Reaching Through Latent Space: From Joint Statistics to Path Planning in Manipulation

Chia-Man Hung, *Graduate Student Member, IEEE*, Shaohong Zhong, Walter Goodwin, Oiwi Parker Jones, Martin Engelcke, Ioannis Havoutis, *Member, IEEE*, and Ingmar Posner

***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 obstacle avoidance on a real 7-DoF robot arm.**

***Index Terms*—Constrained motion planning, representation learning, deep learning in grasping and manipulation, optimization and optimal control.**

I. 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]–[6]). 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. Traditional 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 [7].

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, attempts have been made to apply deep learning to robotic path planning (e.g. [8]). Learnt heuristics and neural network collision detectors have been used as drop-in replacements to stages of traditional methods (e.g. [9]–[11]). A number of works explore the use of structured latent spaces to effect planning and control (e.g. [12]–[15]). 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 [16] and quadruped locomotion [17]

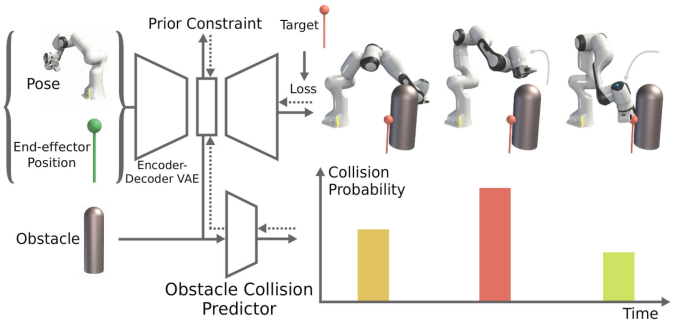

Fig. 1. A VAE is trained to produce a latent representation $\mathbf{z}$ of the joint states and corresponding end-effector positions and an obstacle collision predictor learns the probability of collision. Once trained, gradients through the VAE decoder and collision predictor enable optimisation in 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 $\{\mathbf{z}_i\}_{i=1}^{T}$ that describe a joint-space path to the target that satisfies the collision constraint.

Manuscript received September 9, 2021; accepted January 29, 2022. Date of publication February 23, 2022; date of current version March 15, 2022. This letter was recommended for publication by Associate Editor J. D. Hernandez and Editor S. J. Guy upon evaluation of the reviewers' comments. This work was supported in part by the UKRI/EPSRC Programme under Grant EP/V000748/1, in part by NIA under Grant EP/S002383/1, in part by RAIN under Grant EP/R026084/1, and in part by ORCA under Grant EP/R026173/1, the Clarendon Fund and Amazon Web Services as part of the Human-Machine Collaboration Programme. *(Corresponding author: Chia-Man Hung.)*

Shaohong Zhong, Oiwi Parker Jones, Martin Engelcke, and Ingmar Posner are with the Applied AI Lab (A2I), Oxford Robotics Institute (ORI), University of Oxford, Oxford OX2 6NN, U.K. (e-mail: shaohong@robots.ox.ac.uk; oiwi@robots.ox.ac.uk; engelcke@deepmind.com; hip@robots.ox.ac.uk).

Chia-Man Hung and Walter Goodwin are with the Applied AI Lab (A2I), U.K., and also with the Dynamic Robot Systems (DRS), Oxford Robotics Institute (ORI), University of Oxford, Oxford OX2 6NN, U.K. (e-mail: chiaman@robots.ox.ac.uk; walter@robots.ox.ac.uk).

Ioannis Havoutis is with the Dynamic Robot Systems (DRS), Oxford Robotics Institute (ORI), University of Oxford, Oxford OX2 6NN, U.K. (e-mail: ioannis@robots.ox.ac.uk).

This letter has supplementary downloadable material available at https://doi.org/10.1109/LRA.2022.3152697, provided by the authors.

Digital Object Identifier 10.1109/LRA.2022.3152697

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.

Inspired by [16] and [17], in this work we explore an alternative, entirely data-driven approach to both joint-space planning and constraint satisfaction in a robot manipulation setting (Fig. 1). 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* [18], 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 [19].

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

II. Related Work

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 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 exists 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 [7].

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 by enabling planning over path *and* dynamics. TrajOpt [20] differs from CHOMP in the numerical optimisation method used and the method of collision checking. The Gaussian Process Motion Planner [21] leverages Gaussian process models to represent trajectories and updates them through interpolation. Stochastic Trajectory Optimization for Motion Planning (STOMP) [5] is notable in this context as it is able to produce plans while optimising for *non-differentiable* constraints, which our work enables with gradients through trained *performance predictors*.

Planning with Deep Neural Networks: A recent line of work has explored using deep networks to augment some or all components of conventional planning methods. Qureshi *et al.* [10], [11] 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 [9] 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* [12] 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 [13], [22] 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. *DVBF*s [23] improve on these models by relaxing the assumption that the observation space is Markovian. *PlaNet* [24] uses a latent-space dynamics model for planning in model-based RL. However, planning in all these models tends to consist of rolling out trajectories 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.

A different approach is that of encoding movement primitives under the learning from demonstrations framework. *Conditional Neural Movement Primitives* [25] extracts prior knowledge from

demonstrations and infers distributions over trajectories conditioned on the current observation. Our approach differs in that we do not encode trajectories directly, but rather learn a probabilistic model of robot states and generate trajectories as we optimise in latent space.

Learning Inverse Kinematics: In this work, by learning a joint embedding of joint angles $\boldsymbol{q}$ and end-effector positions $\boldsymbol{e}$, we are able to optimise for achieving an end-effector target $\boldsymbol{e}_{\text{target}}$, while planning state sequences in *joint* space. Note that we do not care about the orientation in which the goal is reached, therefore end-effector orientation is omitted in the formulation of $\boldsymbol{e}$. Learning the 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 [26], [27], and to overcome the problems with analytic and numerical approaches [28]–[30]. Ren *et al.* [27] 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 $p(\boldsymbol{q}|\boldsymbol{e})$, but does not address the multimodality of the true $p(\boldsymbol{q}|\boldsymbol{e})$ IK solutions. Bocsi *et al.* [26] employed structured output learning to learn a generative model for the joint distribution of joint angles and end-effector positions. By modelling the joint instead of conditional distributions, i.e. $p(\boldsymbol{q}, \boldsymbol{e})$ rather than $p(\boldsymbol{q}|\boldsymbol{e})$, their model can capture the multimodal nature of IK, as one set of IK solutions $(\boldsymbol{e}_1, \boldsymbol{q}_1)$ can be learnt without compromising the learning of another set $(\boldsymbol{e}_1, \boldsymbol{q}_2)$. These works are relevant in the way in which they use a learnt statistical model to capture the relationship between $\boldsymbol{q}$ and $\boldsymbol{e}$. However, we differ from prior work in that, although we follow the generative approach, we do not try to find the best $\boldsymbol{q}$ that maximises $p(\boldsymbol{q}, \boldsymbol{e})$ as is done in [26], but instead plan in the latent space that decodes to valid joint configurations, producing smooth trajectories.

While our work is partly inspired by [16] and [17] in its approach, it significantly extends this prior work both in terms of method and application domain. In particular, in exploring this approach in a manipulation context we rely solely on training poses to structure the latent space. This is in contrast to [17], where, in a quadruped locomotion context, structure is induced via especially designed stance labels. Like [16], who first proposed the use of AM for constrained optimisation in a structured latent space in the context of affordance learning, we consider environmental constraints. However, our agent operates in a significantly more complex configuration space to achieve real-world reaching and obstacle avoidance. In addition, we introduce an additional loss term that encourages the model to traverse regions of high likelihood under the learned prior over the latent variables (i.e. to stay close to the training distribution) during planning. We demonstrate that this novel loss term increases efficacy by a large margin, effectively encouraging kinematic feasibility of the plans produced.

III. 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.

A. Problem Formulation

Suppose we have a dataset of joint angles and end-effector positions $\mathbf{x} = \{(\boldsymbol{q}_i, \boldsymbol{e}_i)\}_{i=1}^{m}$. We use a VAE to learn a generative latent-variable model of $\mathbf{x}$. 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 $(\boldsymbol{q}_i, \boldsymbol{e}_i)$ 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 task. To solve path planning in this approach, we use AM to iteratively backpropagate position error relative to a reaching goal into the latent space [18]. In addition, we exploit the probabilistic nature of our model by encouraging solutions to traverse regions of latent space of high likelihood under prior belief via a prior loss. 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. 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.

B. Learning a Latent Representation of Robot State

Our aim is to learn a generative model of $\mathbf{x}$. This can be accomplished with a variational autoencoder (VAE) [31], [32], which defines an encoder $q_\phi(\mathbf{z} \mid \mathbf{x})$ and decoder $p_\theta(\mathbf{x} \mid \mathbf{z})$, where $\mathbf{z}$ is a learned latent representation. To train the VAE, we would like to maximise the evidence, $p_\theta(\mathbf{x}) = \int p_\theta(\mathbf{x} \mid \mathbf{z}) p_\theta(\mathbf{z}) d\mathbf{z}$, which is generally intractable. A common alternative therefore is to maximise the evidence lower bound (ELBO), where $\mathcal{L}^{\text{ELBO}} \leq \log p(\mathbf{x})$:

$$\mathcal{L}^{\text{ELBO}} = \underbrace{\mathbb{E}_{\mathbf{z}\sim q_\phi(\mathbf{z}|\mathbf{x})} \log p_\theta(\mathbf{x} \mid \mathbf{z})}_{\text{Reconstruction Accuracy}} - \underbrace{D_{\text{KL}}\left[q_\phi(\mathbf{z} \mid \mathbf{x}) \mid\mid p(\mathbf{z})\right]}_{\text{KL Term}} \tag{1}$$

To trade off between reconstruction accuracy and the KL term, a β hyperparameter is often added to the ELBO formulation [33]. Rather than setting this hyperparameter manually [33], we adopt an alternative, dynamic GECO approach [34]. The GECO objective formulates the ELBO loss as a constrained optimisation problem, using a Lagrange multiplier λ, such that

$$\mathcal{L}^{\text{GECO}} = \underbrace{-D_{\text{KL}}[q_\phi(\mathbf{z} \mid \mathbf{x}) \mid\mid p(\mathbf{z})]}_{\text{KL Term}} + \lambda \underbrace{\mathbb{E}_{\mathbf{z}\sim q_\phi(\mathbf{z}|\mathbf{x})}\left[\mathcal{C}\left(\mathbf{x}, \hat{\mathbf{x}}\right)\right]}_{\text{Reconstruction Error Constraint}}, \tag{2}$$

where $\hat{\mathbf{x}}$ is the reconstruction of $\mathbf{x}$ through the VAE. The reader is referred to [34] for implementation details on how λ is updated. The Lagrangian optimises the KL divergence

subject to $\mathbb{E}_{\mathbf{z}}[\mathcal{C}(\mathbf{x}, \hat{\mathbf{x}})] \leq 0$, 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):

$$\mathcal{C}\left(\mathbf{x}, \hat{\mathbf{x}}\right) = \|\mathbf{x} - \hat{\mathbf{x}}\|_2 - \tau \tag{3}$$

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 β 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.

C. Activation Maximisation for Path Planning Under a Prior Loss

Given a target position $\boldsymbol{e}_{\text{target}}$, the aim is to produce a sequence of joint configurations $(\boldsymbol{q}_0, \ldots, \boldsymbol{q}_T)$ that drive the robot's end-effector from its initial position $\boldsymbol{e}_0$ to an end position $\boldsymbol{e}_T$ within a distance tolerance $\|\boldsymbol{e}_T, \boldsymbol{e}_{\text{target}}\|_2 < \gamma$. This can be achieved in the probabilistic model through the iterative use of AM [18].

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 $\hat{\mathbf{x}}_0 = \{\hat{\boldsymbol{q}}_0, \hat{\boldsymbol{e}}_0\}$. More generally, $\hat{\mathbf{x}} = \{\hat{\boldsymbol{q}}, \hat{\boldsymbol{e}}\}$. Let $\|\hat{\boldsymbol{e}}_0, \boldsymbol{e}_{\text{target}}\|_2$ denote the Euclidean distance between $\hat{\boldsymbol{e}}_0$ and $\boldsymbol{e}_{\text{target}}$, then we can compute an L_2 loss that we backpropagate through the decoder $p_\theta(\boldsymbol{e}, \boldsymbol{q} \mid \mathbf{z})$. However, rather than update the network weights, we use AM to update the latent vector. In particular, given the AM objective, latent representations are updated iteratively in the following way, where α_{AM} is the learning rate and $\nabla\mathcal{L}^{\text{AM}}$ is the gradient of the AM loss with respect to the input $\mathbf{z}$:

$$\mathbf{z}_{t+1} = \mathbf{z}_t - \alpha_{\text{AM}} \nabla \mathcal{L}^{\text{AM}}, \qquad \mathcal{L}^{\text{AM}} = \underbrace{\|\hat{\boldsymbol{e}}, \boldsymbol{e}_{\text{target}}\|_2}_{\text{Target Loss}} \tag{4}$$

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 be mapped to joint configurations $(\boldsymbol{q}_1, \ldots, \boldsymbol{q}_T)$. 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 $\boldsymbol{q}_T$ to correspond to a new end-effector position $\boldsymbol{e}_T$ such that $\|\boldsymbol{e}_T, \boldsymbol{e}_{\text{target}}\|_2 < \gamma$. Starting with the initial position, the sequence of decoded end-effector positions represents a spatial path $(\boldsymbol{e}_0, \ldots, \boldsymbol{e}_T)$.

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 $(\boldsymbol{q}, \boldsymbol{e})$ 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

$$\mathcal{L}^{\text{AM}} = \underbrace{\|\hat{\boldsymbol{e}}, \boldsymbol{e}_{\text{target}}\|_2}_{\text{Target Loss}} + \lambda_{\text{prior}} \underbrace{(-\log p(\mathbf{z}))}_{\text{Prior Loss}} \tag{5}$$

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

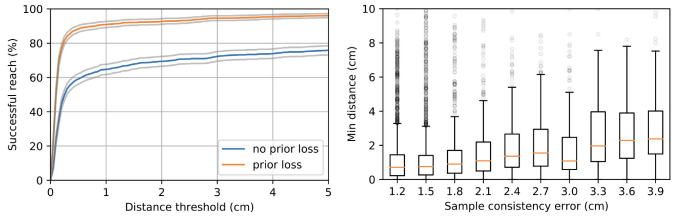

Fig. 2. Left: Target reaching success vs reaching distance threshold, evaluated on 1,000 scenarios. Grey lines are the 95% confidence interval of Wilson score [36]. Adding prior loss in AM objective function improves reaching success rate. Right: Minimum distance between the end-effector and the target vs sample consistency error. Lower sample consistency error leads to better target reaching.

D. 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 Eq. 5.

$$\mathcal{L}^{AM} = \underbrace{\|\hat{\boldsymbol{e}}, \boldsymbol{e}_{\text{target}}\|_2}_{\text{Target Loss}} + \lambda_{\text{prior}} \underbrace{(-\log p(\mathbf{z}))}_{\text{Prior loss}} + \lambda_{\text{obs}} \underbrace{\sum_i (-\log(1 - p_\vartheta(\mathbf{z}, \mathbf{o}_i)))}_{\text{Obstacle loss}}, \tag{6}$$

where λ_{prior} and λ_{obs} are tuned jointly using GECO with mutliple 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.

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

$$\delta = \|\hat{e} - \text{FK}(\hat{\boldsymbol{q}})\|_2 \tag{7}$$

This *sample consistency error* δ is the Euclidean distance between the reconstructed end-effector position $\hat{\boldsymbol{e}}$ and the true forward kinematics (FK) solution for the reconstructed joint angles $\hat{\boldsymbol{q}}$. We find that high sample consistency is a better predictor of a model's downstream planning performance than the more traditional ELBO loss alone (Fig. 2 right).

IV. Implementation Details

This section provides details on model architecture, model training and planning, using a 7-Dof Emika Franka Panda arm.

TABLE I
THE ARCHITECTURE FOR THE ENCODER, THE DECODER, AND THE OBSTACLE COLLISION CLASSIFIER. THE VAE ENCODER TAKES INPUT $\{\boldsymbol{q}, \boldsymbol{e}\}$, WHERE $dim(\boldsymbol{q}) = 7$, $dim(\boldsymbol{e}) = 3$, AND OUTPUTS $\boldsymbol{\mu}, \boldsymbol{\sigma}$, WHERE $dim(\boldsymbol{\mu}) = dim(\boldsymbol{\sigma}) = 7$. THE VAE DECODER TAKES INPUT $\mathbf{z}$ WHERE $dim(\mathbf{z}) = 7$, AND OUTPUTS RECONSTRUCTION WHICH IS OF THE SAME DIMENSION AS THE INPUT. THE COLLISION CLASSIFIER TAKES INPUT $\{\mathbf{z}, \mathbf{o}\}$, WHERE $dim(\mathbf{z}) = 7$, $dim(\mathbf{o}) = 4$

	VAE Encoder	VAE Decoder	Collision Classifier
Input Dimension	10	7	11
Output Dimension	7×2	10	1
No. of Hidden Layers	4	4	4
Units per Hidden Layer	2048	2048	2048
Hidden Layer Activation	ELU	ELU	ELU

A. Architecture Details

The VAE architecture comprises of an encoder and a decoder. The encoder takes as input $\mathbf{x} = \{\boldsymbol{q}, \boldsymbol{e}\}$ and outputs the mean $\boldsymbol{\mu}$ and variance $\boldsymbol{\sigma}$ of the posterior distribution $q_\phi(\mathbf{z} \mid \mathbf{x})$. The latent encoding $\mathbf{z}$ is then obtained using the reparameterisation trick [31]. A multivariate isotropic Gaussian prior is imposed on the latent space. The decoder takes as input the latent sample $\mathbf{z}$ and outputs the reconstruction $\hat{\mathbf{x}} = \{\hat{\boldsymbol{q}}, \hat{\boldsymbol{e}}\}$. The obstacle collision classifier takes $\{\mathbf{z}, \mathbf{o} = \{x, y, h, r\}\}$ (xy coordinates, height, radius of the cylinder) as input and has a single output logit, which when passed through a sigmoid function gives the predicted probability of collision. The encoder, decoder and obstacle collision classifier each contains four fully connected hidden layers of 2048 units, but differ in input and output layers, as shown in Table I.

B. Training Data Generation

In this evaluation, we consider cylindrical objects,[1] which are easily represented in state space as tuples $\{\boldsymbol{q}, \boldsymbol{e}, \mathbf{o}, \mathbf{c}\}$, where $\boldsymbol{q} = (\theta_1, \ldots, \theta_7)$ represents the robot joint configurations; $\boldsymbol{e} = (e_1, e_2, e_3)$ the end-effector coordinates; $\mathbf{o} = (x, y, h, r)$ the obstacle coordinates, height and radius; and $\mathbf{c} \in \{0, 1\}$ the binary collision label. Joint configurations $\boldsymbol{q}$ are sampled uniformly within the joint limits. We take the modified Denavit–Hartenberg parameters in the Panda arm documentation to characterise the forward kinematics relationship, $\boldsymbol{e} = \text{FK}(\boldsymbol{q})$. To generate the position of the obstacles, for each obstacle, we sample a distance to origin L, an angle θ_{obs} in $[0, 2\pi)$ uniformly and set $x = L\ \cos(\theta_{obs})$, $y = L\ \sin(\theta_{obs})$. MoveIt's planning scene interface is used to check whether the arm is in self-collision or in collision with the table; joint configurations that are in such collisions are discarded. We also use MoveIt's planning scene interface to label collision with obstacles in the training data. The dataset contains an equal number of samples in and not in collision with the obstacles. In total, the dataset contains 100 k data points, of which 80 k are used for training and 20 k for validation.

C. Obstacle Scenario Generation

In our experiments, scenarios are generated by sampling a given number of obstacles and two sets of joint angles – one for the initial robot configuration and another for the target position. The joint angle samples for the target are only used to compute the target position through the FK model of the Panda arm that we characterised and are not known to the planner. The obstacles and the target are generated while ensuring that there is at least a feasible set of joint angles reaching the target without collision with the obstacles. The first obstacle is sampled between the initial end-effector position and the target position. Subsequent obstacles are either sampled randomly or sampled between the initial end-effector position and the target position, with a probability of 50% each. The model is evaluated on scenarios in which it would collide with the obstacles if the obstacle loss term was not added to the total loss in the AM objective function.

[1]Our approach readily extends to other obstacle geometries, extending to observation space.

Algorithm 1: Activation Maximisation for Obstacle Avoidance.

initialise path buffer X;
initialise λ_{prior} and λ_{obs};
infer latent representation of initial configuration $\mathbf{z}_0 \sim q_\phi(\mathbf{z} \mid \mathbf{x} = \mathbf{x}_0)$;
for each time step t = 0,1,...,T **do**
  decode latent encoding to state space $\hat{\mathbf{x}}_t \sim p_\theta(\mathbf{x} \mid \mathbf{z} = \mathbf{z}_t)$;
  save states to path $X_t = \hat{\mathbf{x}}_t$;
  if $d(\boldsymbol{e}_t, \boldsymbol{e}_{target}) < \gamma$ **then**
    break;
  end
  compute loss terms $\|\hat{\boldsymbol{e}}, \boldsymbol{e}_{\text{target}}\|_2$, $-\log p(\mathbf{z})$, and $\sum_i(-\log(1 - p_\vartheta(\mathbf{z}, \mathbf{o}_i)))$;
  $\lambda^{t+1}_{\text{prior}}$ = Update ($\lambda^{t}_{\text{prior}}$);
  $\lambda^{t+1}_{\text{obs}}$ = Update (λ^{t}_{obs}) ;
  compute $\mathcal{L}_t^{AM} = \|\hat{\boldsymbol{e}}, \boldsymbol{e}_{\text{target}}\|_2 + \lambda^{t+1}_{\text{prior}}(-\log p(\mathbf{z})) + \lambda^{t+1}_{\text{obs}} \sum_i(-\log(1 - p_\vartheta(\mathbf{z}, \mathbf{o}_i)))$;
  update $\mathbf{z}_{t+1} = \mathbf{z}_t - \alpha_{\text{AM}} \nabla \mathcal{L}_t^{\text{AM}}$;
end

D. Training Details

The input values to the VAE and the collision classifier are standardised, and the output values de-standardised, according to the mean and the standard deviation of the training data. The model is trained using a batch size of 256 for 16,000 epochs using the Adam optimiser [35]. To select hyperparameters, a grid search is run on the following values: number of hidden layers, units per layer, latent dimension, GECO reconstruction target τ (Eq. 3), VAE learning rate, and GECO learning rate.

E. Planning Details

Planning is achieved by applying activation maximisation in the latent space, as outlined in Algorithm 1.

V. RESULTS

We evaluate our approach in the context of a set of robot reaching tasks, described below, using a simulated Panda arm. We further demonstrate that the approach can be deployed on a physical Panda arm.

A. Path Planning for Target Reaching

Before extending to path planning with additional constraints, we explore the ability of iterative AM as described in Section III-C to produce a path plan for goal reaching in free space. We sample 1,000 start and goal configurations for the robot, with

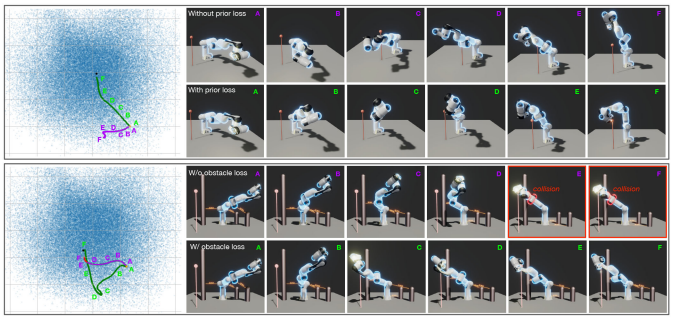

Fig. 3. We project the latent space down to 2D via PCA to visualise AM with and without prior loss (top)/ with and without obstacle loss (bottom). The blue region is the encoding of the training distribution. The green and the purple curves are the robot trajectories from AM. The black dot is the latent 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.

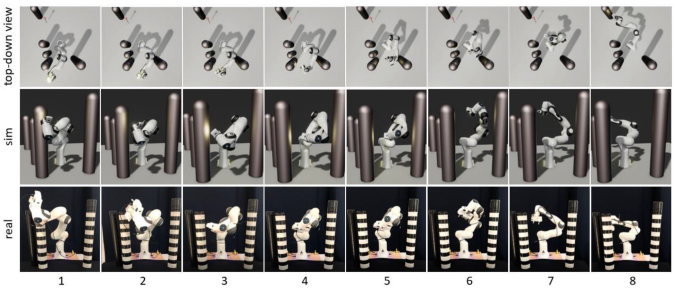

Fig. 4. Real-world experiments: a rollout of a trajectory using LSPP. Our latent-space approach operates in state space and therefore trivially transfers to the real world.

an initial joint position $\boldsymbol{q}_1$ and a goal $\boldsymbol{e}_{\text{target}}$ in $\mathbb{R}^3$. Results of our method are shown in Fig. 2 (left), where we quantify planning success rates at different distance thresholds. We find that the addition of the *prior loss* (Eq. 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 infeasible state reconstructions $\hat{\boldsymbol{q}}$ in the path plans (Fig. 3 top). With the prior loss, over 90% of the planning scenes are solved to within a 5 mm threshold of the goal.

B. Obstacle Avoidance

To demonstrate the effect of adding obstacle loss to accomplish obstacle avoidance, we show qualitative results both in simulation and in the real world in Fig. 3 (bottom) and Fig. 4.

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 cylindrical obstacles. We compare our approach of latent-space path planning (LSPP) to eight planners in widespread use: Potential Field [37], [38], RRTConnect [39], LBKPIECE [40], RRT* [3], LazyPRM* [41], FMT* [42], BIT* [43] and CHOMP [4]. The artificial potential field baseline is a classical local collision avoidance method using Jacobian pseudo-inverse to reduce the error in end-effector position while avoiding obstacles through the use of virtual repulsive forces. It is adapted from [38], but instead of using the depth-space concept to estimate the distances between the robot and the obstacles, it has direct access to the obstacle configurations to generate the repulsive vectors. A grid search is conducted on the hyperparameters (7 in [38]) to optimise for the overall success rate. For all other baselines we use the default parameters from their MoveIt OMPL and CHOMP library implementations [44], [45]. For RRT, LazyPRM* and BIT*, we keep the default planning time of 5 seconds. CHOMP uses a linear initialisation from start to goal position and optimises locally, such that it provides a fair comparison for local planners. Quantitative results are shown in Table II.

For LSPP, a grid search is conducted on the GECO target (Eq. 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 1 cm and without colliding with obstacles.

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. This performance drop is expected and can also be observed in CHOMP, another optimisation-based planner. Our scenario generation process does not ensure there exists a feasible solution to a particular scenario. The success rates are therefore only indicative of relative performance. However, RRTConnect, RRT*, FMT*and BIT* 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. Empirically, this happens more often as the number of obstacles increases, but could potentially be handled by adding a stochastic recovery strategy or a post processor. In addition, the optimisation can be misguided either by a failure in the obstacle classifier or due to low sample consistency.

Overall LSPP's average planning time is commensurate with that of RRTConnect whereas it significantly outperforms Potential Field, LBKPIECE, CHOMP and FMT*. 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.

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. It should be noted that the cost functions in OMPL minimise joint space path length, and a shortest path in joint space does not necessarily translate into a shortest path in Cartesian space. 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.

The artificial potential field baseline is widely used due to its simplicity and serves as a useful comparison for local collision avoidance methods. It is in spirit most similar to LSPP, subject to local minima, and neither of them has theoretical guarantees. However, it is not directly comparable as it assumes access to the FK relationship to compute the Jacobian while ours only relies on it for data collection and model selection, which could be avoided if we have a separate sensor for corresponding end-effector positions and if we choose a different model selection

TABLE II
COMPARISON OF PERFORMANCE OF OUR LATENT-SPACE PATH PLANNING (LSPP) AND BASELINE MOTION PLANNING ALGORITHMS. FOR EACH NUMBER OF OBSTACLES, THE EXPERIMENTS ARE RUN ON A TEST DATASET OF 1,000 SCENARIOS. THE VALUES ARE DISPLAYED WITH A 95% CONFIDENCE INTERVAL (WILSON SCORE [36] FOR PLANNING SUCCESS RATE AND STANDARD DEVIATION FOR PLANNING TIME AND PATH LENGTH). THE PLANNING TIME OF RRT*, LAZYPRM* AND BIT* ARE OMITTED SINCE THEY OPERATE WITH A FIXED TIME BUDGET OF 5 SECONDS

Planning success rate [%]

#obstacles	1	2	3	4	5
LSPP (ours)	**85.8 ± 2.2**	**59.4 ± 3.0**	38.2 ± 3.0	25.0 ± 2.7	15.7 ± 2.3
Potential Field	34.2 ± 2.9	20.5 ± 2.3	15.7 ± 2.3	11.4 ± 2.0	5.3 ± 1.4
RRTConnect	84.9 ± 2.2	58.8 ± 3.1	47.7 ± 3.1	34.5 ± 2.9	26.8 ± 2.7
LBKPIECE	82.9 ± 2.3	57.8 ± 3.1	**49.1 ± 3.1**	32.5 ± 2.9	25.3 ± 2.7
RRT*	85.0 ± 2.2	58.1 ± 3.1	47.7 ± 3.1	33.2 ± 2.9	25.9 ± 2.7
LazyPRM*	82.3 ± 2.4	57.5 ± 3.1	47.2 ± 3.1	33.2 ± 2.9	25.4 ± 2.7
FMT*	66.5 ± 2.9	52.7 ± 3.1	37.2 ± 3.0	29.8 ± 2.8	15.7 ± 2.3
BIT*	85.7 ± 2.2	58.0 ± 3.1	48.3 ± 3.1	**34.8 ± 3.0**	**26.9 ± 2.7**
CHOMP	80.0 ± 2.5	56.1 ± 3.1	37.0 ± 3.0	25.1 ± 2.7	16.2 ± 2.3

Planning time [ms]

#obstacles	1	2	3	4	5
LSPP (ours)	179.8 ± 85.1	185.5 ± 90.8	189.8 ± 91.5	**191.9 ± 92.0**	**201.0 ± 98.2**
Potential Field	1973.5 ± 296.4	2048.2 ± 313.6	2104.1 ± 320.6	2125.8 ± 327.2	2148.4 ± 319.7
RRTConnect	**128.3 ± 254.0**	**150.5 ± 330.0**	**180.9 ± 390.3**	195.9 ± 344.2	231.9 ± 252.8
LBKPIECE	401.7 ± 400.1	437.0 ± 455.9	526.8 ± 601.4	539.1 ± 451.2	561.6 ± 330.5
FMT*	877.8 ± 215.3	887.8 ± 199.4	872.6 ± 241.0	807.4 ± 206.0	820.9 ± 225.8
CHOMP	526.2 ± 308.7	628.3 ± 286.5	745.2 ± 316.1	792.4 ± 280.7	873.4 ± 412.5

Path length

#obstacles	1	2	3	4	5
LSPP (ours)	**1.52 ± 0.36**	1.51 ± 0.34	**1.47 ± 0.30**	**1.50 ± 0.31**	1.48 ± 0.27
Potential Field	1.54 ± 0.44	1.57 ± 0.35	1.54 ± 0.37	1.53 ± 0.36	1.53 ± 0.37
RRTConnect	2.33 ± 1.23	2.25 ± 1.05	2.24 ± 1.13	2.12 ± 1.05	2.15 ± 1.14
LBKPIECE	2.27 ± 1.14	2.26 ± 1.25	2.16 ± 0.99	2.07 ± 1.04	1.94 ± 0.99
RRT*	1.53 ± 0.93	**1.50 ± 0.67**	1.48 ± 0.67	**1.50 ± 0.83**	**1.47 ± 0.57**
LazyPRM*	2.20 ± 1.11	2.18 ± 1.22	2.13 ± 1.07	2.03 ± 0.97	1.96 ± 0.87
FMT*	2.30 ± 1.07	2.01 ± 0.84	2.04 ± 0.66	1.94 ± 0.69	1.91 ± 0.50
BIT*	2.13 ± 1.18	1.94 ± 0.73	1.87 ± 0.56	2.06 ± 0.74	1.98 ± 0.77
CHOMP	2.28 ± 1.23	2.27 ± 1.20	2.25 ± 1.15	2.16 ± 1.12	1.98 ± 0.93

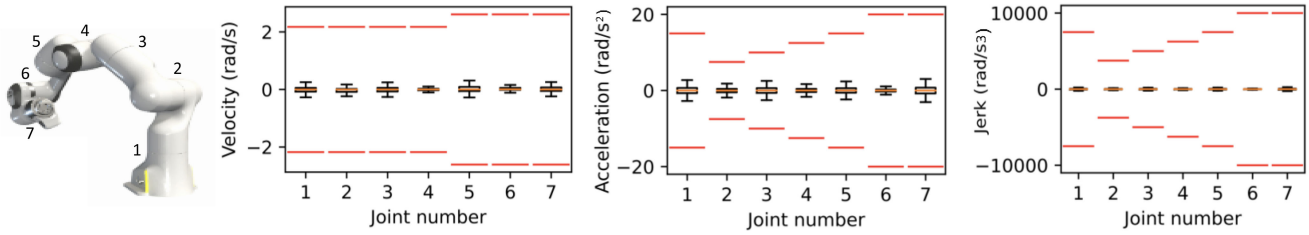

Fig. 5. Dynamic feasibility of motion plans for Panda arm over 1,000 trajectories. No LSPP motion plans violate the joint limits, indicated by the red segments. Left: angular velocity. Middle: angular acceleration. Right: angular jerk.

criterion. In terms of performance, it only achieves around 72% success rate even without any obstacles as it struggles at joint limits. Contrary to global planners, each action can be executed after each update is computed. Thus, it may appear to be surprisingly slow, while in reality it achieves real time performance.

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

C. Dynamic Feasibility

To show that the plans are dynamically feasible, we present an analysis in Fig. 5. The generated motion plans, when executed with a constant control frequency of 50 Hz, demand a relatively small angular velocity, angular acceleration and angular jerk. These are all well below the maximum joint limits for the Panda arm, shown as red segments in the figure. This demonstrates that we can generate feasible state space motion plans by decoding from the latent trajectory we obtain from gradient-based optimisation. Additionally, we can potentially further improve the smoothness by adjusting the learning rate of the Adam optimiser during the optimisation.

VI. CONCLUSION

We present 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 a largely 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. Despite the lack of theoretical guarantees, it is a practical mechanism for path planning. Future directions include algorithmic improvement to handle local minima, generalisation to scenarios with more complex obstacles and dynamic objects, and tasks that involve interaction.

ACKNOWLEDGMENT

The authors would like to thank the University of Oxford for providing Advanced Research Computing (ARC) facility in carrying out this work (http://dx.doi.org/10.5281/zenodo.22558) and the use of Hartree Centre resources. They thank Jonathan Gammell for insightful feedback and discussions, and Rowan Border for helping with setting up BIT* and interfacing between OMPL and MoveIt. They also thank Yizhe Wu for recording real-world experiments, and Jack Collins for proofreading their work.

REFERENCES

[1] S. M. La valle, "Rapidly-exploring random trees: A new tool for path planning," Tech. Rep., Computer Science Dept., Iowa State University, 1998.

[2] L. E. Kavraki, P. Svestka, J. Latombe, and M. H. Overmars, "Probabilistic roadmaps for path planning in high-dimensional configuration spaces," *IEEE Trans. Robot. Automat.*, vol. 12, no. 4, pp. 566–580, Aug. 1996.

[3] S. Karaman and E. Frazzoli, "Sampling-based algorithms for optimal motion planning," *Int. J. Robot. Res.*, vol. 30, no. 7, pp. 846–894, 2011.

[4] N. Ratliff, M. Zucker, J. A. Bagnell, and S. Srinivasa, "CHOMP: Gradient optimization techniques for efficient motion planning," in *Proc. IEEE Int. Conf. Robot. Automat.*, 2009, pp. 489–494.

[5] M. Kalakrishnan, S. Chitta, E. Theodorou, P. Pastor, and S. Schaal, "STOMP: Stochastic trajectory optimization for motion planning," in *Proc. IEEE Int. Conf. Robot. Automat.*, May 2011, pp. 4569–4574. [Online]. Available: https://ieeexplore.ieee.org/document/5980280/

[6] N. D. Ratliff, J. Issac, D. Kappler, S. Birchfield, and D. Fox, "Riemannian motion policies," 2018, *arXiv:1801.02854*.

[7] D. Berenson, S. S. Srinivasa, D. Ferguson, and J. J. Kuffner, "Manipulation planning on constraint manifolds," *Proc. IEEE Int. Conf. Robot. Automat.*, vol. 5, no. 4, pp. 625–632, May 2009. [Online]. Available: https://ieeexplore.ieee.org/document/5152399/

[8] S. Levine, P. Pastor, A. Krizhevsky, J. Ibarz, and D. Quillen, "Learning hand-eye coordination for robotic grasping with deep learning and large-scale data collection," *Int. J. Robot. Res.*, vol. 37, no. 4-5, pp. 421–436, 2018.

[9] B. Ichter and M. Pavone, "Robot motion planning in learned latent spaces," *IEEE Robot. Automat. Lett.*, vol. 4, no. 3, pp. 2407–2414, Jul. 2019. [Online]. Available: https://ieeexplore.ieee.org/document/8653875/

[10] A. H. Qureshi, A. Simeonov, M. J. Bency, and M. C. Yip, "Motion planning networks," in *Proc. IEEE Int. Conf. Robot. Automat.*, 2019, pp. 2118–2124.

[11] A. H. Qureshi, J. Dong, A. Choe, and M. C. Yip, "Neural manipulation planning on constraint manifolds," *IEEE Robot. Automat. Lett.*, vol. 5, no. 4, pp. 6089–6096, Oct. 2020.

[12] A. Srinivas, A. Jabri, P. Abbeel, S. Levine, and C. Finn, "Universal planning networks: Learning generalizable representations for visuomotor control," in *Proc. Int. Conf. Mach. Learn.*, 2018, pp. 4732–4741.

[13] M. Watter, J. Springenberg, J. Boedecker, and M. Riedmiller, "Embed to control: A locally linear latent dynamics model for control from raw images," in *Proc. Adv. Neural Inf. Process. Syst.*, 2015, pp. 2746–2754.

[14] E. Banijamali, R. Shu, M. Ghavamzadeh, H. Bui, and A. Ghodsi, "Robust locally-linear controllable embedding," in *Proc. Int. Conf. Artif. Intell. Statist.*, 2018, pp. 1751–1759.

[15] D. Hafner *et al.*, "Learning latent dynamics for planning from pixels," in *Proc. Int. Conf. Mach. Learn*., 2019, pp. 2555–2565.

[16] Y. Wu *et al.*, "Imagine that! Leveraging emergent affordances for 3D tool synthesis," 2020, *arXiv:1909.13561*.

[17] A. L. Mitchell *et al.*, "First steps: Latent-space control with semantic constraints for quadruped locomotion," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2020, pp. 5343–5350.

[18] D. Erhan, Y. Bengio, A. Courville, and P. Vincent, "Visualizing higher-layer features of a deep network," Univ. Montreal, Tech. Rep. 1341, 2009.

[19] J. Bialkowski, S. Karaman, M. Otte, and E. Frazzoli, "Efficient collision checking in sampling-based motion planning," *Springer Tracts Adv. Robot.*, vol. 86, pp. 365–380, 2013.

[20] J. Schulman *et al.*, "Motion planning with sequential convex optimization and convex collision checking," *Int. J. Robot. Res.*, vol. 33, no. 9, pp. 1251–1270, 2014.

[21] M. Mukadam, J. Dong, X. Yan, F. Dellaert, and B. Boots, "Continuous-time Gaussian process motion planning via probabilistic inference," *Int. J. Robot. Res.*, vol. 37, no. 11, pp. 1319–1340, 2018.

[22] E. Banijamali *et al.*, "Robust locally-linear controllable embedding," in *Proc. Int. Conf. Artif. Intell. Statist.*, 2018, pp. 1751–1759.

[23] M. Karl, M. Soelch, J. Bayer, and P. V. der Smagt, "Deep variational bayes filters: Unsupervised learning of state space models from raw data," 2016, *arXiv:1605.06432*.

[24] D. Hafner *et al.*, "Learning latent dynamics for planning from pixels," in *Proc. Int. Conf. Mach. Learn*, 2019, pp. 2555–2565.

[25] M. Y. Seker, M. Imre, J. H. Piater, and E. Ugur, "Conditional neural movement primitives," in *Proc. Robot.: Sci. Syst.*, 2019, vol. 10.

[26] B. Bócsi, D. Nguyen-Tuong, L. Csató, B. Schölkopf, and J. Peters, "Learning inverse kinematics with structured prediction," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2011, pp. 698–703.

[27] H. Ren and P. Ben-Tzvi, "Learning inverse kinematics and dynamics of a robotic manipulator using generative adversarial networks," *Robot. Auton. Syst.*, vol. 124, 2020, Art. no. 103386.

[28] D. E. Whitney, "Resolved motion rate control of manipulators and human prostheses," *IEEE Trans. Man- Mach. Syst.*, vol. 10, no. 2, pp. 47–53, Jun. 1969.

[29] A. Goldenberg, B. Benhabib, and R. Fenton, "A complete generalized solution to the inverse kinematics of robots," *IEEE J. Robot. Autom.*, vol. 1, no. 1, pp. 14–20, Mar. 1985.

[30] C. W. Wampler, "Manipulator inverse kinematic solutions based on vector formulations and damped least-squares methods," *IEEE Trans. Syst., Man, Cybern.*, vol. 16, no. 1, pp. 93–101, Jan. 1986.

[31] D. P. Kingma and M. Welling, "Auto-encoding variational bayes," in *Proc. Int. Conf. Learn. Representations*, 2014.

[32] D. J. Rezende, S. Mohamed, and D. Wierstra, "Stochastic backpropagation and approximate inference in deep generative models," in *Proc. Int. Conf. Mach. Learn.*, 2014, pp. 1278–1286.

[33] I. Higgins *et al.*, "beta-VAE: Learning basic visual concepts with a constrained variational framework," in *Proc. Int. Conf. Learn. Representations*, 2017.

[34] D. J. Rezende and F. Viola, "Taming VAEs," 2018, *arXiv:1810.00597*.

[35] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," in *Proc. Int. Conf. Learn. Representations*, 2015.

[36] E. B. Wilson, "Probable inference, the law of succession, and statistical inference," *J. Amer. Stat. Assoc.*, vol. 22, no. 158, pp. 209–212, 1927.

[37] O. Khatib, "Real-time obstacle avoidance for manipulators and mobile robots," in *Autonomous Robot Vehicles*. Berlin, Germany: Springer, 1986, pp. 396–404.

[38] F. Flacco, T. Kröger, A. De Luca, and O. Khatib, "A depth space approach to human-robot collision avoidance," in *Proc. IEEE Int. Conf. Robot. Automat.*, 2012, pp. 338–345.

[39] J. J. Kuffner and S. M. LaValle, "RRT-Connect: An efficient approach to single-query path planning," in *Proc. Millennium Conf. IEEE Int. Conf. Robot. Automat. Symposia Proc. (Cat. No 00CH37065)*, 2000, vol. 2, pp. 995–1001.

[40] I. A. Şucan and L. E. Kavraki, "Kinodynamic motion planning by interior-exterior cell exploration," in *Algorithmic Foundation of Robotics 8th*. Berlin, Germany: Springer, 2009, pp. 449–464.

[41] R. Bohlin and L. E. Kavraki, "Path planning using lazy PRM," in *Proc. Millennium Conf. IEEE Int. Conf. Robot. Automat. Symposia Proc. (Cat. No. 00CH37065)*, 2000, vol. 1, pp. 521–528.

[42] L. Janson, E. Schmerling, A. Clark, and M. Pavone, "Fast marching tree: A fast marching sampling-based method for optimal motion planning in many dimensions," *Int. J. Robot. Res.*, vol. 34, no. 7, pp. 883–921, 2015.

[43] J. D. Gammell, T. D. Barfoot, and S.S. Srinivasa, "Batch informed trees (BIT*): Informed asymptotically optimal anytime search," *Int. J. Robot. Res.*, vol. 39, no. 5, pp. 543–567, 2020.

[44] S. Chitta, I. Sucan, and S. Cousins, "MoveIt! [ROS Topics]," *IEEE Robot. Automat. Mag.*, vol. 19, no. 1, pp. 18–19, Mar. 2012.

[45] I. A. Sucan, M. Moll, and L. E. Kavraki, "The open motion planning library," *IEEE Robot. Automat. Mag.*, vol. 19, no. 4, pp. 72–82, Dec. 2012.