Leveraging Scene Embeddings for Gradient-Based Motion Planning in Latent Space

Jun Yamada*[1], Chia-Man Hung*[1,2], Jack Collins[1], Ioannis Havoutis[2], Ingmar Posner[1]

***Abstract*— Motion planning framed as optimisation in structured latent spaces has recently emerged as competitive with traditional methods in terms of planning success while significantly outperforming them in terms of computational speed. However, the real-world applicability of recent work in this domain remains limited by the need to express obstacle information directly in *state-space*, involving simple geometric primitives. In this work we address this challenge by leveraging learned scene embeddings together with a generative model of the robot manipulator to drive the optimisation process. In addition we introduce an approach for efficient collision checking which directly regularises the optimisation undertaken for planning. Using simulated as well as real-world experiments, we demonstrate that our approach, AMP-LS, is able to successfully plan in novel, complex scenes while outperforming competitive traditional baselines in terms of computation speed by an order of magnitude. We show that the resulting system is fast enough to enable closed-loop planning in real-world dynamic scenes.**

I. Introduction

Motion planning is a core capability for robotic manipulation tasks [1], [2] with the fundamental aim of planning a collision-free path from the current state of an articulated configuration of joints to a predefined goal joint or end-effector pose configuration. Sampling-based motion planning algorithms, such as Rapidly-Exploring Random Trees (RRT) [3] and Probabilistic Roadmap (PRM) [4], are widely used within the robotics community as they have well understood properties in regards to planning time and collision avoidance. However, sampling-based methods become increasingly intractable as the problem size increases (i.e., Degrees-of-Freedom (DoF) of the robot, environment complexity, and length of the path) and are also typically too slow to be used for closed-loop planning, as any change to the environment requires re-planning [5].

Recently, learning-based motion planning [6], [7] has gained the attention of the robotics community with the promise of increased computational efficiency and faster planning times. Notably, Latent Space Path Planning (LSPP) [8] introduces motion planning via gradient-based optimisation in the latent space of a VAE. The success rate of LSPP is commensurate with that of commonly used sampling and gradient-based motion planners, but with significantly reduced planning time. By learning a structured latent space using kinematically feasible and easily generated robot states, a learned latent space that is optimised via activation maximisation (AM) [9] can produce diverse and adaptive behaviours [10]. However, LSPP relies on state-based obstacle representation given known object shape, which does not easily transfer to real-world environments.

*Equal contribution.
[1]Applied AI Lab (A2I), [2]Dynamic Robot Systems (DRS)
Oxford Robotics Institute (ORI), University of Oxford
Correspondence to: jyamada@robots.ox.ac.uk

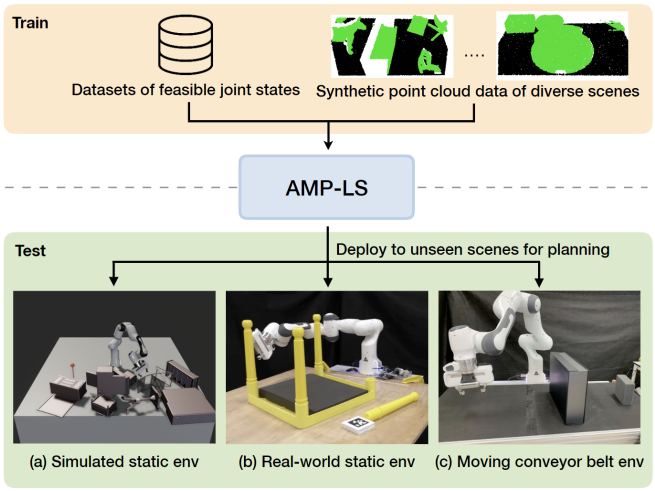

Fig. 1: **Problem setup.** AMP-LS generates a collision-free trajectory via gradient-based optimisation by leveraging scene embeddings. Our model is trained on kinematically feasible robot joint states and synthetic point cloud of diverse scenes. For evaluation, our method is deployed to unseen scenes including simulated and real-world environments: (a) *Simulated static env*: Novel scenes are generated by randomly placing obstacles on a table. (b) *Real-world static env*: A robot avoids the table legs to reach the pre-grasp location of the unassembled table leg. (c) *Moving Conveyor Belt env*: A robot reaches a moving target object while avoiding an obstacle on the conveyor belt by using closed-loop planning.

To address the issues of LSPP, we introduce a method significantly extending the prior work by incorporating a collision predictor that leverages scene embeddings and efficient collision checking, which regularises the optimisation during planning for safe collision avoidance. We name this new method Activation Maximisation Planning in Latent Space (AMP-LS). Specifically, we adapt SceneCollisionNet [11], trained on diverse synthetic point cloud data of scenes generated with objects from ShapeNet datasets [12], for our purpose to facilitate zero-shot transfer to unseen environments including the real-world scenes (see Fig. 1). Due to the speed of our approach, we also show that our method can be applied to closed-loop settings where both the obstacles and goal pose are moving.

The contributions of our work are threefold: (1) we present Activation Maximisation Planning in Latent Space (AMP-LS), which significantly extends LSPP by incorporating a collision predictor that leverages scene embeddings and explicit collision checking in order to reguralise optimisation when planning for obstacle avoidance; (2) we empirically demonstrate that our approach can be zero-shot transferred to

unseen scenes, including real-world environments, through the use of a collision predictor that is trained on diverse synthetic scenes; (3) we show that our method can be applied to closed-loop settings with reactive behaviour, capable of reaching a *moving* target while also avoiding a *moving* obstacle.

II. Related Works

Sampling-based motion planning approaches such as RRT [3], [13] and PRM [4] are widely used to generate collision-free trajectories in robotics. PRM requires a pre-computed roadmap; RRT often struggles to find the solution with the shortest path. While several extensions such as RRT* [13] and BIT* [14] have been proposed to achieve asymptotic optimality and reduce computational cost, these approaches typically demand many samples—a runtime problem that compounds with increases in robot DoF, environmental complexity, or path length [15]. Another limitation of sampling-based motion planners is that they do not support the real-time planning as re-planning is required to navigate dynamic environments.

Optimisation-based planning approaches such as covariant Hamiltonian optimisation for motion planning (CHOMP) [16] and Stochastic Trajectory Optimisation for Motion Planning (STOMP) [17] require a large number of trajectory states when given multiple constraints. These approaches typically start from an initial guess, a trajectory linking the start and desired end states, which is refined through minimisation of a cost function. Computation terminates when a stop condition is reached or the algorithm times out. The artificial potential algorithm [18], [19] is perhaps the closest optimisation-based planning approach to our work. It achieves real-time obstacle avoidance by creating attractive and repulsive fields around goals and obstacles. End-effector movement is then guided by the gradient of these fields. Although appealing in its simplicity, it struggles to handle additional constraints on properties that cannot be fully determined by robot joint configuration.

Several recent works attempt to leverage neural networks for motion planning. Neural motion planning methods [20], [21], [6], [22] employ imitation learning (IL) on expert demonstrations generated by a sampling-based motion planner or reinforcement learning (RL) [23] to learn motion policies. However, many samples are required to train such policies in complex environments. These methods also struggle to generalise to unseen scenes.

Another set of works performs planning in learned latent space [24], [8]. L2RRT [21] plans a path in a learned latent space using RRT. Our work builds upon *Latent Space Path Planning* (LSPP) [8]. LSPP plans a trajectory for a robot via iterative optimisation using activation maximisation (AM) [9] in a latent space of the robot kinematics learned by a generative model. Leveraging a collision predictor as a constraint, LSPP successfully plans a collision-free path with improved efficiency in planning time. However, LSPP approximates a scene as a set of cylindrical obstacles and requires state-based knowledge of the scene, such as position and shape of obstacles. Such narrow scene definitions and lack of complete information limits the application of this method to real-world problems.

To successfully generate a collision-free path in a scene with obstacles, learning a collision predictor to identify collision between a robot and the scene is essential. Prior neural motion planning methods [20], [21], [6], [25] learn obstacle representations either from 2D images, occupancy grids, or point clouds, instead of explicitly predicting a probability of collision. SceneCollisionNet [11] learns the scene embeddings for a collision predictor from a large number of synthetic scenes generated with diverse objects from ShapeNet [12]. To leverage a collision predictor as a constraint for motion planning, we utilise SceneCollisionNet and adapt it to work within our latent planning framework.

III. Approach

In this work, we introduce a method remarkably extending the prior work [8] and name it Activation Maximisation Planning in Latent Space (AMP-LS). Similar to the prior work [8], AMP-LS leverages a variational autoencoder (VAE) [26], [27] to learn a structured latent space to generate kinematically feasible joint trajectories. While a collision predictor in the prior work relies on state-based obstacle representations, our collision predictor leverages scene embeddings obtained from SceneCollisionNet [11] to readily achieve zero-shot transfer to unseen environments. Further, we present an approach for explicit collision checking to directly regularise the optimisation to plan collision-free trajectories. In the following section, we describe an overview of our model (see Fig. 2) and optimisation objective for planning.

A. Problem Formulation

Similar to [8], we consider the problem of generating a collision-free trajectory consisting of robot joint configurations $\{\mathbf{q}_0, \dots, \mathbf{q}_T\}$ for a robot in an environment with obstacles. A state $\mathbf{x}_t$ at time t consists of a kinematically feasible robot joint configuration $\mathbf{q}_t$, and its end-effector position and orientation $\mathbf{e}_t^{pos}$ and $\mathbf{e}_t^{ori}$. The end-effector orientation $\mathbf{e}^{ori}$ employs a 6D representation of SO(3), which consists of the first two column vectors in a rotation matrix $\mathbf{R}$. This representation is suitable for learning rotations using neural networks due to its property of continuity [28]. Note that no prior information of obstacles (e.g., mesh) is given. An observation $\mathbf{o}_t \in \mathcal{R}^{n\times 3}$ at time t is defined as a point cloud with n points from a third-person camera. $\mathbf{o}_t$ includes only scene information; thus, the robot point cloud is excluded from the raw point cloud. The extraction of the robot point cloud is readily achieved using MoveIt [29].

B. Learning Latent Representations of Robot State

To plan cohesive paths for the manipulator using a learned latent space, the latent space must be structured and disentangled. Leveraging a VAE [26], [27], prior work [8] has successfully learned such a latent space and captured a notion of local distance in joint space. In their representation, poses that are close to each other in joint space are also close in latent space. Similarly, we also learn a VAE consisting of an encoder $q_\phi(\mathbf{z}|\mathbf{x})$ and decoder $p_\theta(\mathbf{x}|\mathbf{z})$, where $\mathbf{z}$ is the latent representation. To train the VAE, rather than directly 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, we instead optimise the evidence lower bound (ELBO) $\mathcal{L}^{\mathrm{ELBO}} \leq p(\mathbf{x})$:

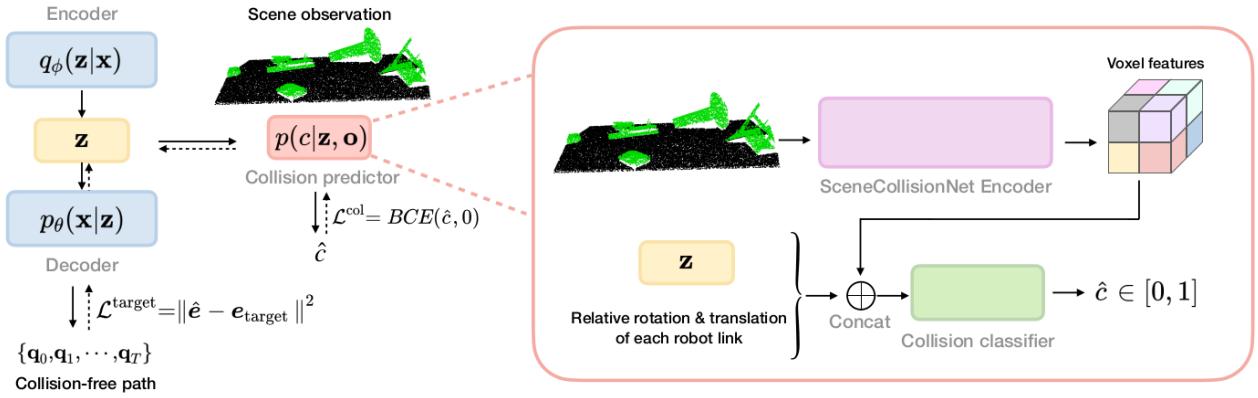

Fig. 2: **Our method overview.** A VAE (blue) is trained using feasible robot states $\mathbf{x}$ consisting of joint states, end-effector position, and end-effector orientation to learn structured latent representations $\mathbf{z}$ (yellow). Then, freezing the weights of the pre-trained encoder in the VAE, the collision predictor (red) takes as input the learned latent representation $\mathbf{z}$ and a scene point cloud observation $\mathbf{o}$. The collision predictor built upon SceneCollisionNet learns to output a probability $\hat{c}$ of collision between the robot arm and obstacles. To plan a collision-free trajectory, gradient-based optimisation is applied to produce a sequence of latent representations $\{\mathbf{z}_t\}_{t=1}^T$ each of which has low probability of collision with the scene using the learned collision predictor. A sequence of joint states $\{\mathbf{q}_t\}_{t=1}^T$ is generated by decoding the sequence of latent representations $\{\mathbf{z}_t\}_{t=1}^T$ using the trained decoder in the VAE.

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

There is a trade-off in the ELBO loss between the reconstruction accuracy and the KL term: accurate reconstruction at the cost of poorly structured latent space, on one hand, or well structured latent space but noisy reconstruction, on the other. These terms are often manually weighted in the ELBO formulation [30]. An alternative to manually tuning the weight is to use GECO [31]. GECO adaptively tunes the trade-off between reconstruction and regularisation by formulating the ELBO loss as a constrained optimisation problem with a Lagrange multiplier λ:

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

This encourages the model to optimise the reconstruction accuracy first, until the it reaches a predefined target. The KL term is then optimised. The generative model is trained on a dataset of valid joint states of the robot.

C. Activation Maximisation for Motion Planning

Our goal is to plan a trajectory consisting of robot joint configurations towards a target pose. That is, given a target end-effector position $\mathbf{e}_{\text{target}}^{pos}$ and orientation $\mathbf{e}_{\text{target}}^{ori}$, a sequence of joint configurations $\{\mathbf{q}_0, \ldots, \mathbf{q}_T\}$ that leads a robot to the target pose is generated. Leveraging the trained VAE inspired by the prior work [8], we can compute such a sequence of robot joints by decoding the latent representation of the VAE model $\{\mathbf{z}_0, \ldots, \mathbf{z}_T\}$. This sequence of the latent representation is computed in a probabilistic model through activation maximisation (AM) [9]:

$$\mathbf{z}_{t+1} = \mathbf{z}_t - \alpha_{\text{AM}} \nabla \mathcal{L}^{\text{AM}} \tag{3}$$

where

$$\mathcal{L}^{\text{AM}} = \lambda_{\text{pos}} \underbrace{\left\|\hat{\boldsymbol{e}}^{pos}, \boldsymbol{e}_{\text{target}}^{pos}\right\|_2}_{\text{Target Position Loss}} + \lambda_{\text{ori}} \underbrace{\left\|\hat{\boldsymbol{e}}^{ori}, \boldsymbol{e}_{\text{target}}^{ori}\right\|_2}_{\text{Target Orientation Loss}} + \underbrace{(-\log p(\mathbf{z}))}_{\text{Prior Loss}} \tag{4}$$

In contrast to the prior work [8], we also introduce an end-effector orientation constraint, which is generally useful for reaching a pre-grasp pose. The first latent representation $\mathbf{z_0}$ is acquired by encoding the current/starting robot state $\mathbf{z}_0 \sim q_\phi(\mathbf{z}|\mathbf{x} = \mathbf{x}_0)$. Note that model parameters are not updated, but only the parameterised latent variable $\mathbf{z}$ is iteratively updated. The first two terms in $\mathcal{L}^{\text{AM}}$ (Eq. 4) guide the latent representation to decode to robot joint state that approaches the target pose. The third term is the likelihood of the current representation under its prior, which is introduced in [8] to encourage the latent representation to stay close to the training distribution, thus decoding to kinematically feasible pair of joint position and end-effector pose.

D. Collision Constraints

To generate a collision-free trajectory, similar to that used in prior work [8], we add collision constraints to the objective function in Eq. 4 by introducing a collision predictor. While the prior work uses a narrowly defined state-based obstacle representation as input to the collision predictor, in our approach, we adapt SceneCollisionNet [11] to embed scene observations for zero-shot transfer to unseen environments. Specifically, the voxel features from SceneCollisionNet are concatenated with the latent representation of the robot $\mathbf{z}$, the relative translation, and the rotation from the centre of each SceneCollisionNet voxel as an input to a collision classifier to predict the probability of collision $\hat{c}$ between the robot and obstacles (see Fig. 2). Note that we train the collision predictor only on features of voxels closest from each robot link to ignore unnecessary voxel information. While training the collision predictor, the weights of the pre-trained VAE are

Algorithm 1 Planning a collision-free path in latent space via activation maximisation

Initialise a buffer $D = \{\mathbf{q}_0\}$, λ_{pos}, λ_{ori}, λ_{col}, $\mathbf{q}_{\text{prev}} = \mathbf{q}_0$
$\mathbf{z}_0 \sim q_\phi(\mathbf{z}|\mathbf{x} = \mathbf{x}_0)$
for $t = 0, 1, 2, \ldots, H$ **do**
  $\{\hat{\mathbf{q}}_t, \hat{\mathbf{e}}_t^{pos}, \hat{\mathbf{e}}_t^{ori}\} \sim p_\theta(\mathbf{x}|\mathbf{z} = \mathbf{z}_t)$
  if $t > 0$ and $p_\vartheta(\mathbf{z}_t, \mathbf{o}_t) < \gamma_{\text{col}}$ **then**
    $\{\mathbf{q}_{\text{prev}}, \ldots, \hat{\mathbf{q}}_t\} = f_{\text{interpolate}}(\mathbf{q}_{\text{prev}}, \hat{\mathbf{q}}_t)$
      ▷ Linear interpolation between $\mathbf{q}_{\text{prev}}$ and $\hat{\mathbf{q}}_t$
    if collision in $\{\mathbf{q}_{\text{prev}}, \ldots, \hat{\mathbf{q}}_t\}$ **then**
      $i \leftarrow$ index of the first joint state with collision in the interpolated trajectory
      $m \leftarrow |\{\mathbf{q}_{\text{prev}}, \ldots, \hat{\mathbf{q}}_t\}|$
      Reduce λ_{pos} and λ_{ori} by $\frac{i}{m}$
      $\hat{\mathbf{q}}_t \leftarrow \mathbf{q}_{\text{prev}}$, $\mathbf{z}_t \leftarrow \mathbf{z}_{\text{prev}}$
        ▷ Back trace to the previous joint and latent representations $\mathbf{q}_{\text{prev}}$ and $\mathbf{z}_{\text{prev}}$ for replanning
    else
      $D \leftarrow D \cup \{\mathbf{q}_{\text{prev}}, \ldots \hat{\mathbf{q}}_t\}$
      if $d(\hat{e}_t, e_{\text{target}}) < \gamma$ **then**
        break
      end if
      $\mathbf{q}_{\text{prev}} \leftarrow \hat{\mathbf{q}}_t$, $\mathbf{z}_{\text{prev}} \leftarrow \mathbf{z}_t$
    end if
  end if
  Compute losses (Eq. 5)
  Update λ_{pos}, λ_{ori}, and λ_{col} using GECO
  $\mathbf{z}_{t+1} \leftarrow \mathbf{z}_t - \alpha_{\text{AM}} \nabla \mathcal{L}_t^{\text{AM}}$
end for

frozen so that the pre-trained latent space does not change. The collision predictor is trained using the binary cross-entropy (BCE) loss with ground truth collision labels. To drive the latent representation away from obstacles, we incorporate the collision predictor loss into Eq. 4:

$$\begin{aligned}\mathcal{L}^{\text{AM}} = \lambda_{\text{pos}} \underbrace{\left\| \hat{e}^{pos}, e^{pos}_{\text{target}} \right\|_2}_{\text{Target Position Loss}} + \lambda_{\text{ori}} \underbrace{\left\| \hat{e}^{ori}, e^{ori}_{\text{target}} \right\|_2}_{\text{Target Orientation Loss}} \\ + \lambda_{\text{col}} \underbrace{\left(-\log\left(1 - p_\vartheta\left(\mathbf{z}, \mathbf{o}\right)\right)\right)}_{\text{Collision Loss}} + \underbrace{\left(-\log p(\mathbf{z})\right)}_{\text{Prior Loss}}\end{aligned} \tag{5}$$

During the planning, three coefficients λ_{pos}, λ_{ori}, and λ_{col} are automatically and dynamically adjusted by GECO [31]. Minimising the collision loss during AM optimisation drives the latent representation $\mathbf{z}$ towards the representation whose decoded joint configuration is collision-free.

E. Collision Checking

While prior work [8] simply optimises the objective function until it reaches a target, we observe that it is hard to perfectly balance multiple loss terms and that such simple optimisation often results in collision between the robot and obstacles. In contrast to the target losses, the collision loss is inherently a hard constraint that should not be violated at any point in the trajectory. To address this issue, our high-level idea is that collision can be predicted and avoided before execution and the coefficients of the objective function determine the direction in which the latent representation is heading towards. Specifically, we introduce explicit collision checking and automatic rescaling for coefficients during the planning to avoid obstacles more safely. That is, if a collision probability of the decoded joint configuration is higher than a predefined threshold γ_{col}, we reject such robot configuration that is highly likely to be in collision and keep optimising the latent space until the decoded joint state is collision-free. Then, we interpolate a trajectory between the current and decoded collision-free joint state in m steps and pass them to the collision predictor to check for collision. If there is any collision in the interpolated trajectory, we obtain its index i of the joint state with collision closest to the current joint state and reduce the coefficients of the target position and orientation loss by multiplying by $\frac{i}{m}$, to encourage the optimisation to minimise the collision loss. Intuitively, this scaling induces the robot to deviate from the original route drastically depending on how close it is to an obstacle. This process continues until the collision-free next joint state is found and there is no collision in the interpolated trajectory between the current joint state and the next joint state. In our experiments, we use the threshold of $\gamma_{\text{col}} = 0.3$ chosen by a grid search. For further details, see Algorithm 1.

IV. Implementation Details

A. Architecture Details

Our VAE encoder and decoder consist of three fully connected hidden layers with 512 units and ELU activation functions [32]. The input dimension to the VAE is 16, consisting of robot joint states $\mathbf{q} \in \mathcal{R}^7$, end-effector position $\mathbf{e}^{pos} \in \mathcal{R}^3$ and 6D representation of end-effector rotation matrix $\mathbf{e}^{ori} \in \mathcal{R}^6$. The dimension of the latent space $\mathbf{z}$ is 7. The collision classifier which takes as input the voxel features, the learned latent representation of robot states, and relative rotation and translation of each robot link, consists of one hidden fully connected layer with units of $[1024, 256]$.

B. Training Details

The VAE is trained using kinematically feasible robot joint configurations. To generate such joint states, we leverage the Flexible Collision Library (FCL) [33] for self-collision checking. The VAE model is trained with a batch size of 256 for about $2M$ training iterations using the Adam optimiser [34] with learning rate of $3\text{e}{-4}$ on a GeForce RTX 3090. Throughout training, valid robot configurations are generated on the fly as it is cheap to do so. In total, the model is exposed to around $500M$ configurations.

The collision predictor is trained on diverse synthetic point cloud data to assist zero-shot transfer to scenes with unseen obstacles. Such scenes are generated by placing objects randomly sampled from the ShapeNet dataset [12], consisting of 8828 3D meshes. Each object is placed on a planar surface with random position and rotation. We sample the number of objects placed on the surface from a uniform distribution between 4 and 8. To train the collision predictor, a new scene is procedurally generated for each training iteration similar to the prior work [11], and we randomly sample 2048 instances of kinematically feasible robot joint configurations and check for collisions between each robot configuration and the generated scene using FCL. A third-person RGB-D camera is directed towards the centre of the scene to sample point clouds. The camera extrinsics are randomly sampled for each query from a predefined range of roll, yaw, and pitch

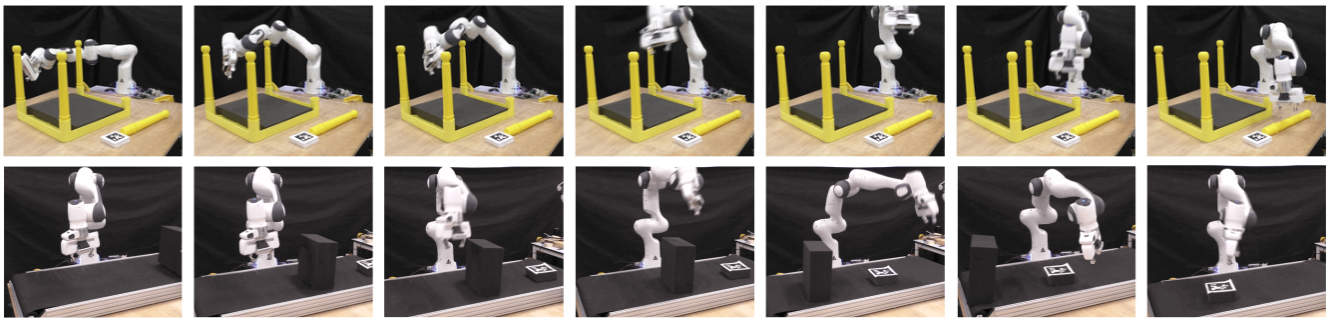

Fig. 3: Visualisation of real-world experiments. **Top**: Our method successfully plans a collision-free trajectory in a complex real-world scene from an impeded start configuration to a pre-grasp goal configuration. By training a collision predictor on diverse synthetic scenes, our method can readily transfer to such unseen scene. **Bottom**: AMP-LS can be applied to closed-loop planning to avoid moving obstacles and reach a moving target object on a conveyor. This reader is referred to our supplementary video for better visualisation.

parameters. Thus, 2048 unique valid robot configurations and point clouds are procedurally generated for each iteration to train the collision predictor. We train the collision predictor for $1M$ training iterations using SGD with a learning rate of $1\mathrm{e}{-3}$ and with momentum 0.9 for approximately 7 days, which is similar to the training time requirement of SceneCollisionNet.

C. Deploymenet details

In open-loop planning, the current state $\mathbf{x}_0$ is encoded to a latent representation $\mathbf{z}_0$. Then, the encoded latent representation is iteratively optimised through AM optimisation (see Eq. 5) until the end-effector reaches the the target pose with tolerance of γ. In closed-loop planning, while the latent representation is similarly optimised, a point cloud input for the collision predictor and the target pose in the objective function (see Eq. 5) are updated from the current observation at each time step for reactive motion.

V. Experiments

We design our experiments to answer the following guiding questions: (1) how does AMP-LS perform compared to traditional motion planning methods such as sampling and optimisation-based approaches in open-loop settings? (2) does AMP-LS transfer zero-shot to real-world static environments? (3) does AMP-LS cope with dynamic environments using closed-loop planning?

A. Experimental Setup

We evaluate our approach in both simulated and real-world environments. In simulation, we use the Gazebo simulator [35] with ROS. In all of simulated and real-world experiments, we use a 7-DoF Franka Panda robot.

B. Open-Loop Planning for Reaching Static Targets

We evaluate AMP-LS in an open-loop planning setup in a simulated environment. In this experiment, obstacles in the environment are static. We select a range of sampling and optimisation-based motion planners typically used by the robotics community and available within the unified MoveIt! library. We compare our method against several sampling-based motion planners and an optimisation-based motion planner: RRT-Connect [36], RRT* [13], Lazy PRM* [37], LBKPIECE [38], BIT* [14], and CHOMP [16]. CHOMP uses a linear initialisation from start to goal joint positions. Since we assume that complete knowledge of the environment is not available, occupancy maps [39] generated from point clouds are used for collision checking in motion planning baseline methods. We evaluate the methods on 100 novel scenes where objects are randomly placed on a table (see Fig. 1 (a)). The hyperparameters used for GECO to determine coefficients of our objective function (see Eq. 5) are found via a grid search similar to that of prior work [8]. For the baselines, we use the default parameters provided by MoveIt OMPL. For RRT*, Lazy PRM*, and BIT*, a 1 second planning budget is given. Across all methods, a motion plan is considered to be successful if a robot reaches a target within a distance tolerance of 1cm and orientation tolerance of 15 degrees.

As illustrated in Table I, our method achieves reasonable success rate with improved planning time compared to most of the motion planning baselines. Specifically, AMP-LS outperforms CHOMP, which is also an optimisation-based motion planner, by a significant margin because CHOMP requires a large number of trajectories to find a feasible path in complex scenes, in contrast to AMP-LS. AMP-LS still has a competitive success rate against RRT-Connect, but the planning time of AMP-LS is an order of magnitude faster than the baseline. Traditional motion planning baselines often fail to find a collision-free path within a short time and sometimes plan a path with collision due to occlusions in the scenes. In contrast, our collision predictor is trained on diverse synthetic scenes with occlusion and can therefore reason about occluded regions, similar to SceneCollisionNet [11]. While our method demonstrates reasonable accuracy and improved planning efficiency, the path length is longer than most of the other baselines. The longer path length is due to the design of the planning strategy used in the prior work [8] that tunes the coefficients of losses automatically to avoid obstacles, thus not directly minimising the path length. To address this issue, additional optimisation constraints could be explored in the future that focus on reducing the path length.

To verify that constraints such as a prior loss and collision loss successfully induce successful collision-free trajectories, we also ablate the constraints of our objective functions. As illustrated in Table I, the success rate of AMP-LS without a collision loss and prior loss significantly drops. This indicates that our collision predictor successfully constrains the latent space even in novel scenes. Furthermore, AMP-LS without the

	Success rate	Planning time (s)	Path length
AMP-LS (ours)	**0.89 ± 0.06**	**0.26 ± 0.13**	3.63 ± 1.05
AMP-LS w/o col. loss	0.46 ± 0.10	0.12 ± 0.04	3.68 ± 1.29
AMP-LS w/o prior loss	0.35 ± 0.09	0.24 ± 0.21	3.23 ± 1.12
RRT-Connect	**0.86 ± 0.07**	1.60 ± 0.89	**2.17 ± 0.84**
RRT*	0.36 ± 0.09	N/A	2.25 ± 0.78
Lazy PRM*	**0.82 ± 0.08**	N/A	2.26 ± 0.82
LBKPIECE	0.23 ± 0.08	2.54 ± 1.12	2.34 ± 0.92
BIT*	0.63 ± 0.09	N/A	2.42 ± 1.02
CHOMP	0.39 ± 0.10	2.24 ± 0.79	2.41 ± 0.90

TABLE I: Comparison of performance of our method AMP-LS against baseline motion planning algorithms and ablation. We also report 95% confidence interval of Wilson score [40] for success rate and standard deviation for planning time and path length. The path length is normalised by dividing the actual path length by the distance between the initial and target end-effector positions for fairer comparison.

prior loss results into significantly poorer performance as the latent representation is optimised to drive into unseen latent representations which decode to kinematically inconsistent configurations. This is consistent with findings in [8].

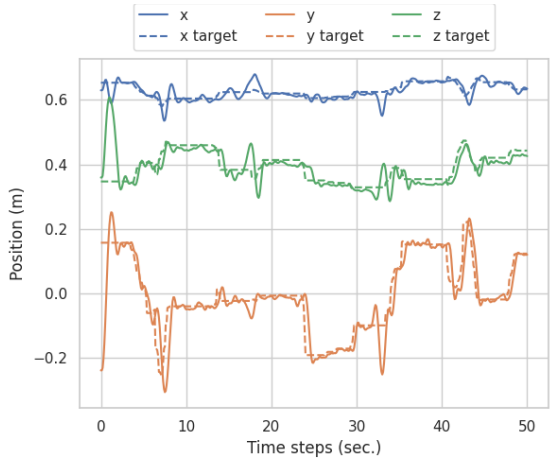

Fig. 4: **Coordinates of end-effector and moving targets in closed-loop settings.** To verify the ability of closed-loop planning in our method, we deploy our method to the real-world robot arm to reach a moving target.

C. Real-World Open-Loop Planning in a Complex Scene

Our method readily transfers to complex real-world scenes. To verify this, we evaluate our method in a complex real-world static scene using open-loop planning as illustrated in Fig. 1 (c). In this task, the robot needs to reach the unassembled table leg while avoiding the other table legs to achieve a pre-grasp pose in a furniture assembly task. We control the robot arm using a torque controller that can track a joint position command. As shown in Fig. 3 Top, our method can successfully plan a collision-free trajectory for a robot starting next to the table legs to avoid the obstacles and reach the unassembled table leg on the table. This demonstrates that our collision predictor, trained on diverse synthetic scenes, is transferable to real-world environments.

D. Closed-Loop Planning for Moving Obstacles and Target

As our method is, by design, an efficient local planner, AMP-LS is able to act reactively when operated as a closed-loop system. To verify the closed-loop potential of AMP-LS we deploy our method on a robot with the goal of reaching a moving target without obstacles. To control the real-world robot, a desired next joint position is sent to a torque controller at 10Hz. Fig. 4 illustrates coordinates of the moving target and the end-effector position over 50 seconds. Since our method can predict the next desired joint state quickly, the robot can reactively follow the moving target.

To further demonstrate the ability of reactive motion using AMP-LS, we evaluate our method on a setup where the robot needs to avoid moving obstacles and reach a target object on a conveyor in both simulated and real-world environments (see Fig. 3 Bottom). Firstly, we quantitatively evaluate our method to examine the ability of reactive motion in the simulated environment. In this evaluation, we randomly generate obstacles with different size, and the obstacle and a target object are randomly placed on the conveyor belt. We observe that the robot successfully avoids the obstacle and reaches a moving target on the conveyor with the success rate of 93.3% (28/30 trials) thanks to the fast planning of our method. Note that we use the threshold of 3cm and 20 degrees in this experiment, because tight tolerance for reaching a moving target is challenging unless a future state of the target is estimated and used for planning.

In the real-world experiment, the robot starts moving towards the target object with the attached AprilTag [41] when it is observed by the third-person camera while avoiding a moving obstacle. For closed-loop planning, the collision predictor takes as input a point cloud data for each time step. As illustrated in Fig. 3 Bottom, the robot successfully avoids the moving obstacle to reach and follow the target object.

VI. Conclusion

In this work, we present AMP-LS, a learning-based motion planning approach that generalises to unseen obstacles in complex environments. AMP-LS is built upon LSPP [8] and inherits a number of desirable properties. However, AMP-LS considerably extends LSPP by introducing a collision predictor trained on diverse synthetic scenes to leverage scene embeddings for unseen scene generalisation, and explicit collision checking during planning for safe obstacle avoidance. We demonstrate that AMP-LS successfully generates collision-free paths in both unseen simulated and real-world scenes. The comparison between AMP-LS and several sampling and optimisation-based motion planning baselines shows that our method achieves commensurate success rate with much improved planning time. Furthermore, our real-world experiments show that AMP-LS can handle both open and closed-loop planning, which significantly broadens the applicability to real-world robotic problems. For future work we look to extend AMP-LS to more complex tasks such as grasping and pick-and-place.

Acknowledgment

This work was supported by a UKRI/EPSRC Programme Grant [EP/V000748/1], we would also 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).

REFERENCES

[1] J. Yamada, Y. Lee, G. Salhotra, K. Pertsch, M. Pflueger, G. S. Sukhatme, J. J. Lim, and P. Englert, "Motion planner augmented reinforcement learning for obstructed environments," in *Conference on Robot Learning*, 2020.

[2] F. Xia, C. Li, R. Martín-Martín, O. Litany, A. Toshev, and S. Savarese, "Relmogen: Leveraging motion generation in reinforcement learning for mobile manipulation," *arXiv preprint arXiv:2008.07792*, 2020.

[3] S. M. Lavalle, "Rapidly-exploring random trees: A new tool for path planning," Iowa State University, Tech. Rep., 1998.

[4] N. M. Amato and Y. Wu, "A randomized roadmap method for path and manipulation planning," in *Proceedings of IEEE International Conference on Robotics and Automation*, 1996.

[5] A. Short, Z. Pan, N. Larkin, and S. Duin, "Recent progress on sampling based dynamic motion planning algorithms," 07 2016, pp. 1305–1311.

[6] A. H. Qureshi, A. Simeonov, M. J. Bency, and M. C. Yip, "Motion planning networks," in *2019 International Conference on Robotics and Automation (ICRA)*. IEEE, 2019, pp. 2118–2124.

[7] A. H. Qureshi, Y. Miao, A. Simeonov, and M. C. Yip, "Motion planning networks: Bridging the gap between learning-based and classical motion planners," *IEEE Transactions on Robotics*, vol. 37, no. 1, pp. 48–66, 2020.

[8] C.-M. Hung, S. Zhong, W. Goodwin, O. P. Jones, M. Engelcke, I. Havoutis, and I. Posner, "Reaching through latent space: From joint statistics to path planning in manipulation," *IEEE Robotics and Automation Letters*, vol. 7, no. 2, pp. 5334–5341, 2022.

[9] D. Erhan, Y. Bengio, A. Courville, and P. Vincent, "Visualizing higher-layer features of a deep network," *Technical Report, Univeristé de Montréal*, 01 2009.

[10] A. L. Mitchell, M. Engelcke, O. P. Jones, D. Surovik, S. Gangapurwala, O. Melon, I. Havoutis, and I. Posner, "First steps: Latent-space control with semantic constraints for quadruped locomotion," in *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2020, pp. 5343–5350.

[11] M. Danielczuk, A. Mousavian, C. Eppner, and D. Fox, "Object rearrangement using learned implicit collision functions," in *2021 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2021, pp. 6010–6017.

[12] A. X. Chang, T. Funkhouser, L. Guibas, P. Hanrahan, Q. Huang, Z. Li, S. Savarese, M. Savva, S. Song, H. Su *et al.*, "Shapenet: An information-rich 3d model repository," *arXiv preprint arXiv:1512.03012*, 2015.

[13] S. Karaman and E. Frazzoli, "Sampling-based algorithms for optimal motion planning," *International Journal of Robotics Research*, vol. 30, no. 7, pp. 846–894, 2011.

[14] J. D. Gammell, T. D. Barfoot, and S. S. Srinivasa, "Batch informed trees (bit*): Informed asymptotically optimal anytime search," *The International Journal of Robotics Research*, vol. 39, no. 5, pp. 543–567, 2020.

[15] K. Hauser, "Lazy collision checking in asymptotically-optimal motion planning," in *2015 IEEE International Conference on Robotics and Automation (ICRA)*, 2015, pp. 2951–2957.

[16] N. Ratliff, M. Zucker, J. A. Bagnell, and S. Srinivasa, "Chomp: Gradient optimization techniques for efficient motion planning," in *2009 IEEE International Conference on Robotics and Automation*, 2009, pp. 489–494.

[17] M. Kalakrishnan, S. Chitta, E. Theodorou, P. Pastor, and S. Schaal, "Stomp: Stochastic trajectory optimization for motion planning," in *2011 IEEE International Conference on Robotics and Automation*, 2011, pp. 4569–4574.

[18] O. Khatib, "Real-time obstacle avoidance for manipulators and mobile robots," in *Proceedings. 1985 IEEE International Conference on Robotics and Automation*, vol. 2. IEEE, 1985, pp. 500–505.

[19] F. Flacco, T. Kröger, A. De Luca, and O. Khatib, "A depth space approach to human-robot collision avoidance," in *2012 IEEE international conference on robotics and automation*. IEEE, 2012, pp. 338–345.

[20] M. Pfeiffer, M. Schaeuble, J. Nieto, R. Siegwart, and C. Cadena, "From perception to decision: A data-driven approach to end-to-end motion planning for autonomous ground robots," in *2017 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, may 2017.

[21] B. Ichter, J. Harrison, and M. Pavone, "Learning sampling distributions for robot motion planning," in *2018 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2018, pp. 7087–7094.

[22] A. H. Qureshi and M. C. Yip, "Deeply informed neural sampling for robot motion planning," in *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2018, pp. 6582–6588.

[23] R. S. Sutton and A. G. Barto, *Reinforcement Learning: An Introduction*, 2nd ed. The MIT Press, 2018.

[24] B. Ichter and M. Pavone, "Robot motion planning in learned latent spaces," *IEEE Robotics and Automation Letters*, vol. 4, no. 3, pp. 2407–2414, jul 2019.

[25] R. Strudel, R. Garcia, J. Carpentier, J.-P. Laumond, I. Laptev, and C. Schmid, "Learning obstacle representations for neural motion planning," *arXiv preprint arXiv:2008.11174*, 2020.

[26] D. P. Kingma and M. Welling, "Auto-encoding variational bayes," *arXiv preprint arXiv:1312.6114*, 2013.

[27] D. J. Rezende, S. Mohamed, and D. Wierstra, "Stochastic backpropagation and approximate inference in deep generative models," 2014.

[28] Y. Zhou, C. Barnes, L. Jingwan, Y. Jimei, and L. Hao, "On the continuity of rotation representations in neural networks," in *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2019.

[29] M. Görner, R. Haschke, H. Ritter, and J. Zhang, "Moveit! task constructor for task-level motion planning," in *2019 International Conference on Robotics and Automation (ICRA)*. IEEE, 2019, pp. 190–196.

[30] I. Higgins, L. Matthey, A. Pal, C. Burgess, X. Glorot, M. Botvinick, S. Mohamed, and A. Lerchner, "beta-VAE: Learning basic visual concepts with a constrained variational framework," in *International Conference on Learning Representations*, 2017.

[31] D. J. Rezende and F. Viola, "Taming vaes," 2018.

[32] D.-A. Clevert, T. Unterthiner, and S. Hochreiter, "Fast and accurate deep network learning by exponential linear units (elus)," *arXiv preprint arXiv:1511.07289*, 2015.

[33] J. Pan, S. Chitta, and D. Manocha, "Fcl: A general purpose library for collision and proximity queries," in *2012 IEEE International Conference on Robotics and Automation*, 2012, pp. 3859–3866.

[34] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," in *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, Y. Bengio and Y. LeCun, Eds., 2015.

[35] N. Koenig and A. Howard, "Design and use paradigms for gazebo, an open-source multi-robot simulator," in *2004 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS) (IEEE Cat. No.04CH37566)*, vol. 3, 2004, pp. 2149–2154 vol.3.

[36] J. J. Kuffner and S. M. LaValle, "Rrt-connect: An efficient approach to single-query path planning," in *Proceedings of IEEE International Conference on Robotics and Automation*, vol. 2. IEEE, 2000, pp. 995–1001.

[37] R. Bohlin and L. E. Kavraki, "Path planning using lazy prm," in *Proceedings 2000 ICRA. Millennium conference. IEEE international conference on robotics and automation. Symposia proceedings (Cat. No. 00CH37065)*, vol. 1. IEEE, 2000, pp. 521–528.

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

[39] A. Hornung, K. M. Wurm, M. Bennewitz, C. Stachniss, and W. Burgard, "OctoMap: An efficient probabilistic 3D mapping framework based on octrees," *Autonomous Robots*, 2013.

[40] E. B. Wilson, "Probable inference, the law of succession, and statistical inference," *Journal of the American Statistical Association*, vol. 22, no. 158, pp. 209–212, 1927.

[41] E. Olson, "Apriltag: A robust and flexible visual fiducial system." in *ICRA*. IEEE, 2011.