associated with mapping from the manipulator’s Cartesian workspace to a valid joint configuration, thereby simplifying the inference model. To solve path planning in this approach, we use AM to iteratively backpropagate position error relative to a reaching goal into the latent space [17]. Via the decoder, each location in latent space can be decoded into a robot configuration such that trajectories in latent space, when decoded, result in sequences of robot poses.

We posit, first, that this approach will produce valid paths from an initial end-effector position to the given target position. Although the application differs, our approach is directly inspired by the models in [15] and [16]. Our second hypothesis is that the accuracy of reaching operation will be correlated with the sample consistency of the model. That is, if the model demonstrates a closer coupling of joint angles and Cartesian end-effector position, as defined by the analytic FK relationship, then it will produce more accurate reaching solutions via AM. We will demonstrate how the approach can be extended to deal with reaching tasks while avoiding obstacles. One strength of this approach is the conceptual ease with which additional constraints can be added.

3.2 Learning a Latent Representation of Robot State

Our aim is to learn a generative model of \( x \). This can be accomplished with a variational autoencoder (VAE) [27, 28], which defines an encoder \( q_\phi (z \mid x) \) and decoder \( p_\theta (x \mid z) \), where \( z \) is a learned latent representation. To train the VAE, we would like to maximise the log-likelihood of the data \( \log p(x) \). However, the marginal log-likelihood \( \int p_\theta(x|z)p(z)dz \) is in general intractable. A common alternative therefore is to maximise the evidence lower bound (ELBO), where \( L_{\text{ELBO}} \leq p(x) \):

\[
L_{\text{ELBO}} = \underbrace{\mathbb{E}_{z \sim q_\phi(z|\mathbf{x})} \log p_\theta (\mathbf{x} \mid \mathbf{z})}_{\text{Reconstruction Accuracy}} - \underbrace{\mathcal{D}_{\text{KL}}[q_\phi(z \mid x) \mid \mid p(z)]}_{\text{KL Term}}
\]

(1)

To trade off between reconstruction accuracy and the KL term, a \( \beta \) hyperparameter is often added to the ELBO formulation [29]. Rather than setting this hyperparameter manually [29], we adopt an alternative GECO approach [30]. The GECO objective formulates the ELBO loss as a constrained optimisation problem, using a Lagrange multiplier \( \lambda \), such that

\[
L_{\text{GECO}} = -\mathcal{D}_{\text{KL}}[q_\phi(z \mid x) \mid \mid p(z)] + \lambda \underbrace{\mathbb{E}_{z \sim q_\phi(z|\mathbf{x})} \mathcal{C}(\mathbf{x}, \hat{\mathbf{x}})}_{\text{Reconstruction Error Constraint}}
\]

(2)

The Lagrangian optimises the KL divergence subject to \( \mathbb{E}_{z} \mathcal{C}(\mathbf{x}, \hat{\mathbf{x}}) \leq 0 \), for a given constraint function \( \mathcal{C} \). The constraint typically models an upper bound on a predefined reconstruction error (e.g. an \( L_2 \) loss):

\[
\mathcal{C}(\mathbf{x}, \hat{\mathbf{x}}) = \| \mathbf{x} - \hat{\mathbf{x}} \|_2 - \tau
\]

(3)

Although the GECO formulation still contains a hyperparameter, \( \tau \geq 0 \), this represents an interpretable quantity: an upper bound on the reconstruction error. In practice, this is easier to work with than tuning the \( \beta \) hyperparameter in the latent space, which is difficult to interpret. VAEs in our experiments are trained by optimising the GECO objective with the \( L_2 \) reconstruction loss.
3.3 Activation Maximisation for Path Planning Under a Prior Loss

Given a target position \( \mathbf{e}_{\text{target}} \), the aim is to produce a sequence of joint configurations \( (\mathbf{q}_0, \ldots, \mathbf{q}_T) \) that drive the robot’s end-effector from its initial position \( \mathbf{e}_0 \) to an end position \( \mathbf{e}_T \) within a distance tolerance \( d(\mathbf{e}_T, \mathbf{e}_{\text{target}}) < \gamma \). This can be achieved in the probabilistic model through the iterative use of AM [17].

Let the initial \( \mathbf{x}_0 \) be encoded such that the corresponding latent configuration \( \mathbf{z}_0 \) is drawn from the posterior. Decoding \( \mathbf{z}_0 \) then gives rise to \( \mathbf{x}_0 = \{ \hat{\mathbf{q}}_0, \hat{\mathbf{e}}_0 \} \). Let \( d(\mathbf{e}_0, \mathbf{e}_{\text{target}}) \) denote the Euclidean distance between \( \mathbf{e}_0 \) and \( \mathbf{e}_{\text{target}} \), then we can compute an \( L_2 \) loss that we backpropagate through the decoder \( p_\theta(\mathbf{e}, \mathbf{q} | \mathbf{z}) \). However, rather than update the network weights, we use AM to update the latent representation. In particular, given the AM objective, latent representations are updated iteratively in the following way, where \( \alpha_{\text{AM}} \) is the learning rate and \( \nabla L^{\text{AM}} \) is the gradient of the AM loss with respect to the input \( \mathbf{z} \):

\[
\mathbf{z}_{t+1} = \mathbf{z}_t - \alpha_{\text{AM}} \nabla L^{\text{AM}}, \quad L^{\text{AM}} = \| \hat{\mathbf{e}}, \mathbf{e}_{\text{target}} \|_2^2
\]

This produces a progression of latent representations \( (\mathbf{z}_1, \ldots, \mathbf{z}_T) \), which continues for a set number of \( T \) steps. Through the decoder, these latent representations can each be mapped to joint angles \( (\mathbf{q}_1, \ldots, \mathbf{q}_T) \). If the kinematics relationships represented by the decoder network are valid, and a sufficient number of steps \( T \) are taken, then we expect the final joint angle configuration \( \hat{\mathbf{q}}_T \) to correspond to a new end-effector position \( \mathbf{e}_T \) such that \( d(\mathbf{e}_T, \mathbf{e}_{\text{target}}) < \gamma \). Starting with the initial position, the sequence of decoded end-effector positions represents a spatial path \( (\mathbf{e}_0, \ldots, \mathbf{e}_T) \).

Without modification, AM may often drive the values \( \mathbf{z} \) into parts of the latent space that have not been seen during training. Decoding these latent representations can lead to poor \( (\mathbf{q}, \mathbf{e}) \) pairs that are inconsistent with the desired kinematics. To encourage the optimisation to traverse regions in which the model is well defined (i.e. to stay as close to the training distribution as possible) we introduce an additional loss term to the AM objective consisting of the likelihood of the current latent representation under its prior \( p(\mathbf{z}) \), such that

\[
L^{\text{AM}} = \| \hat{\mathbf{e}}, \mathbf{e}_{\text{target}} \|_2^2 + \lambda_{\text{prior}} \left( -\log p(\mathbf{z}) \right) + \lambda_{\text{obs}} \sum_t \left( -\log p_\theta(\mathbf{z}, \mathbf{o}_t) \right)
\]

This encourages the reconstructed joint configurations to remain valid. Again, \( \lambda_{\text{prior}} \) is tuned automatically during training using a GECO formulation, by selecting an upper bound on the prior loss.

3.4 Obstacle Avoidance via Performance Predictors

A key requirement for path planning is obstacle avoidance. In our framework this is effected by a binary classifier predicting whether the current arm configuration, as represented in latent space, is in collision with an obstacle. By back-propagating gradients forcing the collision response of this classifier to zero we effectively drive the robot away from obstacles. The classifier is trained using a binary cross entropy (BCE) loss while the VAE weights remain frozen.

When performing AM in the case of obstacle avoidance, we add an obstacle loss term from BCE to the AM loss in Equation (5).

\[
L^{\text{AM}} = \| \hat{\mathbf{e}}, \mathbf{e}_{\text{target}} \|_2^2 + \lambda_{\text{prior}} \left( -\log p(\mathbf{z}) \right) + \lambda_{\text{obs}} \sum_t \left( -\log (1 - p_\theta(\mathbf{z}, \mathbf{o}_t)) \right)
\]

where \( \lambda_{\text{prior}} \) and \( \lambda_{\text{obs}} \) are tuned jointly using GECO with multiple constraints. Avoidance of multiple obstacles can be achieved by repeatedly deploying the same classifier and adding the resulting gradients into the optimisation. The ease with which multiple constraints can be expressed and enforced is an explicit strength of this approach.

3.5 Model Selection Through Sample Consistency

While the downstream performance we seek from our models is better path planning, this is not continuously measurable during training. For VAE model selection and hyperparameter tuning,
Figure 2: We project the latent space down to 2D via PCA to visualise AM with and without prior loss (top) / with and without obstacle loss (bottom). The blue region is the encoding of the training distribution. The green and the purple curves are the robot trajectories from AM. The black dot is the latent representation of the target joint angles and coordinates. In the case of no prior loss, the encoding of the robot initial configuration lies in the trusted region, but drifts to its boundary as we perform gradient descent, which decodes to a meaningless output. In the case of no obstacle loss, the robot collides with an obstacle. The link in collision is shown in red.

we consider three metrics as predictors of path planning success: (a) the data reconstruction loss $\|\hat{e} - e\|_2 + \|\hat{q} - q\|_2$, (b) ELBO (Equation (1)) and (c) kinematic sample consistency, which we define as

$$\delta = \|\hat{e} - \text{FK}(\hat{q})\|_2$$

This sample consistency error $\delta$ is the Euclidean distance between the reconstructed end-effector position $\hat{e}$ and the true forward kinematics (FK) solution for the reconstructed joint angles $\hat{q}$.

This high sample consistency is a better predictor of a model’s downstream planning performance than the more traditional ELBO loss.

4 EXPERIMENTS

We evaluate our approach in the context of a set of robot reaching tasks using the 7-DoF Franka Emika Panda arm. Details of the model architecture and data collection process can be found in the Appendix.

4.1 Path Planning for Target Reaching

Before extending to path planning with additional constraints, we explore the ability of iterative AM as described in Section 3.3 to produce a path plan for goal reaching in open space. We sample 1,000 start and goal configurations for the robot, with an initial joint position $q_1$ and a goal $e_{\text{target}}$ in $\mathbb{R}^3$.

Results of our method are shown in Figure 3, where we quantify planning success rates at different distance thresholds. We find that the addition of the prior loss (Equation (5)) to the AM objective is instrumental in improving success rates, while when we optimise AM for reducing distance to goal with no additional constraint, we observe more frequent instabilities in the iterative AM procedure, which lead to infeasible state reconstructions $\hat{q}$ in the path plans. With the prior loss, over 90% of the planning scenes are solved to within a 5mm threshold of the goal.

4.2 Obstacle Avoidance

To evaluate the efficacy of our approach in finding feasible plans in the presence of obstacles we generate 1,000 scenarios for each of one to five obstacles. We compare our approach of latent-space path planning (LSPP) to five planners in widespread use: RRTConnect [32], CHOMP, LBKPIECE [33],
Figure 3: Left: target reaching success vs reaching distance threshold, evaluated on 1,000 scenarios. Grey lines are the 95% confidence interval of Wilson score [31]. Adding prior loss in AM objective function improves reaching success rate. Right: minimum distance between the end-effector and the target vs sample consistency error. Higher sample consistency leads to better target reaching.

Table 1: Comparison of performance of our latent-space path planning (LSPP) and baseline motion planning algorithms. They are run on a test dataset of 1,000 scenarios. The values are displayed with a 95% confidence interval (Wilson score [31] for planning success rate and standard deviation for planning time and path length).

<table>
<thead>
<tr>
<th>Planning success rate [%]</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
</tr>
</thead>
<tbody>
<tr>
<td>LSPP (ours)</td>
<td>85.8 ± 2.2</td>
<td>59.4 ± 3.0</td>
<td>38.2 ± 3.0</td>
<td>25.0 ± 2.7</td>
<td>15.7 ± 2.3</td>
</tr>
<tr>
<td>RRTConnect</td>
<td>84.9 ± 2.2</td>
<td>58.8 ± 3.1</td>
<td>47.7 ± 3.1</td>
<td><strong>34.5 ± 2.9</strong></td>
<td><strong>26.8 ± 2.7</strong></td>
</tr>
<tr>
<td>CHOMP</td>
<td>84.9 ± 2.2</td>
<td>58.3 ± 3.1</td>
<td>47.8 ± 3.1</td>
<td>34.3 ± 2.9</td>
<td>26.7 ± 2.7</td>
</tr>
<tr>
<td>LBKPIECE</td>
<td>82.9 ± 2.3</td>
<td>57.8 ± 3.1</td>
<td>49.1 ± 3.1</td>
<td>32.5 ± 2.9</td>
<td>25.3 ± 2.7</td>
</tr>
<tr>
<td>RRT*</td>
<td>85.0 ± 2.2</td>
<td>58.1 ± 3.1</td>
<td>47.7 ± 3.1</td>
<td>33.2 ± 2.9</td>
<td>25.9 ± 2.7</td>
</tr>
<tr>
<td>LazyPRM*</td>
<td>82.3 ± 2.4</td>
<td>57.5 ± 3.1</td>
<td>47.2 ± 3.1</td>
<td>33.2 ± 2.9</td>
<td>25.4 ± 2.7</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Planning time [ms]</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
</tr>
</thead>
<tbody>
<tr>
<td>LSPP (ours)</td>
<td>179.8 ± 85.1</td>
<td>185.5 ± 90.8</td>
<td>189.8 ± 91.5</td>
<td><strong>191.9 ± 92.0</strong></td>
<td><strong>201.0 ± 98.2</strong></td>
</tr>
<tr>
<td>RRTConnect</td>
<td><strong>128.5 ± 254.0</strong></td>
<td><strong>150.5 ± 330.0</strong></td>
<td><strong>180.9 ± 390.5</strong></td>
<td>195.9 ± 344.2</td>
<td>231.9 ± 252.8</td>
</tr>
<tr>
<td>CHOMP</td>
<td>128.3 ± 266.6</td>
<td>161.0 ± 389.0</td>
<td>188.9 ± 423.6</td>
<td>192.4 ± 197.9</td>
<td>232.7 ± 350.1</td>
</tr>
<tr>
<td>LBKPIECE</td>
<td>401.7 ± 400.1</td>
<td>437.0 ± 455.9</td>
<td>526.8 ± 601.4</td>
<td>539.1 ± 451.2</td>
<td>561.6 ± 330.5</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Path length</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
</tr>
</thead>
<tbody>
<tr>
<td>LSPP (ours)</td>
<td>1.52 ± 0.36</td>
<td>1.51 ± 0.34</td>
<td>1.47 ± 0.30</td>
<td><strong>1.50 ± 0.31</strong></td>
<td><strong>1.48 ± 0.27</strong></td>
</tr>
<tr>
<td>RRTConnect</td>
<td>2.33 ± 1.23</td>
<td>2.25 ± 1.05</td>
<td>2.24 ± 1.13</td>
<td>2.12 ± 1.05</td>
<td>2.15 ± 1.14</td>
</tr>
<tr>
<td>CHOMP</td>
<td>2.29 ± 1.14</td>
<td>2.26 ± 1.19</td>
<td>2.18 ± 1.05</td>
<td>2.14 ± 1.12</td>
<td>1.96 ± 0.91</td>
</tr>
<tr>
<td>LBKPIECE</td>
<td>2.27 ± 1.14</td>
<td>2.26 ± 1.25</td>
<td>2.16 ± 0.99</td>
<td>2.07 ± 1.04</td>
<td>1.94 ± 0.99</td>
</tr>
<tr>
<td>RRT*</td>
<td>1.53 ± 0.93</td>
<td><strong>1.50 ± 0.67</strong></td>
<td>1.48 ± 0.67</td>
<td><strong>1.50 ± 0.83</strong></td>
<td><strong>1.47 ± 0.57</strong></td>
</tr>
<tr>
<td>LazyPRM*</td>
<td>2.20 ± 1.11</td>
<td>2.18 ± 1.22</td>
<td>2.13 ± 1.07</td>
<td>2.03 ± 0.97</td>
<td>1.96 ± 0.87</td>
</tr>
</tbody>
</table>

For LSPP, a grid search is conducted on the GECO target (Equation (6)), GECO smoothing factor and GECO learning rate for the obstacle loss term to optimise for the overall success rate. Across all methods, a run is considered a success if the robot reaches the target within a distance threshold of 1cm and without colliding with obstacles.

RRT* [3] and LazyPRM* [34]. For all baselines we use the default parameters from their MoveIt OMPL and CHOMP library implementations [35, 36]. For RRT* and LazyPRM*, we keep the default planning time of 5 seconds. We only use CHOMP as a post-processor for OMPL with the default RRTConnect planner. Quantitative results are shown in Table 1.
In terms of planning success rate, LSPP performs commensurate to the baselines in the case of one and two obstacles, but suffers a performance drop when more obstacles are present. Our scenario generation process (detailed in the Appendix) does not ensure there exists a feasible solution to a particular scenario. The success rates are therefore only indicative of relative performance. However, RRTConnect and RRT* serve as useful calibration as they are probabilistically complete, ensuring a solution will be found if one exists, given sufficient runtime. There are a number of factors which influence LSPP performance. There exists an inherent tension due to the AM objective between reaching a goal and avoiding obstacles. This is, in effect, regulated by the GECO parameters. As LSPP is inherently a gradient-based optimisation method it is subject to local minima. In addition, the optimisation can be misguided either by a failure in the obstacle classifier or due to low sample inconsistency.

Overall LSPP’s average planning time is commensurate with that of RRTConnect and CHOMP whereas it significantly outperforms LBKPIECE. We note also that LSPP exhibits consistently lower variances in planning time than the baselines. In LSPP, each additional obstacle requires an extra forward and backward pass of the collision predictor, and thus planning time increases linearly with obstacles. However, in these experiments this remains a negligible effect on the overall LSPP time. RRT* and LazyPRM* both operate with a fixed time budget and have therefore been omitted from the planning time comparison.

The path length is normalised by dividing the actual length of the planned path by the Euclidean distance between the initial end-effector position and the target position to ensure a fairer comparison among different scenarios. RRT* is an asymptotically optimal algorithm, thus it is not surprising that it finds near optimal paths. Nevertheless, LSPP outperforms most of the other baselines.

Overall, it is encouraging to see that LSPP, an intuitive and data-driven formulation, is approaching the performance of established path planning algorithms.

5 CONCLUSION

We presented a novel approach to path planning for robot manipulation that learns a structured latent representation of the robot’s state space and uses constrained optimisation to produce joint space paths to reach end-effector goals. Our approach differs significantly from related work in that it performs path planning based on a generative model of robot state, which is trained in an entirely task-agnostic manner. In addition to the goal and obstacle losses, we introduce a novel constraint which maximises the likelihood of the latent variable being explored under its learned prior, thereby encouraging the model to stay near the training distribution of robot configurations. In doing so, we bypass the traditional computational challenges encountered by established planning methods while achieving commensurate performance in terms of reaching success, planning time and path length. Future directions include generalisation to scenarios with more complex obstacles, dynamic objects and tasks that involve interaction.
References


