From Visuomotor Control to Latent Space Planning for Robot Manipulation

Chia-Man Hung

Keble College

University of Oxford

A thesis submitted for the degree of

Doctor of Philosophy

Trinity 2022

From Visuomotor Control to Latent Space Planning for Robot Manipulation
Candidate: Chia-Man Hung
Supervisors: Dr. Ioannis Havoutis, Prof. Ingmar Posner
Examiners: Dr. Lars Kunze, Prof. Edward Johns

University of Oxford
Dynamic Robot Systems Group (DRS)
Applied Artificial Intelligence Lab (A2I)
Oxford Robotics Institute (ORI)
Department of Engineering Science

Statement of Authorship

This thesis is submitted to the Department of Engineering Science, University of Oxford, in fulfilment of the requirements for the degree of Doctor of Philosophy. This thesis is entirely my own work, and except where otherwise stated, describes my own research.

Chia-Man Hung
Keble College
October 2022

Acknowledgements

First and foremost, I am eternally grateful and indebted to my supervisors Dr. Ioannis Havoutis and Prof. Ingmar Posner for their endless support, enthusiasm, understanding, and patience in guiding me through this journey. Their research insights and constructive feedback have been invaluable in leading me through the ups and downs and in shaping my work over the past few years.

This thesis would not have happened had I not been inspired to pursue research by my former internship host Ralf Perpeet at Google London. He has not only taught me technical skills while working together, but has also become my role model, someone I aspire to be. Likewise, I thank Dr. Stephan Wenger for sharing his expertise at Google Zurich. I equally value the guidance Mme. Marie-Claire Casteran provided during my time at Lycée d'État Jean Zay. I take this opportunity to thank Prof. Maks Ovsjanikov, Prof. Luca Castelli Aleardi, Prof. Renaud Keriven, and Prof. Yanlei Diao for sharing their knowledge and energy at Ecole Polytechnqiue, as well as Monsieur Mansuy, Monsieur Cubizolles, Monsieur Pommelet, and Monsieur Logeais for giving me a foundation of mathematics and physics and sharing the passion for science through their teaching at Lycée Louis le Grand.

I am also immensely grateful for my collaborators and labmates at the Oxford Robotics Institute. From brainstorming wild ideas, pushing through deadlines, to lifting each other up, they have made my research life so much more enjoyable. Especially, I would like to thank Oliver Groth, Yizhe Wu, Kevin Li Sun, Walter Goodwin, Shaohong Zhong, Jun Yamada, Sasha Salter, Oiwi Parker Jones, Jack Collins, Martin Engelcke, Rob Weston, Alexander Mitchell, Sudhanshu Kasewa, Mark Finean, and Rowan Border. Special mention deserves Jack Collins for his help with proofreading this manuscript and providing useful suggestions.

The work described in this thesis was funded by the Clarendon Fund and Keble College Sloane Robinson Scholarship, as part of the UKRI EPSRC Centre for Doctoral Training (CDT) in Autonomous Intelligent Machines and Systems (AIMS). I thank them for providing the financial support allowing me to undertake this research, and for giving me the opportunity to meet so many interesting people.

Getting through the PhD programme requires more than academic support, and I would like to take this opportunity to express my gratitude to our AIMS CDT administrator Wendy Poole. She is the unsung hero who has resolved all my requests well and promptly, saving me from potential disasters every single time.

I would like to acknowledge the use of the University of Oxford Advanced Research Computing (ARC) facility in carrying out this research (`http://dx.doi.org/10.5281/zenodo.22558`) and the use of Hartree Centre Resources via JADE HPC UK. In addition, I would like to express my gratitude to their technical support teams for the help with various issues.

I am fortunate to have crossed paths with Ruo-Chun Tzeng, Chi-Yun Hsu, Chia-An Liu, Jhih-Huang Li, Chieh-An Lin, Tikai Chang, Hsueh-Yung Lin, Quei-An Chen, Li-Ming Tu, Hua-Ting Yao, Bo-Yuan Huang, En-Hung Chao, Yueh-Ning Lee, Hung-Ling Chen, Anne You Wu, Clémence Graftieaux, Mélanie Finas, Akrem Bahri, Pritish Luchoomun, Bettina Chung, Ana-Maria Cretu, Zhengying Liu, Dexiong Chen, Yuesong Shen, Yixin Shen, Alexandre Tuel, Joseph-André Turk, Khaled Zaouk, Hala Lamdouar, Ada Alevizaki, Siddhant Gangapurwala, Yuki Asano, and Tim Rudner. Every one of them has shared some of my burden. I would like to thank all my friends who have helped me, some of whom I may have missed here.

Outside of work, I have probably spent more time than I should have in gliding and judo, but nothing beats the feeling of freedom in the blue silent sky and the sense of accomplishment from a beautifully executed throw. To Ania Liu, Dinant Riks, Julie Dequaire, Bogdan Toader, Philipp Kerth, Pete Stratten, Jamie Allen, George Tvalashvili, Jim Gulliver, Bob Bromwich, Jeremy Mahrer, Alison Mulder, Tai-Ying Lee, Chris Doherty, Carol Doherty, Jacob Armstrong, Alice Godson, Brittany-Amber Jacobs, Miles Soloman, Eugene Soh, Bailey Anderson, Henrique Aguiar, Fenella Gross, Chi-Yu Liu, Coach Hsiao-Mao, Coach Shih, and Master Yu-Kang – thank you for all the moments we have shared together and helping me to be a better person in various aspects of my life.

To my parents and my brother – thank you for your kindness and support, without which I would not have come this far. This thesis stands as a testament to your unconditional love.

Lastly, to my co-pilot Ruoqi He – thank you for your continued technical and emotional support. It would be an understatement to say that we have experienced some ups and downs in the past few years. Every time I felt like giving up, you were there to help me get back on track. You know how much you have contributed to my journey.

Abstract

Deep visuomotor control is emerging as an active research area for robot manipulation. Recent advances in learning sensory and motor systems in an end-to-end manner have achieved remarkable performance across a range of complex tasks. Nevertheless, a few limitations restrict visuomotor control from being more widely adopted as the de facto choice when facing a manipulation task on a real robotic platform. First, imitation learning-based visuomotor control approaches tend to suffer from the inability to recover from an out-of-distribution state caused by compounding errors. Second, the lack of versatility in task definition limits skill generalisability. Finally, the training data acquisition process and domain transfer are often impractical. In this thesis, individual solutions are proposed to address each of these issues.

In the first part, we find policy uncertainty to be an effective indicator of potential failure cases, in which the robot is stuck in out-of-distribution states. On this basis, we introduce a novel uncertainty-based approach to detect potential failure cases and a recovery strategy based on action-conditioned uncertainty predictions. Then, we propose to employ visual dynamics approximation to our model architecture to capture the motion of the robot arm instead of the static scene background, making it possible to learn versatile skill primitives. In the second part, taking inspiration from the recent progress in latent space planning, we propose a gradient-based optimisation method operating within the latent space of a deep generative model for motion planning. Our approach bypasses the traditional computational challenges encountered by established planning algorithms, and has the capability to specify novel constraints easily and handle multiple constraints simultaneously. Moreover, the training data comes from simple random motor-babbling of kinematically feasible robot states. Our real-world experiments further illustrate that our latent space planning approach can handle both open and closed-loop planning in challenging environments such as heavily cluttered or dynamic scenes. This leads to the first, to our knowledge, closed-loop motion planning algorithm that can incorporate novel custom constraints, and lays the foundation for more complex manipulation tasks.

Contents

Acronyms

2D	Two-Dimensional
3D	Three-Dimensional
AM	Activation Maximisation
AMP-LS	Activation Maximisaion Planning in Latent Space
BCE	Binary Cross-Entropy
BIT	Batch Informed Trees
BNN	Bayesian Neural Network
BVMC	Bayesian Visuomotor Control
CCE	Categorical Cross-Entropy
CEM	Cross-Entropy Method
CHOMP	Covariant Hamiltonian Optimisation for Motion Planning
CNN	Convolutional Neural Network
DAGGER	Dataset Aggregation
DoF	Degree of Freedom
DVBF	Deep Variational Bayes Filters
E2EVMC	End-to-End Visuomotor Control
ELBO	Evidence Lower Bound
ELU	Exponential Linear Unit
FC	Fully Connected
FCL	Flexible Collision Library
FK	Forward Kinematics
FMT	Fast Marching Trees
GAIL	Generative Adversarial Imitation Learning
GECO	Generalised ELBO with Constrained Optimisation
GEECO	Goal-Conditioned End-to-End Control

IBVS	Image-Based Visual Servoing
i.i.d.	Independent and Identically Distributed
IK	Inverse Kinematics
IL	Imitation Learning
KL	Kullback-Leibler
KPIECE	Kinodynamic Motion Planning by Interior-Exterior Cell Exploration
L2RRT	Learned Latent RRT
LBKPIECE	Lazy Bi-Directional KPIECE
LSPP	Latent Space Path Planning
LSTM	Long Short-Term Memory
MAML	Model Agnostic Meta-Learning
MLP	Multi-Layer Perceptron
MPC	Model-Predictive Control
MSE	Mean Squared Error
OMPL	Open Motion Planning Library
PBVS	Position/Pose-Based Visual Servoing
PCA	Principal Component Analysis
PID	Proportional-Integral-Derivative
PlaNet	Deep Planning Network
PRM	Probabilistic Roadmap
ReLU	Rectified Linear Unit
RGB	Red, Green, Blue
RGB-D	Red, Green, Blue-Depth
RL	Reinforcement Learning
RRT	Rapidly-Exploring Random Tree
SAVP	Stochastic Adversarial Video Prediction
SGD	Stochastic Gradient Descent
SQIL	Soft Q Imitation Learning
STOMP	Stochastic Trajectory Optimisation for Motion Planning
TecNet	Task-Embedded Control Network

TrajOpt	Trajectory Optimisation
VAE	Variational Autoencoder
VFS	Visual Foresight
VMC	Visuomotor Control

1 Introduction

1.1 Motivation

Robot manipulation, the capability of interacting with the objects around and effecting change on the world, is a key characteristic that sets robots apart from other automated and computerised systems. The world constantly evolves and the environment is often unpredictable, having a system that learns to adapt to the environmental changes is thus of paramount importance. A key learning challenge in manipulation is learning skill policies [53]. The goal of a skill policy is for the robot to accomplish certain objectives, such as reaching a target, pushing a cube, picking up a stick, etc.

Among all possible policy learning schemes, visuomotor control (VMC) [60] is an effective means of learning to perform manipulation tasks from demonstration trajectories of raw visual sensor data. Learning from demonstrations has the advantages of describing the task or user preference that may be hard to be specified or encoded programmatically; bypassing exploration or reward shaping that would usually be required in a reinforcement learning setting that could be computationally expensive or have safety concerns. Recent advances in VMC have achieved impressive feats across a range of tasks [33, 65, 90]. In particular, it has been demonstrated to

achieve remarkable performance in pick-and-place, a fundamental building block of more complex manipulation tasks, with successful zero-shot sim-to-real transfer [42].

Despite its multiple strengths, VMC also has its own limitations as listed in the following. (a) The issue of distribution shift, which refers to the phenomenon in supervised learning when the data a model is trained on changes over time causing the prediction to be less accurate, is common in imitation learning [79]. Standard approaches greedily imitate demonstrated actions without reasoning about the consequences of those actions. During training, the policy has only seen expert demonstrations, while during execution, very often due to noise in the environment and compounding errors, it may drift into out-of-distribution states and does not know how to return to demonstrated states. (b) Policies are designed and trained to perform tasks that are narrowly defined, resulting in a lack of versatility. For example, in the controllers of [42, 105], putting a red cube into a blue basket is a different task than putting a yellow cube into a green basket. Although these two controllers have much in common, they need to be trained separately and using representative data. This approach is inefficient and more importantly, it is impossible to cover an infinite amount of task variations. (c) VMC typically requires a significant amount of data. However, expert demonstrations required as training data may be hard to acquire. Collecting human demonstrations in the real world may be time-consuming, while relying on an expert policy to collect demonstrations in simulation may limit the complexity of task definition. An expert policy in simulation usually relies on state-based information and it is generally hard to design one for tasks that require more than just getting into designated positions and closing or opening the gripper, e.g., water pouring, door opening, in-hand manipulation.

While VMC is effective in learning different skills, it merely generalises from what it has seen in the training data and cannot reason about obstacles and plan paths around them. Indeed, another key challenge in robotics is that of motion planning, which consists of finding a sequence of valid joint configurations moving the robot from an initial to a target configuration, typically in the presence of obstacles and subject to robot joint limits and velocity and torque constraints. It is

a fundamental skill in manipulation. For instance, given a task of grasping an object, the first step would be to perform motion planning to reach that object. Traditional approaches [57] are limited in a number of ways. First, as the complexity of the configuration increases (i.e., Degrees of Freedom (DoF) of the robot, environmental complexity), the decreasing efficiency of traditional approaches makes reactive behaviours challenging. While existing approaches [22, 46] can find solutions to motion planning problems, their planning time scales super-linearly with the DoF of the robot and the scene complexity [62]. Requiring extensive computational power limits the applications of traditional planning approaches in time-sensitive scenarios, e.g., in adapting to changes in the environment. Second, traditional approaches are often restricted in their ability to meet multiple constraints simultaneously. On the one hand, optimisation-based planners are often not designed to handle constraints that cannot be expressed directly in joint space [73]. On the other hand, sampling-based planners struggle to find solutions in scenarios where constraints render only a small volume of configurations feasible [4, 58]. To sum up, motion planning in high-dimensional space with a variety of constraints remains an open challenge.

A promising direction in learning-based approaches to motion planning attempts to build a statistical model of a robot system from observations [6]. Conceptually, the statistical approach provides a simple way to collect training data without requiring extensive demonstrations. Recent advances in deep generative modelling [48] offers a powerful technique to unsupervised learning of a compact internal representation of high-dimensional data. We posit that a latent-variable deep generative model is able to capture the correlations between variables of interest and the coordination of complex high degree-of-freedom robot systems for motion planning.

1.2 Guiding Questions

In this thesis, we focus on four guiding questions that address the main limitations mentioned above. The guiding questions that we propose are the following:

1. Can a visuomotor control model detect potential task failures before the end of a policy rollout and recover from them?

2. Can a visuomotor control model learn versatile skills from visual demonstrations that can be generalised to different tasks?

3. Can a generative model capture an accurate model of forward kinematics and learn kinematics constraints for motion planning?

4. Can a generative model incorporate environmental constraints from sensor inputs (e.g., point clouds) and handle multiple constraints for motion planning in challenging environments, such as complex static scenes or dynamic scenes with moving obstacles and a moving target?

More specifically, we aim to show that (a) by adopting a probabilistic approach to VMC, we can predict policy uncertainty, which can be further utilised to guide the robot to return to in-distribution states seen in training; (b) by introducing a novel architecture for goal-conditioning, we can learn versatile skills that are general enough to solve tasks with different object colours or shapes; (c) a novel approach to motion planning that does not rely on expert demonstrations bypasses traditional computational challenges while achieving commensurate performance to established motion planning algorithms. Through the use of a latent-variable generative model in motion planning, we can embed correlations between state space variables into a structured latent space. By traversing the latent space through optimisation, we can generate valid motion plans while simultaneously satisfying multiple constraints, in particular target reaching, end-effector orientation constraint, and obstacle avoidance.

1.3 Thesis Outline

To put the topic of the thesis into perspective, we begin our investigation by providing context and reviewing existing literature in robot learning for manipulation in chapter 2. We first give an introduction to robot manipulation and how learning-based approaches come into play, which together form the broad theme of the thesis. Then, we delve into individual learning challenges that encompass most manipulation

problems. After surveying the broad challenges, we dedicate individual sections to a focused literature review of visuomotor control and latent space planning for manipulation, which are the central themes of this thesis.

Thereafter, in chapter 3, we present concept preliminaries essential to the development of the thesis. More specifically, we lay out the frameworks and conventions adopted to enable end-to-end visuomotor control, uncertainty estimation in neural networks, and deep generative modelling, which form the basis of the subsequent chapters.

In the remainder of this section, we provide an overview of individual chapters from chapter 4 to 7 and how they address each guiding question. Each of the subsections aims to outline the motivation, contribution and conclusion of chapters 4 to 7. An overview of the key contributions in this thesis is illustrated in fig. 1.1.

Finally, chapter 8 revisits our guiding questions, summarises our findings and contributions, reflects critically on our own limitations, and provides an outlook for future research directions building upon the contributions of this thesis.

1.3.1 Visuomotor Failure Recovery using Policy Uncertainty Prediction

Visuomotor control is a special case of sequence prediction, which consists of predicting a sequence of actions based on visual and proprioceptive observations. A typical approach in imitation learning is to train a regressor to predict an expert's behaviour given training data of the encountered observations and actions from the expert demonstrations. However, during the execution of a learned policy, the current action impacts future observations and actions, violating the standard i.i.d. assumption in most statistical learning problems. Faced with the issue of distribution shift, DAGGER [79] uses an expert/supervisor to provide corrective labels at each iteration under the current policy and trains the next policy under the aggregate of all collected datasets to counter compounding errors. More recent approaches GAIL [36] and SQIL [75] address this issue by training a reinforcement

learning agent to match the demonstrations and providing an incentive to encourage the agent to return to demonstrated states respectively.

In chapter 4, we provide an alternative solution by investigating when a learned policy is likely to be in an out-of-distribution state and designing an effective failure recovery strategy. We posit that the policy uncertainty can indicate a potential failure. In order to estimate the policy uncertainty, we adopt a probabilistic approach [24] to the existing neural network architecture that consists of adding dropout layers [86]. Drawing Monte-Carlo samples from a neural network with dropout provides us with an uncertainty estimate. Afterwards, we analyse the correlation between uncertainty and task failure and propose a threshold for recovery that is optimal for our purpose. We further introduce a recovery strategy involving taking the action that leads to minimum uncertainty. Simply knowing the uncertainty of an action after it has been executed is not enough to find the action leading to minimum uncertainty. Therefore, we learn a model mapping the current feature embedding and action to the future uncertainty.

The experimental results show that task success rate is inversely correlated with uncertainty, providing an empirical grounding of using uncertainty as an indicator of how well a model performs. Having verified our hypothesis, we evaluate our proposed recovery strategy against several other recovery baselines in simulated pushing, pick-and-place, and pick-and-reach tasks. In all three tasks, our approach outperforms baselines and exhibits interesting behaviours: after detecting high uncertainty, the robot switches to a recovery strategy and recovers from a potential failure. This framework holds a tantalising prospect of endowing existing deterministic policies with the capability of recovering from failures by simply adopting a Bayesian approach for uncertainty estimation.

1.3.2 Versatile Visuomotor Skill Primitives via Dynamic Representations

In the literature of goal-conditioned visuomotor control for manipulation, the versatility appears to be a major limitation. Common approaches to goal-conditioning

mainly build on Model-Predictive Control (MPC) [14] and few-shot imitation learning [13, 21]. An established line of MPC work on Deep Visual Foresight [14, 20, 68, 98] learns an action-conditioned video predictor, samples action sequences, predicts outcomes according to a video predictor, and finally chooses the best action sequence under a specific goal distance metric. However, the imprecision of the forward model compounds over long time horizons and the sampling process can be computationally expensive, limiting the applications to simple pushing or placing tasks. Few-shot imitation learning approaches on the other hand, learn general task representations which are quickly adaptable to unseen scene setups. They typically learn a visuomotor control policy of a variety of tasks during training time, which needs to be fine-tuned on a few demonstrations of a novel task during test time. Nevertheless, such new demonstrations may not be easily accessible, limiting their deployability. Unlike those methods, in chapter 5, we propose a visuomotor controller named GEECO that can adapt to a new task via a single goal image and without the need for additional demonstrations for fine-tuning.

In the approach section, we inquire how GEECO extends end-to-end VMC architecture [42] to incorporate goal-conditioning. More specifically, by leveraging dynamic images to represent the dynamics of an entire frame sequence in a single image, it captures the current motion of the robot arm and focuses on the difference in location and geometry of objects instead of the static scene background. Consequently, the downstream network makes use of this representation to make control command predictions more effectively and without the need of additional supervision labels for tasks with a variety of object properties. In the experiments section, we compare the performance of our proposed model against two representative baselines of visual MPC and one-shot imitation learning, and demonstrate its efficacy in complex goal-conditioned pushing and pick-and-place tasks. Furthermore, we show that our method transfers well to challenging, unseen environments with heavy clutter, visual distortions or novel object geometries, accomplished without the need of domain randomisation which is commonly

employed to make VMC more robust. All in all, GEECO offers a novel approach to learning data-efficient and versatile skill primitives for manipulation.

1.3.3 Motion Planning Through Latent Space with Primitive Shapes

As discussed earlier in this chapter, traditional motion planning approaches are confronted with computational challenges and struggle to specify or handle multiple constraints simultaneously. Common approaches to real-time re-planning mainly build on Potential Field methods [22, 46]. While the potential field principle is widely used, substantial shortcomings that are inherent to this principle have been identified in [50]. In comparison to our work, they struggle to handle additional constraints on properties that cannot be fully determined by robot joint configuration.

With the advance of deep learning, learning-based approaches offer a new perspective to motion planning with the potential of bypassing existing limitations. One possible approach to motion planning is learning inverse kinematics (IK), which can provide a solution of the joint configuration for the goal that can be further used to plan a path. As a highlight from the literature, Bócsi et al. [6] offer a different view to motion planning and to deep learning by building a statistical model of robot system from experience. In greater detail, they attempt to apply a kernel-based density estimation to learn a joint distribution between end-effector positions and joint angles. By leveraging this joint distribution, a solution is proposed to an ill-posed IK problem for robot arms with redundant degrees of freedom (DoF). Inspired by [66, 95], we posit that by employing a deep generative model, we can learn a structured latent representation capturing the kinematics relationship that can be useful for downstream motion planning tasks.

We propose to learn a generative model of robot states by using a variational autoencoder (VAE) and a neural network-based high-level performance predictor to incorporate desired constraints. More specifically, we train a collision predictor for collision avoidance. The iterative optimisation process in the latent space translates to a sequence of joint angles, generating a motion plan. The experimental

results are presented using a 7-DoF panda arm in a scenario with simple primitive shaped colliders in both simulated environment and the real world. Our approach achieves performance in line with established traditional approaches in terms of reaching success rate, and outperforms most baselines in terms of planning time. The improvement of the planning time stems from the design of our approach, in that it only performs a forward and backward pass of the neural network at every planning step, irrespective to the complexity of the scene setup. It is our belief that this generative approach is an interesting alternative to explore for motion planning.

1.3.4 Generalisation of Motion Planning from State-Based Observations to Complex Scenes

Prior work [38] introduces motion planning through learning a structured latent space of kinematically feasible robot states and optimisation in such latent space for planning. However, the collision predictor relies on state-based inputs consisting of the positions and sizes of the primitive shaped colliders, limiting its applicability to complex scenes in the real world. In contrast, we learn a collision predictor mapping the point cloud of the scene and the structured latent representation to collision probability. Additionally, we incorporate an orientation constraint of the end-effector by taking it into account in the generative model. The orientation constraint can be useful when trying to reach a pre-grasp pose. Algorithmically, we improve the multi-constraint optimisation process in prior work [38] by performing explicit collision checking before execution of a joint configuration and rescaling the coefficients of different losses to deviate from the original route and focus on collision avoidance, resulting in fewer collision cases. For evaluation, we compare our method against several representative motion planning baselines in a complex simulated tabletop environment of a variety of objects and demonstrate its efficacy and robustness by achieving competitive reaching success and much improved planning time. We further illustrate the reactive behaviour of our method in a challenging scenario of reaching a moving target while avoiding a moving obstacle in both the simulated and the real-world environments. In short, we are able to

carry out motion planning in high-dimensional space from observation-based inputs while satisfying multiple constraints simultaneously.

1.4 Publications

The following publications, listed in their order of appearance, form the main body of chapter 4 to 7.

1. **Chia-Man Hung**, Li Sun, Yizhe Wu, Ioannis Havoutis, Ingmar Posner. "Introspective Visuomotor Control: Exploiting Uncertainty in Deep Visuomotor Control for Failure Recovery". In: *IEEE International Conference on Robotics and Automation (ICRA)*. June 2021. [37]

2. Oliver Groth, **Chia-Man Hung**, Andrea Vedaldi, Ingmar Posner. "Goal-Conditioned End-to-End Visuomotor Control for Versatile Skill Primitives". In: *IEEE International Conference on Robotics and Automation (ICRA)*. June 2021. [28]

3. **Chia-Man Hung**, Shaohong Zhong, Walter Goodwin, Oiwi Parker Jones, Martin Engelcke, Ioannis Havoutis, Ingmar Posner. "Reaching Through Latent Space: From Joint Statistics to Path Planning in Manipulation". In: *IEEE Robotics and Automation Letters (RA-L)*. Feb. 2022. [38]

4. Jun Yamada*, **Chia-Man Hung***, Jack Collins, Ioannis Havoutis, Ingmar Posner. "Leveraging Scene Embeddings for Gradient-Based Motion Planning in Latent Space". Under review at: *IEEE International Conference on Robotics and Automation (ICRA)*. June 2023. * Equal contribution.

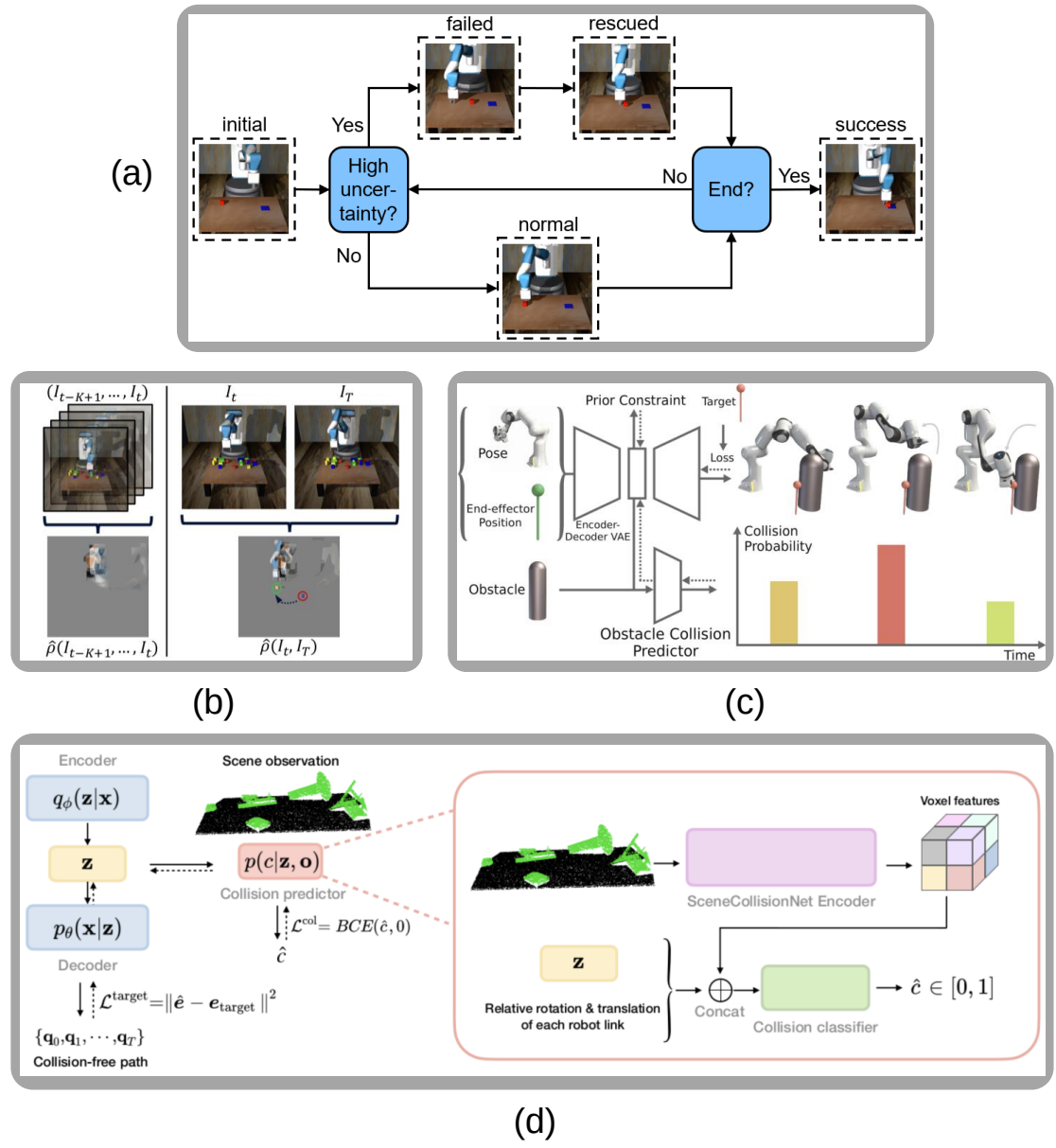


Figure 1.1: An overview of the key contributions in this thesis. (a) In chapter 4, we employ a probabilistic approach to predict policy uncertainty, which is used to guide the robot to enter a *failure recovery* mode and rescue it from *out-of-distribution* states. (b) GEECO (chapter 5) is a *goal-conditioned* visuomotor control trained in an end-to-end manner. It learns an approximation of the *task dynamics* and transfers to unseen scenarios. (c) LSPP (chapter 6) is a novel approach to *motion planning* by learning the joint distribution of robot poses and end-effector positions in a generative model and iteratively performing back propagation in the *latent space*. (d) In chapter 7, AMP-LS extends LSPP to incorporate *orientation constraints* and trains an obstacle collision predictor based on point cloud scene observation to generalise motion planning to *complex scenes*. It is shown to handle both open and *closed-loop* planning.

2
Background

In this chapter, we present the context of robot learning for manipulation and an overview of the topics related to the central theme of the thesis: visuomotor control and latent space planning for robot manipulation. We start in section 2.1 by introducing robot manipulation and providing a summary of learning problems in manipulation. This section adheres to the categorisation of learning problems by Kroemer et al. [53]. Following the summary, a learning problem is introduced in more details in each subsection. We limit the subsections to only discussing the subcategories of learning problems that fall within the scope of our work. At the end of individual subcategories, we illustrate how our exploration and investigation fit into the larger picture. Afterwards, section 2.2 is dedicated to related work. We outline in section 2.2.1 background knowledge about visuomotor control on top of which the first half of the thesis is built. Finally, we conclude in section 2.2.2 by elaborating the evolution of latent space planning and the pieces of pioneer work that inspire the second half of the thesis.

2.1 Robot Learning for Manipulation

Robot manipulation refers to the study of ways robot systems interact with objects around them. Common tasks include reaching a target, grasping a bottle, stacking

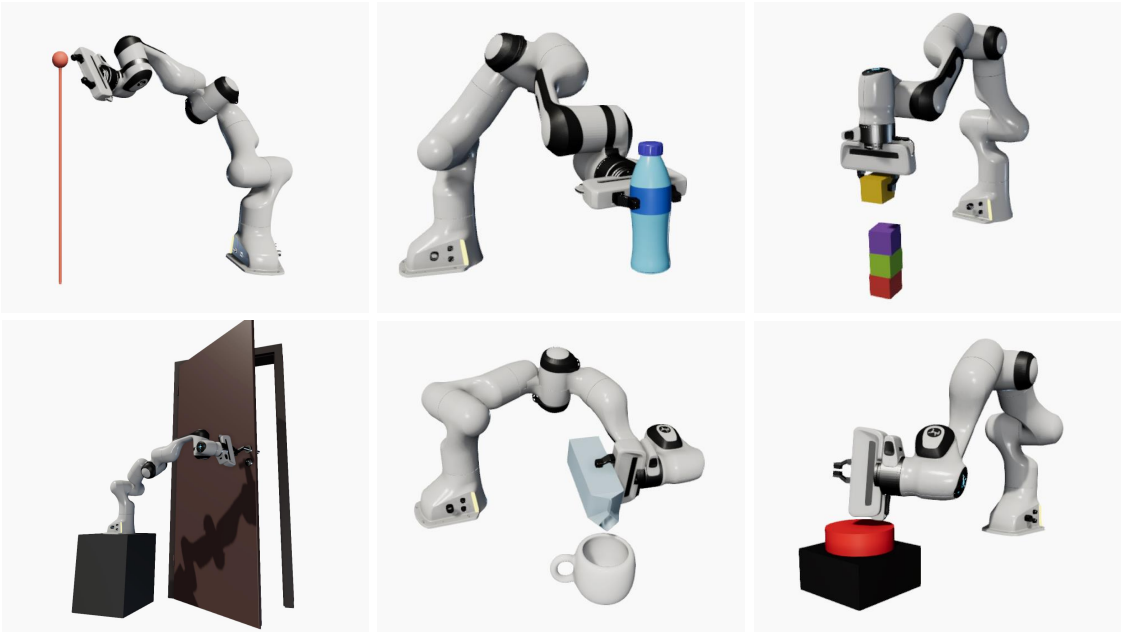


Figure 2.1: Example manipulation skills: reaching a target, grasping a bottle, stacking cubes, opening a door, pouring milk into a cup, pressing a button.

cubes, opening a door, pouring water, pressing a button, pushing a chair, inserting an item, etc (cf. fig. 2.1). A robot manipulator typically consists of a robot arm and an end-effector. The simplest end-effector option is perhaps a suction cup [84, 102], enabling a robot to perform simple pick-and-place tasks without the need of precise grasping. It may however be limited in more complex tasks, such as reorienting an object, inserting a peg, and pressing a button. More sophisticated end-effectors, for instance a mutli-finger hand, allow for more complex in-hand manipulation tasks [10, 100] like positioning a pencil, cutting with scissors, and using knife and fork. Throughout this thesis, we focus on parallel grippers, which strikes the balance between the simplicity of the end-effector and the range of tasks.

Manipulators are currently widely used in the industry, in particular the manufacturing industry [5]. Most of the time, arms are fixed to mounts on an assembly line, controlled by teleoperation [55] or pre-programmed sequences of commands using a mix of techniques namely object detection and open-loop motion planning. Traditional non learning-based approaches focusing on designing repetitive systematic behaviours may be suitable in precise manufacturing settings, but naturally fall short in scenarios with unsystematic environmental variations

in which neither the robot nor its creators have encountered before. A significant amount of research has been carried out on using learning-based approaches to solve robot manipulation tasks.

As described in [53], learning problems posed by manipulation tasks can be broadly placed into five categories: (a) learning object and environment representations; (b) learning a transition model of the environment; (c) learning motor skills; (d) learning to characterise motor skills; (e) learning compositional and hierarchical task structures. In the following, we provide more details on what each problem entails.

2.1.1 Object and Environment Representations

In manipulation, information about the environment and the objects within the environment is usually not given. Robots observe and sometimes even interact with the environment to acquire such knowledge. Learning object properties and selecting useful features can further aid the generalisation across the task family.

Object Representations. Robots interact with a world composed of objects that have different properties. Some objects are static, such as wall, floor and counter, while some are movable, such as box, cup, door and handle. To accomplish a task, a robot first needs to learn a representation of objects in its environment. Furthermore, this representation can potentially render learned skills versatile by transferring or adapting to objects with similar attributes.

What object representations need to capture depends on task variations. Broadly speaking, task variations can be categorised into within-task variations and across-task variations. While within-task variations capture features a manipulation action can change, such as object pose and object shape for deformable or divisible objects, across-task variations are attributes that are fixed in any specific task that can aid generalisation across tasks, such as material properties.

In both chapters 4 and 5, object representations are implicitly encoded by a neural network from the visual observations, and learned in an end-to-end manner. While chapter 4 only considers within-task variations, chapter 5 attempts to include

across-task variations, in particular object geometries and colours. By leveraging the dynamic images representation that helps the inference process retain the object positions of the current observation and the goal image, while ignoring all static parts of the scene, our proposed method is able to generalise across tasks.

In the context of motion planning, in chapter 6, we assume complete knowledge of the primitive shaped obstacles in the scene, represented by their positions and sizes. This representation limits the applications to complex scenes that cannot be easily factorised into primitive shapes. In chapter 7, we reason at a task-level and encode the entire scene from point cloud observation, enabling our approach to generalise to complex scenes.

Passive and Interactive Perception. Depending on whether physical interaction with the environment is involved, robot perception can be divided into passive perception and interactive perception. Passive perception describes non-interactive perception, for example localising and classifying an object from a camera image. In interactive perception, the robot physically interacts with the environment to acquire a better understanding of its surroundings, for example picking an object up to estimate its weight or touching objects for tactile sensing. This kind of perception often requires more time and energy to perform, but has the advantage of disambiguation between similar scenarios. Throughout this thesis, we perform passive perception with a static third-person camera mounted at an angle to observe the task scene.

Learning Object Properties. To learn about the objects in the scene, we are first tasked with distinguishing them, which is formulated as a segmentation problem. Once an object is identified, depending on what the robot wants to do with it, the robot may need to learn about interaction-relevant properties such as graspability and stackability; dynamics properties such as weight and centre of mass; or material properties such as size, shape, and friction coefficient. If an object has a more complex structure, it may be necessary to learn about its DoF

and kinematic chain. Again, in chapters 4 and 5, object properties are implicitly learned in an end-to-end manner.

Feature Learning and Selection. While having complete information of all object properties can be advantageous, when performing a manipulation task, often only a small set of features is relevant. Learning and selecting relevant features is a key aspect of learning useful object representations. Unsupervised learning approaches extract information from unlabelled data by capturing the correlations in the data while discarding noisy signals. They typically employ dimensionailty reduction or clustering techniques to filter out the most important information to a given task. Supervised approaches on the other hand, learn features as part of the model training process from existing data labels. In chapters 4 and 5, a deep neural network model is used to encode high-dimensional visual inputs as an alternative automatic method for feature learning and selection. In chapters 6 and 7, we incorporate architectural priors for learning useful features. More specifically, we employ a variational autoencoder to capture the correlation between joint configurations and end-effector poses.

2.1.2 Transition Models

To accomplish a task, a robot takes actions that change the state of the environment. Learning a transition model capturing the way states change in response to actions is a common approach in robot learning.

A transition model is a deterministic or stochastic distribution over the next states given the current state and action. It can also depend on some context state invariant to actions, separated from the state of the robot and the objects. Generally, in manipulation tasks, robots operate in continuous state space. Action space is often continuous, but can sometimes be discrete, e.g., opening or closing the gripper can be formulated as a binary action. Common continuous models include regression models, for instance neural networks, Gaussian processes [91], Gaussian mixture models [76], and linear regressions. Discrete transition models

are typically used to capture high-level tasks. Basic models include tabular models and finite state machine [67]. Hybrid models combine both continuous and discrete aspects of the transition model, often required in hierarchical tasks that involve planning high-level actions in a discrete model and using a low-level continuous model for accomplishing individual sub-tasks.

In chapters 4 and 5, our approach does not learn any transition model, but rather learns to predict the next action conditioned on the history by using a neural network-based regression model. In chapters 6 and 7, we do not learn any transition model either. Instead, we embed the states into latent representations, and plan a sequence of latent representations that translates to a motion plan.

2.1.3 Skill Policies

One vital component for a robot to perform a manipulation task is learning a skill policy (or skill controller). There are different types of action spaces and policies can be categorised by their parameterisation. Two major approaches to learning a skill policy are reinforcement learning and imitation learning. In reinforcement learning, a robot learns from its experience in the world, while in imitation learning, a robot learns from expert demonstrations. In this subsection, we only discuss imitation learning as it is more relevant to the following chapters. Once a skill is learned, there are various options for it to transfer to solve other tasks.

Action Spaces. From a low-level point of view, a robot needs to send a control signal to its actuators to physically perform actions. Robots that are difficult to model may benefit from sending direct control signals. However, in most cases, it is more practical to have an additional layer of controller between the policy and the actuator. A commonly used controller is a PID controller where the action space of the policy can be desired positions, velocities and accelerations in the joint space or for the end-effector in Cartesian space. In general, learning a skill policy in Cartesian space is easier as the dimension of the space is smaller. For example, in chapters 4 and 5, we learn a skill policy with actions defined as delta end-effector

position in Cartesian space. In chapters 6 and 7, our trajectories are represented as sequences of joint positions by the definition of motion planning.

Policy Structures. Parameterisation is often imposed on policies to encode a prior and improve data efficiency. However, there is a trade-off between representational power of the policy and data efficiency. If the restrictions imposed by the parameterisation does not respect the underlying structure of the task, it may significantly negatively impact the performance. The choice of parameterisation is thus an essential component of a policy.

Non-parametric policies are more expressive representations, but also less data efficient. Examples include Gaussian processes [72], locally-weighted regression, and Riemannian Motion Policies [74]. This category of polices is typically flexible and can be applied to manipulation tasks with little domain knowledge, but requires a large amount of data. Parametric policies, on the other hand, are restricted by their underlying representation. Common examples include neural networks and dynamic movement primitives [40]. The sample efficiency depends on the number of restrictions that are placed on the structure of the policy. In chapters 4 to 7, all our policies are represented by neural networks. Our VMC neural networks are highly parameterised (i.e., using a large amount of parameters), leading to higher degree of generalisation, at the cost of requiring more data.

Imitation Learning. Imitation learning is a form of learning from demonstration trajectories. Demonstrations are typically collected in the real world by teleoperation [103] (where the robot is controlled remotely by a human) or kinesthetic teaching [51] (where the robot is physically guided through the task by a human), and in simulation by teleoperation [55] or through the use of a hand-designed expert policy [42]. Behavioural cloning [89] is perhaps the most straight-forward way of leveraging demonstrations to learn a skill policy. The demonstrations provide state-action pairs that can be used as training data to learn the policy parameters that reproduce the demonstrated behaviours. In the context of manipulation, in chapters 4 and 5, our controllers are built upon End-to-End

Visuomotor Control (E2EVMC) [42]. E2EVMC is in essence a type of behavioural cloning that continuously maps a sliding window of past images and proprioceptive features to the next actions and some auxiliary outputs (cf. section 3.1).

Skill Transfer. In our earlier example of a task definition, a parameterised skill of putting a red cube into a blue basket may have parts that are in common with another skill of putting a yellow cube into a green basket. It may be desirable to have a task context parameter that is changing depending on the specific task to facilitate skill transfer. This task context parameter may be tuned when given a new task rather than training a new skill policy from scratch.

Rather than reusing a skill, meta-learning learns a distribution of tasks. When given a particular task in that distribution in the future, it can be learned more efficiently. A representative work Model Agnostic Meta-Learning (MAML) [19] learns easily adaptable model parameters through gradient descent, such that it can solve new learning tasks using only a small number of training samples.

Collecting demonstrations and training policies in simulation is often easier than doing it directly on a real physical robot. However, there may be noise or inaccuracies that are not entirely captured by the simulation. When it comes to deploying in the real world, domain adaptation techniques are often required. To bridge the gap from sim to real, a common technique is domain randomisation, which randomises environmental attributes, such as dynamics in simulation, texture of the objects, to make the system invariant to those changes. E2EVMC [42] employs domain randomisation technique to transfer pick-and-place tasks from sim to real successfully.

2.1.4 Characterising Skills by Preconditions and Effects

Before a skill policy is applied, the robot needs to understand the circumstances under which it can be executed. Preconditions is a term used to describe a model of those circumstances. Similarly, after a skill is executed, the robot needs to know whether it has achieved the desired outcomes. Postconditions describes a

model of those outcomes. Preconditions and postconditions can be used jointly to sequence skills for planning a long task as a sequence of sub-tasks. In the tasks that we consider throughout the thesis, we assume that the initial task state already satisfies the preconditions and that we can directly observe whether an execution is successful or not.

2.1.5 Compositional and Hierarchical Task Structures

Manipulation tasks sometimes have hierarchical structures and it can be helpful to decompose them into sub-tasks. Decomposing tasks has the advantage of breaking long tasks into shorter sub-tasks that can be solved with individual skills. For example, if a task consist of multiple pick-and-place tasks, solving it with individual pick-and-place skill makes the entire task more manageable and allows for the reuse of the same skill. Although it is also an important learning problem in manipulation, it is out of scope as the tasks that we consider do not have this hierarchical task structure.

2.2 Related Work

In this section, we present work related to the various aspects of visuomotor control and latent space planning addressed in this thesis.

2.2.1 Visuomotor Control for Robot Manipulation

In the previous section, we have described the main learning challenges in robot manipulation. Among those challenges, we concentrate on learning motor skills in combination with learning object and environment representations. In neuroscience, "visuomotor integration is the coordination of neuronal activity between visual-related and motor-related parts of the brain in order to influence behavior and perception," as defined in [83]. Similarly, in robotics, we regard visuomotor control as the coordination between vision sensors and motion control for the completion of a task. Based on the current visual inputs from vision sensors, the robot computes and executes motion commands, which in turn gives new visual feedback.

Early work on visual servoing focuses on using real-time feedback from vision sensors to control a robot's motion and does not consider long-term planning. In order to accomplish a task, it attempts to match the visual input to a target image. Model-predictive control plans robot motion from visual feedback by learning a forward model of the world, which forecasts the outcome of an action. Actions are sampled and selected to match the predicted outcome to the goal. Another line of work learns from demonstrations and regresses the current visual input to the next action. In the following, we survey those three established approaches to visuomotor control.

Visual Servoing. Closed-loop control of a robot using visual feedback is referred to as visual servoing, and was first introduced in [34] to distinguish it from earlier work on open-loop robot control. Position/Pose-Based Visual Servoing (PBVS) [2, 12, 93, 97] typically derives the current pose of the robot end-effector from a depth image and generates an ideal pose for the object of interest. The error is then formulated as the difference between the two poses and is updated at each frame. PBVS allows easier trajectory planning, e.g., obstacle avoidance, as it is controlled directly in Cartesian space. However, PBVS is not very often adopted in servoing tasks because it relies on a model to generate a target pose and most importantly, it requires precise system calibration of the camera and between the camera and the robot for accurate pose estimation. Unlike PBVS, Image-Based Visual Servoing (IBVS) [8, 32, 63] extracts image features from visual observations, and formulates the error function in 2D image plane. IBVS is considered to be robust to system calibration errors, but typically relies on hand-crafted image features for object detection. A comprehensive survey on visual servoing is provided by Kragic et al. [52]. Although visual servoing methods are able to adapt to dynamic environments, the complexity of a manipulation task can easily exceed the capabilities of visual servoing. The manipulation of objects typically involves object recognition, servoing onto the object, grasping, placement of the object; visual servoing only solves part of the task.

Model-Predictive Control. To plan and control the robot motion, the capability of predicting the effects in state or image space is required. Conventional planning methods work on robot state space without consideration of the physical interaction between the end-effector and the object. To plan robot motion from visual feedback, an established line of research is to use visual model-predictive control. The idea is to learn a forward model of the world, which forecasts the outcome of an action. In the case of robot control, a popular approach is to learn the state-action transition models in a latent feature embedding space, which are further used for motion planning [1, 92, 101]. Likewise, visual foresight [20] leverages a deep video prediction model to plan the end-effector motion by sampling actions leading to a state which approximates the goal image. Afterwards, visual foresight is adapted to visual navigation by driving the robot through topological key-frame images [35]. However, visual model-predictive control relies on learning a good forward model, and sampling suitable actions is not only computationally expensive but also requires finding a good action distribution.

End-to-End Imitation Learning. Deep learning has been successfully used in the visual guidance for robot grasping. Viereck et al. [90] integrate deep grasping detection with end-effector control to close the loop of perception and grasping. Then, deep neural networks are used as policy network for imitation learning tasks to infer robot motor commands from the visual input. Levine et al. [60] were the first to employ an end-to-end trained neural network to learn visual motor skills from admittance control examples, yet their approach requires months of training and multiple robots. In end-to-end visuomotor control [42], a similar network architecture is used for joint velocity control for a specific multi-stage task, and domain randomisation through generating procedural textures is used to achieve a zeros-shot sim-to-real adaption. Now, end-to-end imitation learning has been employed to address more challenging tasks, such as autonomous driving [69] and in-hand manipulation [41].

2.2.2 Latent Space Planning for Robot Manipulation

Latent space planning addresses manipulation tasks by planning from latent dynamics or planning as optimisation in latent space. It should be noted that if it leverages visual observations, it can be regarded as visuomotor control in the sense that it coordinates vision inputs and motion control. Nonetheless, we separate it in a new subsection as it describes a new concept.

Planning from Latent Dynamics. The idea of learning embedding spaces from high-dimensional space and then learning forward dynamics models operating on this learned latent space for planning has been explored in the literature. L2RRT [39] learns latent dynamics and a collision predictor conditioned on adjacent latent representations, and applies a sampling-based motion planning algorithm on the learned latent space to plan a collision-free trajectory. Universal Planning Networks [85] learn representations of high-dimensional observations and optimise a global motion plan from the unrolling of a learned forward model by gradient descent. The Embed-to-Control works [3, 92] perform optimal control in the latent space of a deep generative model in which the dynamics is constrained to be locally linear. Based on the previous models, DVBFs [45] further impose a Markovian assumption on the latent space and leverage a variational inference mechanism to scale to large datasets. A recent line of work combines the latent dynamics model with reinforcement learning. PlaNet [31] performs model-based RL in a learned latent dynamics model for planning. Dreamer [30] uses an actor-critic method to learn a parametric policy by propagating gradients of multi-step values back through learned latent dynamics. However, models that plan from latent dynamics resolve planning tasks primarily by rolling out trajectories in time, finding a promising trajectory, and performing the given actions. In this respect, these approaches tend to become intractable for longer time horizons. Most recently, Director [29] learns hierarchical behaviors by compressing goal representations and planning in the latent space of PlaNet's world model.

Planning as Optimisation in Latent Space. Instead of planning from latent dynamics, planning directly in latent space through Activation Maximisation is an emerging research area. Activation Maximisation was first introduced in [18] as a means to visualise higher-layer features of a deep neural network. A remarkably different use was put forward when this concept was later adopted for 3D tool synthesis [95] and quadruped locomotion [66]. The idea is to frame planning as an optimisation process in the structured latent space of a deep generative model. The optimisation is driven by constraints specified by task-dependant performance predictors. By decoding the latent representations, a path in the latent space is then translated to a path in tool state space in [95] or robot joint space in [66]. The nature of this approach makes it suitable for both open and closed-loop planning. In chapters 6 and 7, we adopt this concept for motion planning.

3 Preliminaries

In this chapter, we present concept preliminaries that are at the heart of the development of this thesis. The first half of our work primarily extends end-to-end visuomotor control and the second half employs a deep generative model. Uncertainty estimation enables failure recovery in chapter 4.

3.1 End-to-End Visuomotor Control

End-to-End Visuomotor Control (E2EVMC) [42] learns a skill policy from trajectory demonstrations in a supervised manner. The model as shown in fig. 3.1 continuously maps a sliding window of size K of past visual inputs i.e., RGB images $\{I_{t-K+1}, \ldots, I_t\}$ and proprioceptive features i.e., joint angles $\{\mathbf{x}_{t-K+1}, \ldots, \mathbf{x}_t\}$ to the next joint velocity command $\hat{\mathbf{u}}_J$ and the next discrete gripper action $\hat{\mathbf{u}}_{GRP}$ (open, close or no-op) as well as the end-effector position $\hat{\mathbf{q}}_{EE}$ and object position $\hat{\mathbf{q}}_{OBJ}$ as auxiliary targets with the following loss objective:

$$\mathcal{L}_{\text{total}} = \text{MSE}(\hat{\mathbf{u}}_J, \mathbf{u}_J) + \text{CCE}(\hat{\mathbf{u}}_{GRP}, \mathbf{u}_{GRP}) + \text{MSE}(\hat{\mathbf{q}}_{EE}, \mathbf{q}_{EE}) + \text{MSE}(\hat{\mathbf{q}}_{OBJ}, \mathbf{q}_{OBJ}), \tag{3.1}$$

where MSE and CCE represent mean squared error and categorical cross-entropy respectively. The model is trained end-to-end with stochastic gradient descent.

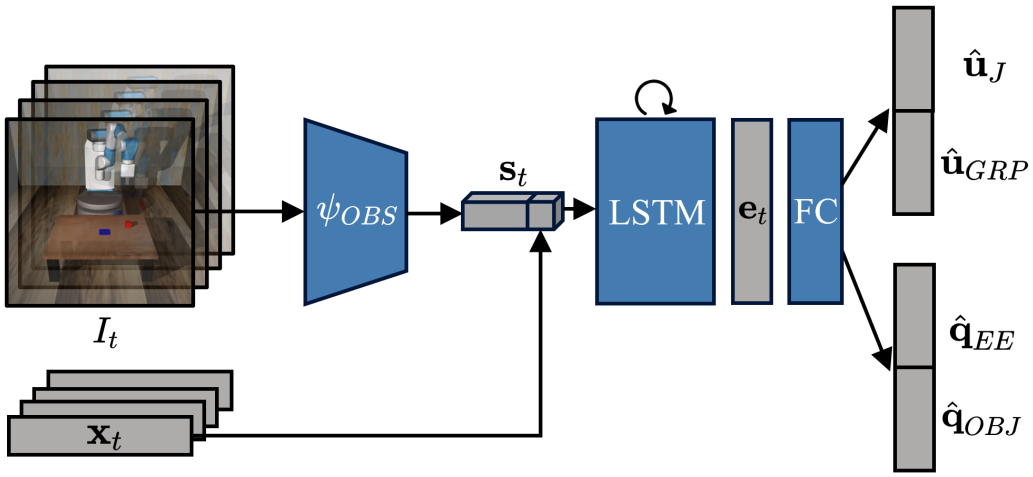


Figure 3.1: E2EVMC [42] model architecture. The current visual observation I_t is passed through a CNN ψ_{OBS} and concatenated with the proprioceptive feature $\mathbf{x}_t$ to form the current state representation $\mathbf{s}_t$. This state representation $\mathbf{s}_t$ is then fed into an LSTM to obtain the current embedding $\mathbf{e}_t$. The LSTM embedding $\mathbf{e}_t$ is finally passed through a fully connected layer and decoded into action commands $\hat{\mathbf{u}}_J$ and $\hat{\mathbf{u}}_{GRP}$, as well as auxiliary position predictions $\hat{\mathbf{q}}_{EE}$ and $\hat{\mathbf{q}}_{OBJ}$.

In our work, we use delta end-effector position command rather than joint velocity command $\hat{\mathbf{u}}_J$ as a model output. The dimensionality reduction of action command slightly simplifies the policy learning and we have found this to be less prone to the accumulated error over a long time horizon.

3.2 Uncertainty Estimation in Deep Learning

Neural networks have been widely applied to real-world applications in different fields, yet basic neural networks do not provide reliable uncertainty estimates for their decisions. Such inability often results in overconfident or underconfident predictions, thus making it difficult to trust the outcomes. This section first identifies different types of uncertainty and then introduces a Bayesian approach for uncertainty estimation in neural networks.

3.2.1 Types of Uncertainty

In probabilistic modelling, it is essential to distinguish between two types of uncertainty: aleatoric uncertainty and epistemic uncertainty. Aleatoric uncertainty

refers to the notion of randomness. For example, when we flip a coin or roll a dice, there is equal probability to each outcome (assuming a fair coin or dice). That is, the outcome of an experiment is due to inherent randomness (atmospheric noise) that cannot be reduced by any additional source of information. In contrast, epistemic uncertainty refers to uncertainty caused by a lack of knowledge. For example, if a robot attempting to open a door does not have information about whether the door is locked, its prediction is uncertain due to its lack of knowledge. In chapter 4, we adopt a Bayesian approach that explicitly captures the epistemic uncertainty in the model predictions by representing probability distributions over different models.

3.2.2 Bayesian Modelling

Given training inputs $\mathbf{X} = \{\mathbf{x}_1, \ldots, \mathbf{x}_N\}$ and their corresponding outputs $\mathbf{Y} = \{\mathbf{y}_1, \ldots, \mathbf{y}_N\}$, we would like to find the parameters θ of a function $\mathbf{y} = f_\theta(\mathbf{x})$ that is likely to have generated the data. In Bayesian modelling, we assume a *prior* distribution over the space of parameters $p(\theta)$. This represents the prior belief we have on our model before any data point is observed. We also define a *likelihood* distribution $p(\mathbf{y}|\mathbf{x}, \theta)$ as the probability distribution of outputs given an input and some parameter setting θ.

Given a new input point $\mathbf{x}^*$ and the training data $\mathbf{X}, \mathbf{Y}$, we perform the *inference* process to predict a new output $\mathbf{y}^*$:

$$p(\mathbf{y}^*|\mathbf{x}^*, \mathbf{X}, \mathbf{Y}) = \int p(\mathbf{y}^*|\mathbf{x}^*, \theta)p(\theta|\mathbf{X}, \mathbf{Y})d\theta. \tag{3.2}$$

In this integration, the *posterior* distribution $p(\theta|\mathbf{X}, \mathbf{Y})$ can be further decomposed by applying Bayes' theorem:

$$p(\theta|\mathbf{X}, \mathbf{Y}) = \frac{p(\mathbf{Y}|\mathbf{X}, \theta)p(\theta)}{p(\mathbf{Y}|\mathbf{X})}. \tag{3.3}$$

The normaliser in eq. (3.3) is called *model evidence*:

$$p(\mathbf{Y}|\mathbf{X}) = \int p(\mathbf{Y}|\mathbf{X}, \theta)p(\theta)d\theta. \tag{3.4}$$

This term is also called *marginal likelihood* as this integration marginalises the likelihood over θ. The integral in eq. (3.4) is in general intractable and approximation techniques are typically applied.

3.2.3 Variational Inference

Among different approximation techniques, variational inference approaches approximate the posterior distribution by optimising over a family of tractable distributions. We define an approximating *variational distribution* $q(\theta)$ and we would like it to be as close as possible to the posterior distribution $p(\theta|\mathbf{X}, \mathbf{Y})$. Kullback–Leibler (KL) divergence is commonly used to measure the similarity between two probability distributions, and is thus adopted here:

$$\mathrm{KL}(q(\theta) \,||\, p(\theta|\mathbf{X}, \mathbf{Y})) = \int q(\theta) \log \frac{q(\theta)}{p(\theta|\mathbf{X}, \mathbf{Y})} d\theta. \tag{3.5}$$

Due to the posterior distribution of which the integral of the normaliser is in general intractable to compute, the KL divergence cannot be optimised directly. An *evidence lower bound* (ELBO) is optimised instead:

$$\mathcal{L}_{\mathrm{ELBO}} = \int q(\theta) \log \frac{p(\mathbf{Y}|\mathbf{X}, \theta)}{q(\theta)} d\theta, \tag{3.6}$$

and the following holds:

$$\mathrm{KL}(q(\theta) \,||\, p(\theta|\mathbf{X}, \mathbf{Y})) = -\mathcal{L}_{\mathrm{ELBO}} + \log p(\mathbf{Y}|\mathbf{X}). \tag{3.7}$$

We denote $q^*(\theta)$ as a minimum of the KL divergence in eq. (3.7). This optimisation allows us to approximate the predictive distribution (cf. eq. (3.2)):

$$q^*(\mathbf{y}^*|\mathbf{x}^*) = \int p(\mathbf{y}^*|\mathbf{x}^*, \theta) q^*(\theta) d\theta. \tag{3.8}$$

3.2.4 Bayesian Neural Networks

Bayesian Neural Networks (BNNs) frame neural networks as Bayesian models by inferring distributions over the networks' weights. Monte Carlo Dropout casts existing stochastic elements of deep learning as variational inference by formulating dropout layers as Bernoulli distributed random variables, so that training

a neural network with dropout layers can be viewed as performing approximate variational inference. In practice, we train a neural network with dropout layers until convergence. At test time, activating the dropout layers is equivalent to sampling from the posterior distribution $\{\hat{\theta}_1, \ldots, \hat{\theta}_T\}$ (assuming we sample T times). Passing a new input $\mathbf{x}^*$ through the neural networks parameterised by $\{\hat{\theta}_1, \ldots, \hat{\theta}_T\}$, we obtain output samples from stochastic forward passes $\{f_{\hat{\theta}_1}(\mathbf{x}^*), \ldots, f_{\hat{\theta}_T}(\mathbf{x}^*)\}$. From these stochastic samples from an approximate predictive distribution, we can get an empirical estimator for the predictive mean and the predictive variance (uncertainty) of the predictive distribution (cf. sections 1.5 and 3.3, and eq. (3.16) of [23]):

$$\begin{aligned} \mathbb{E}\left[\mathbf{y}^*\right] &\approx \frac{1}{T}\sum_{i=1}^{T} f_{\hat{\theta}_i}(\mathbf{x}^*) \\ \operatorname{Var}\left[\mathbf{y}^*\right] &\approx \tau^{-1}\mathbf{I}_D + \frac{1}{T}\sum_{i=1}^{T} f_{\hat{\theta}_i}(\mathbf{x}^*)^T f_{\hat{\theta}_i}(\mathbf{x}^*) - \mathbb{E}\left[\mathbf{y}^*\right]^T \mathbb{E}\left[\mathbf{y}^*\right], \end{aligned} \tag{3.9}$$

where τ is the model precision of the Gaussian likelihood (for regression) and D is the output dimensionality. In the expression of the predictive variance, the first term (inverse model precision) captures the aleatoric uncertainty and the remaining terms (obtained from stochastic forward passes) capture the epistemic uncertainty (cf. section 6.7 of [23]). Besides BNNs, ensemble of neural networks [56, 64] is another widely used uncertainty estimation approach. We refer the reader to [27] for a detailed survey of uncertainty in deep neural networks.

3.3 Deep Generative Modelling

In discriminative modelling, one aims to learn a predictor given the observations, while in generative modelling, one aims to learn the joint distribution over all the variables. Conceptually, generative modelling is a way to simulate how the data is generated in the real world. Deep generative modelling attempts to do so by approximating the underlying generative process as deep neural networks i.e., highly parameterised neural networks. Generative modelling can be helpful in learning abstractions that can be used for downstream tasks. In particular,

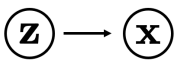


Figure 3.2: VAE assumption: A hidden variable $\mathbf{z}$ generates an observation $\mathbf{x}$.

variational autoencoders (VAEs) [48, 77] have considerably served in the quest of finding compact disentangled variations in data for unsupervised representation.

Suppose there exists a hidden variable $\mathbf{z}$ which generates an observation $\mathbf{x}$ (cf. fig. 3.2). We can only observe $\mathbf{x}$, but we would like to infer $\mathbf{z}$. In other words, we would like to compute $p(\mathbf{z}|\mathbf{x})$. By applying Bayes' theorem, we obtain:

$$p(\mathbf{z}|\mathbf{x}) = \frac{p(\mathbf{x}|\mathbf{z})p(\mathbf{z})}{p(\mathbf{x})}. \tag{3.10}$$

Same as previously, the integral to compute $p(\mathbf{x})$ is often intractable.

$$p(\mathbf{x}) = \int p(\mathbf{x}|\mathbf{z})p(\mathbf{z})d\mathbf{z}. \tag{3.11}$$

We use a variational distribution $q(\mathbf{z}|\mathbf{x})$ to approximate $p(\mathbf{z}|\mathbf{x})$. In variational inference, optimising the KL divergence $\mathrm{KL}(q(\mathbf{z}|\mathbf{x}) \;||\; p(\mathbf{z}|\mathbf{x}))$ can be cast into optimising the ELBO loss:

$$\mathcal{L}_{\mathrm{ELBO}} = \mathbb{E}_{\mathbf{z}\sim q(\mathbf{z}|\mathbf{x})} \log p(\mathbf{x}|\mathbf{z}) - \mathrm{KL}(q(\mathbf{z}|\mathbf{x}) \;||\; p(\mathbf{z})). \tag{3.12}$$

From a different point of view, a VAE defines an encoder $q_\phi(\mathbf{z}|\mathbf{x})$ and a decoder $p_\theta(\mathbf{x}|\mathbf{z})$, where $\mathbf{z}$ is a learned latent representation (cf. fig. 3.3). In the variational inference framework, a VAE is trained by minimising the ELBO loss (with parameterisation θ and ϕ):

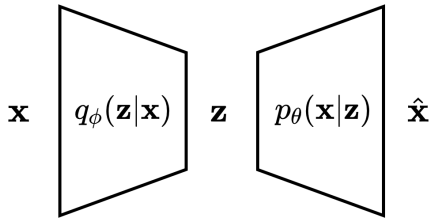


Figure 3.3: VAE model architecture: an input $\mathbf{x}$ is passed through the encoder $q_\phi(\mathbf{z}|\mathbf{x})$ and embedded to a latent representation $\mathbf{z}$. This representation is then passed through the decoder $p_\theta(\mathbf{x}|\mathbf{z})$ and reconstructed as $\hat{\mathbf{x}}$.

$$\mathcal{L}_{\text{ELBO}} = \mathbb{E}_{\mathbf{z}\sim q_\phi(\mathbf{z}|\mathbf{x})} \log p_\theta(\mathbf{x}|\mathbf{z}) - \text{KL}(q_\phi(\mathbf{z}|\mathbf{x}) \,||\, p(\mathbf{z})). \tag{3.13}$$

The first term represents the reconstruction loss. The second term, the KL divergence, ensures that the latent distribution $q_\phi(\mathbf{z}|\mathbf{x})$ is close to the prior distribution $p(\mathbf{z})$, which regularises the latent space. The reader is referred to [81] for an introduction to deep generative modelling and [49] for an introduction to VAEs.

4

Introspective Visuomotor Control: Exploiting Uncertainty in Deep Visuomotor Control for Failure Recovery

In this chapter, to address the problem of distribution shift in imitation learning, we investigate the use of uncertainty as a means to guide the visuomotor control model to recovery from failures before the end of a policy rollout. We adopt a probabilistic approach and propose a simple model to predict policy uncertainty. We demonstrate the effectiveness of this policy uncertainty as an indicator of task failure by analysing their correlation. Together with our proposed failure recovery strategy, we observe performance gains over the original visuomotor control model and several basic recovery strategy baselines on different manipulation tasks. This work is published as:

Chia-Man Hung, Li Sun, Yizhe Wu, Ioannis Havoutis, Ingmar Posner. "Introspective Visuomotor Control: Exploiting Uncertainty in Deep Visuomotor Control for Failure Recovery". In: *IEEE International Conference on Robotics and Automation (ICRA)*. June 2021.

Introspective Visuomotor Control: Exploiting Uncertainty in Deep Visuomotor Control for Failure Recovery

Chia-Man Hung[1,2], Li Sun[1], Yizhe Wu[1], Ioannis Havoutis[2], Ingmar Posner[1]

***Abstract*— End-to-end visuomotor control is emerging as a compelling solution for robot manipulation tasks. However, imitation learning-based visuomotor control approaches tend to suffer from a common limitation, lacking the ability to recover from an out-of-distribution state caused by compounding errors. In this paper, instead of using tactile feedback or explicitly detecting the failure through vision, we investigate using the uncertainty of a policy neural network. We propose a novel uncertainty-based approach to detect and recover from failure cases. Our hypothesis is that policy uncertainties can implicitly indicate the potential failures in the visuomotor control task and that robot states with minimum uncertainty are more likely to lead to task success. To recover from high uncertainty cases, the robot monitors its uncertainty along a trajectory and explores possible actions in the state-action space to bring itself to a more certain state. Our experiments verify this hypothesis and show a significant improvement on task success rate: 12% in pushing, 15% in pick-and-reach and 22% in pick-and-place.**

I. INTRODUCTION

Deep visuomotor control (VMC) is an emerging research area for closed-loop robot manipulation, with applications in dexterous manipulation, such as manufacturing and packing. Compared to conventional vision-based manipulation approaches, deep VMC aims to learn an end-to-end policy to bridge the gap between robot perception and control, as an alternative to explicitly modelling the object position/pose and planning the trajectories in Cartesian space.

The existing works on deep VMC mainly focus on domain randomisation [1], to transfer visuomotor skills from simulation to the real world [2], [3]; or one-shot learning [4], [5], to generalise visuomotor skills to novel tasks when large-scale demonstration is not available. In these works, imitation learning is used to train a policy network to predict motor commands or end-effector actions from raw image observations. Consequently, continuous motor commands can be generated, closing the loop of perception and manipulation. However, with imitation learning, the robot may fall into an unknown state-space to which the policy does not generalise, where it is likely to fail. Early diagnosis of failure cases is thus important for policy generalisation but an open question in deep VMC research.

Instead of using vision or tactile feedback to detect failure cases [6], [7], we extend the widely-used deterministic policy network to an introspective Bayesian network. The uncertainty obtained by this Bayesian network is then used to detect the failure status. More importantly, as a supplement to the existing deep VMC methods, we propose a recovery mechanism to rescue the manipulator when a potential failure is detected, where a predictive model can learn the intuitive uncertainty to indicate the status of manipulation without the need of simulating the manipulation using a physics engine.

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

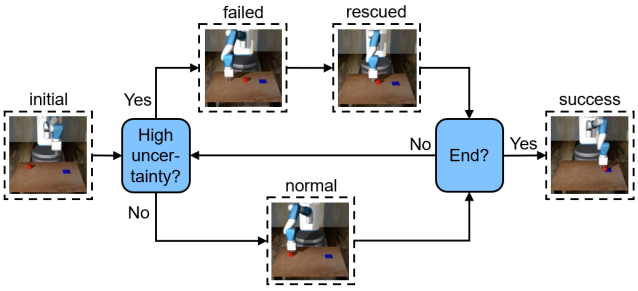


Fig. 1. An overview of the proposed VMC approach with failure case recovery. In this example, the task is to push the red cube onto the target.

In summary, our contributions are three-fold: First, we extend VMC to a probabilistic model which is able to estimate its epistemic uncertainty. Second, we propose a simple model to predict the VMC policy uncertainty conditioned on the action without simulating it. Finally, leveraging the estimated policy uncertainty, we propose a strategy to detect and recover from failures, thereby improving the success rate of a robot manipulation task.

II. RELATED WORK

The problem we are considering is based on learning robot control from visual feedback and monitoring policy uncertainty to optimise overall task success rate. Our solution builds upon visuomotor control, uncertainty estimation and failure case recovery.

Visuomotor Control. To plan robot motion from visual feedback, an established line of research is to use visual model-predictive control. The idea is to learn a forward model of the world, which forecasts the outcome of an action. In the case of robot control, a popular approach is to learn the state-action transition models in a latent feature embedding space, which are further used for motion planning [8], [9], [10]. Likewise, visual foresight [11] leverages a deep video prediction model to plan the end-effector motion by sampling actions leading to a state which approximates the goal image. However, visual model-predictive control relies on learning a good forward model, and sampling suitable actions is not only computationally expensive but also requires finding a good action distribution. End-to-end methods solve the issues mentioned above by directly predicting the next action. Guided policy search [12] was one of the first to employ

an end-to-end trained neural network to learn visuomotor skills, yet their approach requires months of training and multiple robots. Well-known imitation learning approaches such as GAIL [13] and SQIL [14] could also serve as backbones upon which we build our probabilistic approach. However, we chose end-to-end visuomotor control [1] as our backbone network architecture, for its simplicity and ability to achieve a zero-shot sim-to-real adaption through domain randomisation.

Uncertainty Estimation. Approaches that can capture predictive uncertainties such as Bayesian Neural Networks [15] and Gaussian Processes [16] usually lack scalability to big data due to the computational cost of inferring the exact posterior distribution. Deep neural networks with dropout [17] address this problem by leveraging variational inference [18] and imposing a Bernoulli distribution over the network parameters. The dropout training can be cast as approximate Bayesian inference over the network's weights [19]. Gal et al. [20] show that for the deep convolutional networks with dropout applied to the convolutional kernels, the uncertainty can also be computed by performing Monte Carlo sampling at the test phase. Rather than doing a grid search over the dropout rate which is computationally expensive, concrete dropout [21] relaxes the discrete Bernoulli distribution to the concrete distribution and thus allows the dropout rate to be trained jointly with other model parameters using the reparameterisation trick [22].

Failure Case Recovery. Most of the existing research utilise the fast inference of deep models to achieve closed-loop control [23], [24], [25]. However, failure case detection and recovery in continuous operation has not been considered in other works. Moreover, predicted actions are usually modelled as deterministic [26], [27], while the uncertainty of the policy networks has not been thoroughly investigated. Another line of research considering failure recovery is interactive imitation learning, which assumes access to an oracle policy. Similar to our work, HG-DAgger [28] estimates the epistemic uncertainty in an imitation learning setting, but by formulating their policy as an ensemble of neural networks, and they use the uncertainty to determine at which degree a human should intervene. In this paper, our intuition is to detect the failure cases by monitoring the uncertainty of the policy neural network and rescue the robot when it is likely to fail by exploring into the robot state-action space under high confidence (i.e. low uncertainties).

III. MODELLING UNCERTAINTY IN DEEP VISUOMOTOR CONTROL

To detect the potential failure cases in manipulation, we build a probabilistic policy network for VMC. Uncertainty is viewed as an indicator of the likelihood of task failure.

End-to-End Visuomotor Control. For clarity, we first briefly review the end-to-end visuomotor control model [1]. At timestep t, it takes K consecutive frames of raw RGB images $(I_{t-K+1},...,I_t)$ as input to a deep convolutional neural network and outputs the embedding $(\mathbf{e}_{t-K+1},...,\mathbf{e}_t)$. To incorporate the configuration space information, the embedding

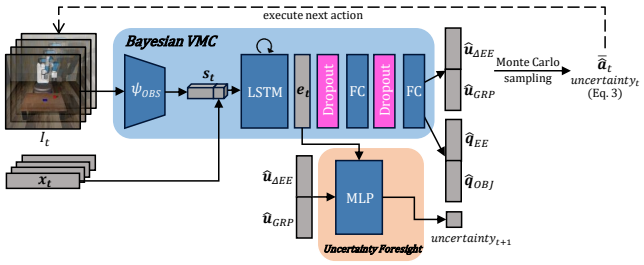


Fig. 2. Network architecture of Introspective Visuomotor Control model. Blue: the backbone Bayesian Visuomotor Control model. The current observation I_t is passed through a CNN ψ_{OBS}. This spatial feature map is concatenated to the tiled proprioceptive feature $\mathbf{x}_t$. The concatenated state representation $\mathbf{s}_t$ is fed into an LSTM. The LSTM embedding $\mathbf{e}_t$ is passed through a number of concrete dropout layers and fully connected layers interleavingly, whose output is then decoded into action commands $\hat{\mathbf{u}}_{\Delta EE}$ and $\hat{\mathbf{u}}_{GRP}$ as well as auxiliary position predictions $\hat{\mathbf{q}}_{EE}$ and $\hat{\mathbf{q}}_{OBJ}$. During test time, the mean action $\bar{\hat{\mathbf{a}}}_t$ is executed as the next action. The uncertainty estimate of the next timestep is used to supervise the prediction of the uncertainty foresight model. Orange: uncertainty foresight model. The LSTM embedding $\mathbf{e}_t$ is concatenated with the action commands $\hat{\mathbf{u}}_{\Delta EE}$ and $\hat{\mathbf{u}}_{GRP}$. It is passed through an MLP with 2 fully connected layers to predict the uncertainty associated with the next embedding $\mathbf{e}_{t+1}$.

is first concatenated with the corresponding robot joint angles $(\mathbf{x}_{t-K+1},...,\mathbf{x}_t)$ and then fed into a recurrent network followed by a fully connected layer. The buffered history information of length K is leveraged to capture the higher-order states, e.g. the velocity and acceleration. In an object manipulation task using a robot gripper, the model predicts the next joint velocity command $\hat{\mathbf{u}}_J$ and the next discrete gripper action $\hat{\mathbf{u}}_{GRP}$ (open, close or no-op) as well as the object position $\hat{\mathbf{q}}_{OBJ}$ and gripper position $\hat{\mathbf{q}}_{EE}$ as auxiliary targets with the following loss objective:

$$\begin{aligned}\mathscr{L}_{\text{total}} = &\text{MSE}(\hat{\mathbf{u}}_J,\mathbf{u}_J)+\text{CCE}(\hat{\mathbf{u}}_{GRP},\mathbf{u}_{GRP})\\ &+\text{MSE}(\hat{\mathbf{q}}_{OBJ},\mathbf{q}_{OBJ})+\text{MSE}(\hat{\mathbf{q}}_{EE},\mathbf{q}_{EE}),\end{aligned} \tag{1}$$

where MSE and CCE stand for *Mean-Squared Error* and *Categorical Cross-Entropy* respectively. The losses are equally weighted and the model is trained end-to-end with stochastic gradient descent.

In this work, we use delta end-effector position command $\hat{\mathbf{u}}_{\Delta EE}$ rather than joint velocity command $\hat{\mathbf{u}}_J$ as a model output. We have found this to be more stable and less prone to the accumulated error over a long time horizon. We feed a buffer of $K=4$ input frames at every timestep, and as we rollout the model, we keep the LSTM memory updated along the whole trajectory, as opposed to just K buffered frames.

Uncertainty Estimation. In the Bayesian setting, the exact posterior distribution of the network weights is intractable in general, due to the marginal likelihood. In the variational inference case, we consider an approximating variational distribution, which is easy to evaluate. To approximate the posterior distribution, we minimise the Kullback-Leibler divergence between the variational distribution and the posterior distribution. Gal et al. [19] propose using dropout as a simple stochastic regularisation technique to approximate the variational distribution. Training a deep visuomotor control policy with dropout not only reduces overfitting, but also

enforces the weights to be learned as a distribution and thus can be exploited to model the epistemic uncertainty.

In practice, we train a Bayesian dropout visuomotor control policy and evaluate the posterior action command distribution by integrating Monte Carlo samples. At test time, we rollout the policy by performing stochastic forward passes at each timestep. Figure 2 depicts the network architecture of our model. To learn the dropout rate adaptively, we add concrete dropout layers. Concrete dropout [21] uses a continuous relaxation of dropout's discrete masks and enables us to train the dropout rate as part of the optimisation objective, for the benefit of providing a well-calibrated uncertainty estimate. We also experiment with the number of dropout layers. We choose one and two layers since we do not want to add unnecessary trainable parameters and increase the computation cost. The number of fully connected layers is adjusted according to that of dropout layers.

At timestep t, we draw action samples $A_t = \{\hat{\mathbf{a}}_t^1, \hat{\mathbf{a}}_t^2, ...\}$, where $\hat{\mathbf{a}}_t^i = [\hat{\mathbf{u}}_{\Delta EE,t}^i, \hat{\mathbf{u}}_{GRP,t}^i]^T$ is a model output, and use their mean $\bar{\hat{\mathbf{a}}}_t = \text{mean}(A_t)$ as the action command to execute in the next iteration. For an uncertainty estimate, following probabilistic PoseNet [29], we have experimented with the trace of covariance matrix of the samples and the maximum of the variance along each axis. Similarly, we have found the trace to be a representative scalar measure of uncertainty.

Simply computing the trace from a batch of sampled action commands does not capture the uncertainty accurately in cases where the predicted values vary significantly in norm in an episode. For instance, when the end-effector approaches an object to interact with, it needs to slow down. At such a timestep, since the predicted end-effector commands are small, the trace of the covariance matrix is also small. To calibrate the uncertainty measure, we transform every predicted delta end-effector position command $\hat{\mathbf{u}}_{\Delta EE}$ into norm and unit vector, weight them with λ and $1-\lambda$ respectively, and concatenate them as a 4-dimensional vector $\hat{\mathbf{X}}$, before computing the trace:

$$\hat{\mathbf{u}}_{\Delta EE} = [\hat{u}_x, \hat{u}_y, \hat{u}_z]^T \mapsto \hat{\mathbf{X}} = \\ [\lambda \|\hat{\mathbf{u}}_{\Delta EE}\|, (1-\lambda)\tfrac{\hat{u}_x}{\|\hat{\mathbf{u}}_{\Delta EE}\|}, (1-\lambda)\tfrac{\hat{u}_y}{\|\hat{\mathbf{u}}_{\Delta EE}\|}, (1-\lambda)\tfrac{\hat{u}_z}{\|\hat{\mathbf{u}}_{\Delta EE}\|}]^T. \quad (2)$$

Here λ is treated as a hyper-parameter. The superscripts i denoting sample id and the subscripts t denoting timestep are omitted for readability.

To determine how many Monte Carlo samples are required to achieve convergence, we compare the predicted action commands with the ground truth in validation episodes. We compute the median error in each episode and average over validation episodes. Monte Carlo sampling converges after around 50 samples and no more improvement is observed with more samples. We thus define:

$$\text{uncertainty}_t = \text{Tr}\left(\text{cov}\left([\hat{\mathbf{X}}_t^1, \hat{\mathbf{X}}_t^2, ..., \hat{\mathbf{X}}_t^{50}]^T\right)\right), \quad (3)$$

where $\hat{\mathbf{X}}_t^i \in \mathbb{R}^{4\times 1}$ is a sampled prediction transformed into weighted norm and unit vector in Eq. 2.

IV. RECOVERY FROM FAILURES

Our Bayesian visuomotor control model provides us with an uncertainty estimate of the current state at each timestep. In this section, we describe how we make use of it to recover from failures.

Knowing When to Recover. Continuously executing an uncertain trajectory is likely to lead to failure; diagnosis in an early stage and recovery can bring execution back on track. The question is, at which point shall we switch to a recovery mode to optimise overall success? Having a Bayesian VMC model trained, we deploy it on validation episodes to pick an optimal threshold of uncertainty for recovery. Section V details how to pick this threshold. During test time, as we rollout the model, when the uncertainty estimate is over the threshold, we switch to a recovery mode.

Following Minimum Uncertainty. Once the robot is switched to a recovery mode, our intuition is to explore in the state-action space and modify the robot configuration to an area trained with sufficient training examples. Hence, we propose moving along the trajectory with minimisation of uncertainty. However, the uncertainty estimate from the Bayesian VMC model in Figure 2 is associated with the current state. The Bayesian VMC model cannot provide the uncertainty of future frames without physically trying it. To address this issue, drawing inspiration from Embed to Control [8] which extracts a latent dynamics model for control from raw images, we came up with the idea of learning a transition model mapping from the current latent feature embedding $\mathbf{e}_t$ given by our Bayesian VMC model to future $\mathbf{e}_{t+1}$ conditioned on an action $\mathbf{a}_t$. Then the predicted feature embedding $\mathbf{e}_{t+1}$ could be fed as input to the first dropout layer through the last fully connected layer to sample actions and estimate the uncertainty. However, this approach of predicting next embedding $\mathbf{e}_{t+1}$ conditioned on action $\mathbf{a}_t$ would require further Monte Carlo sampling to estimate the uncertainty, making it computationally costly during test time.

Instead of predicting in the latent space, inspired by Visual Foresight [11], we predict the uncertainty of the next embedding $\mathbf{e}_{t+1}$ after executing $\mathbf{a}_t$ directly. This can be achieved by Knowledge Distillation [30]. Specifically, we use the model uncertainty of time t+1 as the learning target to train the uncertainty foresight model. We refer the reader to Figure 2.

During test time, when the minimum uncertainty recovery mode is activated, we first backtrack the position of the end-effector to a point of minimum uncertainty within 20 steps. This is implemented by storing action, LSTM memory, uncertainty estimate and timestep in a FIFO queue of a maximum size of 20. Although the original state cannot always be recovered exactly in the case when the object is moved or when considering sensing and motor noise on a real system, backtracking guides the robot back into the vicinity of states where previous policy execution was confident. Then, at each timestep, we sample actions from the Bayesian VMC model and choose the action leading to

Algorithm 1 Failure recovery for Bayesian VMC (test time)

Require: f: trained Bayesian VMC model, g: trained Bayesian VMC model and uncertainty foresight module, outputting the action with the minimum epistemic uncertainty among samples from f, $T_{recovery}$: minimum recovery interval, S: number of samples used to compute uncertainty, C: recovery threshold.

 # Rollout a trained model.
while true do
  Sample S actions from f and compute their mean and uncertainty estimate.
  Update the sum of a sliding window of uncertainties.
  # Check if failure recovery is needed.
  if time since last recovery attempt $> T_{recovery}$ **and** uncertainty sum $> C$ **then**
    # Uncertainty is high: start recovery.
    Double $T_{recovery}$.
    Update last recovery attempt timestep.
    Backtrack to a position with min uncertainty within the last few steps; restore memory.
    Rollout g for a number of steps.
  else
    # Uncertainty is low: perform a normal action.
    Execute the mean action command of Monte Carlo sampling from f.
  end if
  if maximum episode steps reached **or** task success **then**
    break
  end if
end while

return binary task success

the next state with minimum uncertainty according to our uncertainty foresight model. Algorithm 1 explains how this works within the Bayesian VMC prediction loop. With the same minimum recovery interval, we have observed that it is common to get stuck in a recovery loop, where after recovery the robot becomes too uncertain at the same place and goes into recovery mode again. Inspired by the binary exponential backoff algorithm – an algorithm used to space out repeated retransmissions of the same block of data to avoid network congestion – we double the minimum recovery interval every time that the recovery mode is activated. This simple intuitive trick solves the problem mentioned above well empirically.

V. EXPERIMENTS

Our experiments are designed to answer the following questions: **(1)** Is uncertainty computed from stochastic sampling from our Bayesian VMC models a good indication of how well the model performs in an episode? **(2)** How well can our model recover from failures? **(3)** How well does our proposed minimum uncertainty recovery strategy perform compared to other recovery modes?

Experimental Setup and Data Collection. We follow Gorth et al. [31] and use the MuJoCo physics engine [32] along with an adapted Gym environment [33] provided by [4] featuring the *Fetch Mobile Manipulator* [34] with a 7-DoF arm and a 2-finger gripper. Three tasks (Figure 3) are designed as they are fundamental in manipulation and commonly used as building blocks for more complex tasks. In the pushing and pick-and-place tasks, the cube and the target are randomly spawned in a 6x8 grid, as opposed to only 16 initial cube positions and 2 initial target positions in the VMC [1] pick-and-place task. In the pick-and-reach task, the stick and the target are spawned in 2 non-overlapping 6x8 grids. Similarly, we generate expert trajectories by placing pre-defined waypoints and solving the inverse kinematics. For each task, 4,000 expert demonstrations in simulation are collected, each lasting 4 seconds long. These are recorded as a list of observation-action tuples at 25 Hz, resulting in an episode length of $H = 100$. For the uncertainty foresight model, we collect 2,000 trajectories from deploying a trained Bayesian VMC. At every timestep, we execute an action sampled from the Bayesian VMC. We record the current embedding, the action executed and the uncertainty of the next state after the action is executed, as described in Section III. An episode terminates after the task is completed or after the maximum episode limit of 200 is reached.

Fig. 3. Top: Example of a pushing expert demonstration. The robot first pushes the red cube forward to align it with the blue target, and then moves to the side to push it sideways onto the target. Middle: Example of pick-and-place expert demonstration. The robot first moves toward the red cube to pick it up, and then moves to the blue target to drop the cube. Bottom: Example of a pick-and-reach expert demonstration. The robot first moves towards the red stick to pick it up at one end, and then reaches the blue target with the other end.

Picking Uncertainty Threshold. Uncertainty estimates can sometimes be noisy, so we smooth them out using a sliding window, given the assumption that uncertainties contiguously change throughout the course of a trajectory. We have found a sliding window of 20 frames best avoids noisy peaks. It is worth mentioning that the simulator runs at 25 Hz and 20 frames correspond to only 0.8 seconds. For each evaluation episode, we record a binary label (i.e. task fail/success) and the maximum sum of a sliding window of uncertainties along the episode. In the following, we denote

the maximum sum of a sliding window of uncertainties as u or maximum uncertainty. We sort the episodes by their maximum uncertainty in increasing order. Under the assumption that the probability of success after recovery is the overall average task success rate which is already known, we pick a threshold to maximise the overall task success rate after recovery, which is equivalent to maximising the increase of successes. We find the sorted episode index as follows.

$$i^* = \underset{i}{\text{argmax}}(\,|\{x \mid u(x) > u_i\}| \cdot \bar{r} - |\{x \mid u(x) > u_i, \text{result}(x) = \text{success})\}|), \tag{4}$$

where x is an episode, $u(x)$ is the maximum uncertainty of episode x, u_i is the maximum uncertainty of episode indexed i, and $\bar{r}$ is the overall average success rate.

During test time, as we rollout the model, when the sum of a sliding window of 20 previous uncertainties is greater than the threshold of maximum uncertainty u_{i^*}, we switch to the recovery mode.

Baselines for Visuomotor Control Recovery. Our aim is to show our proposed failure recovery mode outperforms other failure recovery modes, as well the backbone VMC [1]. Thus, we do not directly compare it against other visuomotor control approaches. We compare our failure recovery mode MIN UNC in Section IV against two baselines: RAND and INIT. The recoveries all happen when the uncertainty is high while deploying a Bayesian VMC (line 7 of Algorithm 1). We use a maximum of 25 recovery steps in all cases. **(1)** RAND: The end-effector randomly moves 25 steps and we keep the gripper open amount as it is (no-op). Then, we reset the LSTM memory. **(2)** INIT: We open the gripper, sample a point in a sphere above the table and move the end-effector to that point. Then, we reset the LSTM memory. This recovery mode is designed to reset to a random initial position. All the recovery modes attempt to move the robot from an uncertain state to a different one, with the hope of it being able to interpolate from the training dataset starting from a new state.

VI. RESULTS

Task Success vs Uncertainty Estimate. Is uncertainty estimate a good indication of how well the model performs in an episode? To address this first guiding question in Section V, we analyse how the task success rate varies with respect to the uncertainty estimate from our Bayesian VMC models. We evaluate on 800 test scene setups and regroup them by maximum uncertainty into 10 bins. Figure 4 shows the task success rate versus maximum uncertainty in each bin. We observe that task success rate is inversely correlated with maximum uncertainty, which corroborates our hypothesis of high uncertainty being more likely to lead to failure.

Manipulation with Failure Recovery Results. Regarding the last two guiding questions in Section V, we evaluate the performance of the controllers on 100 held-out test scene setups for all three tasks. We report all model performances in Table I.

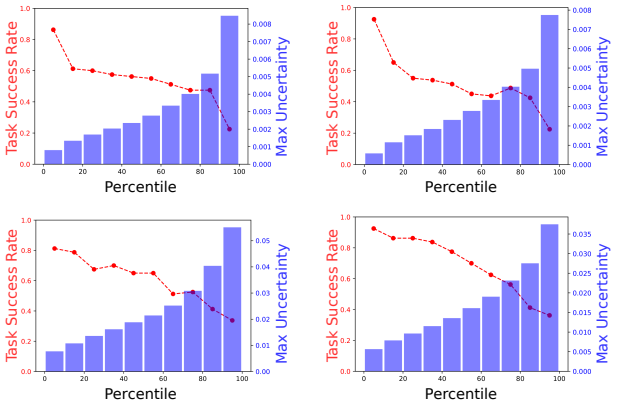


Fig. 4. Evaluation of task success rate vs maximum uncertainty of different models evaluated over 800 test episodes. Left: one dropout layer. Right: two dropout layers. Top: pushing. Bottom: pick-and-place. These plots are drawn by sorting episodes by their maximum uncertainty and regrouping them into 10 bins. Subsequently, the average task success rate and the average maximum uncertainty are computed for each bin.

In the first row, we compare against VMC, the original deterministic VMC model [1], but with one or two fully connected layers after the LSTM. Next, BVMC, the Bayesian VMC model executing the mean of the sampled predictions at each timestep, but not using the uncertainty estimate information for recovery. Although this does not perform any recovery, the network architecture is slightly different than VMC due to the added concrete dropout layer(s). BVMC + RAND and BVMC + INIT are the baseline recovery modes (Section V). Last, we present BVMC + MIN UNC, our proposed recovery mode following minimum uncertainty (Section IV).

In the pushing task, although the reaching performance of BVMC drops compared to VMC, the pushing performance is slightly better. In general, adding stochasticity and weight regularisation prevents overfitting, but it does not always boost performance. BVMC + RAND and BVMC + INIT outperform BVMC by approximately 5% in both cases of one and two fully connected layers. The performance increase is moderate because a large proportion of bins of episodes in the mid maximum uncertain range has a task success rate close to the average overall task success rate (Figure 4) and the threshold of maximum uncertainty picked is relatively high, thus not allowing many episodes to switch to a recovery mode. In general, the models with two fully connected layers have higher performance than their counterparts with one fully connected layer. This can be understood as having more trainable parameters helps learn a better function approximation. Our proposed BVMC + MIN UNC surpasses other two baseline recovery modes, indicating that following actions with minimum uncertainty contributes further to the task success.

In pick-and-place and pick-and-reach, all VMC and Bayesian VMC models exhibit near perfect reaching performance. Also, surprisingly, all models do better than their counterparts in the pushing task. At first glance, both tasks

MODEL #FC=1	PUSHING		PICK-AND-PLACE			PICK-AND-REACH		
	REACH [%]	PUSH [%]	REACH [%]	PICK [%]	PLACE [%]	REACH [%]	PICK [%]	TASK [%]
VMC [1]	**97.00 ± 1.62**	49.00 ± 4.74	99.00 ± 0.94	77.00 ± 3.99	52.00 ± 4.74	**99.00 ± 0.94**	77.00 ± 3.99	69.00 ± 4.39
BVMC	91.00 ± 2.71	50.00 ± 4.75	99.00 ± 0.94	84.00 ± 3.48	60.00 ± 4.65	**99.00 ± 0.94**	88.00 ± 3.08	78.00 ± 3.93
+ RAND	93.00 ± 2.42	**56.00 ± 4.71**	99.00 ± 0.94	**85.00 ± 3.39**	68.00 ± 4.43	**99.00 ± 0.94**	**89.00 ± 2.97**	**81.00 ± 3.72**
+ INIT	93.00 ± 2.42	**55.00 ± 4.72**	99.00 ± 0.94	**88.00 ± 3.08**	67.00 ± 4.46	**99.00 ± 0.94**	**93.00 ± 2.42**	**79.00 ± 3.86**
+ MIN UNC	**94.00 ± 2.25**	**58.00 ± 4.68**	99.00 ± 0.94	**90.00 ± 2.85**	70.00 ± 4.35	**99.00 ± 0.94**	**93.00 ± 2.42**	**82.00 ± 3.64**
MODEL #FC=2	PUSHING		PICK-AND-PLACE			PICK-AND-REACH		
	REACH [%]	PUSH [%]	REACH [%]	PICK [%]	PLACE [%]	REACH [%]	PICK [%]	TASK [%]
VMC [1]	**96.00 ± 1.86**	50.00 ± 4.75	97.00 ± 1.62	79.00 ± 3.86	60.00 ± 4.65	**99.00 ± 0.94**	79.00 ± 3.86	70.00 ± 3.64
BVMC	88.00 ± 3.08	**53.00 ± 4.74**	**100.00 ± 0.00**	**87.00 ± 3.19**	69.00 ± 4.39	**99.00 ± 0.94**	**89.00 ± 2.97**	**79.00 ± 3.86**
+ RAND	88.00 ± 3.08	**60.00 ± 4.65**	**100.00 ± 0.00**	**91.00 ± 2.71**	74.00 ± 4.16	**99.00 ± 0.94**	**91.00 ± 2.71**	**82.00 ± 3.64**
+ INIT	93.00 ± 2.42	**58.00 ± 4.68**	**100.00 ± 0.00**	**89.00 ± 2.97**	**76.00 ± 4.05**	**99.00 ± 0.94**	**93.00 ± 2.42**	**83.00 ± 3.56**
+ MIN UNC	91.00 ± 2.71	**62.00 ± 4.61**	**100.00 ± 0.00**	**89.00 ± 2.97**	**82.00 ± 3.64**	**99.00 ± 0.94**	**94.00 ± 2.25**	**85.00 ± 3.39**

TABLE I

COMPARISON OF MODEL PERFORMANCES WITH AND WITHOUT FAILURE RECOVERY IN THE PUSHING, PICK-AND-PLACE AND PICK-AND-REACH TASKS. TOP: ONE FULLY CONNECTED LAYER. BOTTOM: TWO FULLY CONNECTED LAYERS. BEST TASK PERFORMANCES ARE BOLD-FACED.

seem to be more difficult than pushing. In fact, the design of our pushing task requires a two-stage rectangular push. We observe most failure cases in pushing happen when the end-effector does not push at the centre of the cube, so that the cube is pushed to an orientation never seen in the training dataset. This rarely happens in the pick-and-place and pick-and-reach tasks. Similarly, BVMC + RAND and BVMC + INIT show a performance increase compared to BVMC + NO. Last but not least, BVMC + MIN UNC almost surpasses all other models in reaching, picking and placing/task, with a task success rate increase of 22% compared to VMC for pick-and-place and 15% for pick-and-reach.

Qualitatively, we observe interesting behaviours from our uncertainty estimates and recovery modes. In all three tasks, when a Bayesian VMC controller approaches the cube with a deviation to the side, we often see the controller fall into the recovery mode, while a VMC controller with the same scene setup continues the task and eventually get stuck in a position without further movements. Occasionally, in the pick-and-place and pick-and-reach tasks when the end-effector moves up without grasping the cube successfully, the Bayesian VMC controller monitors high uncertainty and starts recovery.

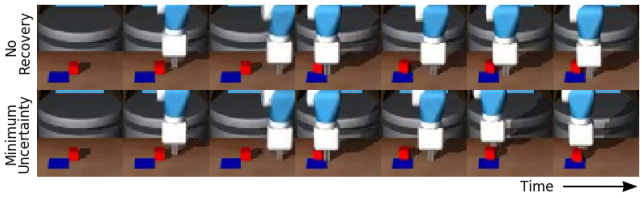


Fig. 5. Recovery comparison. The top row depicts operation without recovery, while the bottom row shows the results with recovery based on the minimum uncertainty. The robot fails to accomplish the pushing task without the recovery. The images are cropped to emphasise the difference.

System Efficiency. Recovery from uncertain states improves task performance. However, drawing stochastic samples also comes at an additional time cost. By design of our network architecture, only the last dropout layers and fully connected layers need to be sampled, since the first 8 layers of convolutional neural network and LSTM are deterministic. For reference, on an NVIDIA GeForce GTX 1080, averaging 50 Monte Carlo samples and computing the uncertainty take around 0.1 seconds, while the original VMC takes around 0.03 seconds per timestep. If treating the inference as a mini-batch of operations, this extra computation can be further reduced [35].

VII. CONCLUSIONS

This paper investigates the usage of policy uncertainty for failure case detection and recovery. In our method, a Bayesian neural network with concrete dropout is employed to obtain the model epistemic uncertainty by Monte Carlo sampling. We further make use of a deterministic model and knowledge distillation to learn the policy uncertainty of a future state conditioned on an end-effector action. Consequently, we are able to predict the uncertainty of a future timestep without physically simulating the actions. The experimental results verified our hypothesis – the uncertainties of the VMC policy network can be used to provide intuitive feedback to assess the failure/success in manipulation tasks, and, reverting and driving the robot to a configuration with minimum policy uncertainty can recover the robot from potential failure cases.

ACKNOWLEDGMENT

Chia-Man Hung is funded by the Clarendon Fund and receives a Keble College Sloane Robinson Scholarship at the University of Oxford. Yizhe Wu is funded by the China Scholarship Council. This research is also supported by an EPSRC Programme Grant [EP/M019918/1] and a gift from Amazon Web Services (AWS). The authors acknowledge the use of the University of Oxford Advanced Research Computing (ARC) facility in carrying out this work (`http://dx.doi.org/10.5281/zenodo.22558`). We also thank Ruoqi He, Hala Lamdouar, Walter Goodwin and Oliver Groth for proofreading and useful discussions, and the reviewers for valuable feedback.

REFERENCES

[1] S. James, A. J. Davison, and E. Johns, "Transferring end-to-end visuomotor control from simulation to real world for a multi-stage task," in *Conference on Robot Learning*, 2017, pp. 334–343.

[2] K. Bousmalis, A. Irpan, P. Wohlhart, Y. Bai, M. Kelcey, M. Kalakrishnan, L. Downs, J. Ibarz, P. Pastor, K. Konolige, S. Levine, and V. Vanhoucke, "Using simulation and domain adaptation to improve efficiency of deep robotic grasping," in *2018 IEEE International Conference on Robotics and Automation (ICRA)*, May 2018, pp. 4243–4250.

[3] S. James, P. Wohlhart, M. Kalakrishnan, D. Kalashnikov, A. Irpan, J. Ibarz, S. Levine, R. Hadsell, and K. Bousmalis, "Sim-to-real via sim-to-sim: Data-efficient robotic grasping via randomized-to-canonical adaptation networks," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2019, pp. 12 627–12 637.

[4] Y. Duan, M. Andrychowicz, B. Stadie, O. J. Ho, J. Schneider, I. Sutskever, P. Abbeel, and W. Zaremba, "One-shot imitation learning," in *Advances in neural information processing systems*, 2017, pp. 1087–1098.

[5] C. Finn, T. Yu, T. Zhang, P. Abbeel, and S. Levine, "One-shot visual imitation learning via meta-learning," in *Conference on Robot Learning*, 2017, pp. 357–368.

[6] D. Kragić, L. Petersson, and H. I. Christensen, "Visually guided manipulation tasks," *Robotics and Autonomous Systems*, vol. 40, no. 2-3, pp. 193–203, 2002.

[7] L. Sun, G. Aragon-Camarasa, S. Rogers, R. Stolkin, and J. P. Siebert, "Single-shot clothing category recognition in free-configurations with application to autonomous clothes sorting," in *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2017, pp. 6699–6706.

[8] M. Watter, J. Springenberg, J. Boedecker, and M. Riedmiller, "Embed to control: A locally linear latent dynamics model for control from raw images," in *Advances in neural information processing systems*, 2015, pp. 2746–2754.

[9] P. Agrawal, A. V. Nair, P. Abbeel, J. Malik, and S. Levine, "Learning to poke by poking: Experiential learning of intuitive physics," in *Advances in neural information processing systems*, 2016, pp. 5074–5082.

[10] T. Yu, G. Shevchuk, D. Sadigh, and C. Finn, "Unsupervised visuomotor control through distributional planning networks," *arXiv preprint arXiv:1902.05542*, 2019.

[11] C. Finn and S. Levine, "Deep visual foresight for planning robot motion," in *2017 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2017, pp. 2786–2793.

[12] S. Levine, C. Finn, T. Darrell, and P. Abbeel, "End-to-end training of deep visuomotor policies," *The Journal of Machine Learning Research*, vol. 17, no. 1, pp. 1334–1373, 2016.

[13] J. Ho and S. Ermon, "Generative adversarial imitation learning," in *Advances in neural information processing systems*, 2016, pp. 4565–4573.

[14] S. Reddy, A. D. Dragan, and S. Levine, "Sqil: Imitation learning via reinforcement learning with sparse rewards," *arXiv preprint arXiv:1905.11108*, 2019.

[15] D. J. MacKay, "A practical bayesian framework for backpropagation networks," *Neural computation*, vol. 4, no. 3, pp. 448–472, 1992.

[16] C. E. Rasmussen, "Gaussian processes in machine learning," in *Summer School on Machine Learning*. Springer, 2003, pp. 63–71.

[17] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: a simple way to prevent neural networks from overfitting," *The journal of machine learning research*, vol. 15, no. 1, pp. 1929–1958, 2014.

[18] M. I. Jordan, Z. Ghahramani, T. S. Jaakkola, and L. K. Saul, "An introduction to variational methods for graphical models," *Machine learning*, vol. 37, no. 2, pp. 183–233, 1999.

[19] Y. Gal and Z. Ghahramani, "Dropout as a bayesian approximation: Representing model uncertainty in deep learning," in *international conference on machine learning*, 2016, pp. 1050–1059.

[20] ——, "Bayesian convolutional neural networks with bernoulli approximate variational inference," *arXiv preprint arXiv:1506.02158*, 2015.

[21] Y. Gal, J. Hron, and A. Kendall, "Concrete dropout," in *Advances in neural information processing systems*, 2017, pp. 3581–3590.

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

[23] Y. Zhu, Z. Wang, J. Merel, A. Rusu, T. Erez, S. Cabi, S. Tunyasuvunakool, J. Kramár, R. Hadsell, N. de Freitas, *et al.*, "Reinforcement and imitation learning for diverse visuomotor skills," *arXiv preprint arXiv:1802.09564*, 2018.

[24] L. Pinto, M. Andrychowicz, P. Welinder, W. Zaremba, and P. Abbeel, "Asymmetric actor critic for image-based robot learning," *arXiv preprint arXiv:1710.06542*, 2017.

[25] J. Matas, S. James, and A. J. Davison, "Sim-to-real reinforcement learning for deformable object manipulation," *arXiv preprint arXiv:1806.07851*, 2018.

[26] A. Srinivas, A. Jabri, P. Abbeel, S. Levine, and C. Finn, "Universal planning networks," *arXiv preprint arXiv:1804.00645*, 2018.

[27] Y. Lee, E. S. Hu, Z. Yang, and J. J. Lim, "To follow or not to follow: Selective imitation learning from observations," *arXiv preprint arXiv:1912.07670*, 2019.

[28] M. Kelly, C. Sidrane, K. Driggs-Campbell, and M. J. Kochenderfer, "Hg-dagger: Interactive imitation learning with human experts," in *2019 International Conference on Robotics and Automation (ICRA)*. IEEE, 2019, pp. 8077–8083.

[29] A. Kendall and R. Cipolla, "Modelling uncertainty in deep learning for camera relocalization," in *2016 IEEE international conference on Robotics and Automation (ICRA)*. IEEE, 2016, pp. 4762–4769.

[30] S. R. Bulò, L. Porzi, and P. Kontschieder, "Dropout distillation," in *International Conference on Machine Learning*, 2016, pp. 99–107.

[31] O. Groth, C.-M. Hung, A. Vedaldi, and I. Posner, "Goal-conditioned end-to-end visuomotor control for versatile skill primitives," *arXiv preprint arXiv:2003.08854*, 2020.

[32] E. Todorov, T. Erez, and Y. Tassa, "Mujoco: A physics engine for model-based control," in *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, 2012, pp. 5026–5033.

[33] G. Brockman, V. Cheung, L. Pettersson, J. Schneider, J. Schulman, J. Tang, and W. Zaremba, "Openai gym," 2016.

[34] M. Wise, M. Ferguson, D. King, E. Diehr, and D. Dymesich, "Fetch & freight: Standard platforms for service robot applications," 2018. [Online]. Available: https://fetchrobotics.com/wp-content/uploads/2018/04/Fetch-and-Freight-Workshop-Paper.pdf

[35] Y. Gal, "Uncertainty in deep learning," *University of Cambridge*, vol. 1, p. 3, 2016.

5

Goal-Conditioned End-to-End Visuomotor Control for Versatile Skill Primitives

In this chapter, we propose a novel network architecture for goal-conditioned end-to-end visuomotor control. Unlike prior work, it can be efficiently conditioned on a new task with just a single target image without fine-tuning, as is common in imitation learning and meta learning. In other words, it learns general skill primitives to solve new tasks with unseen scene setups. We demonstrates the model's efficacy by its strong performance benchmarking on simulated pushing and pick-and-place tasks. Furthermore, we show its robustness and versatility in that it transfers directly to new scenarios without the need of domain randomisation during training or fine-tuning during execution in visually challenging circumstances. This work is published as:

Oliver Groth, Chia-Man Hung, Andrea Vedaldi, Ingmar Posner. "Goal-Conditioned End-to-End Visuomotor Control for Versatile Skill Primitives". In: *IEEE International Conference on Robotics and Automation (ICRA)*. June 2021.

Goal-Conditioned End-to-End Visuomotor Control for Versatile Skill Primitives

Oliver Groth[1], Chia-Man Hung[1], Andrea Vedaldi[1], Ingmar Posner[1]

***Abstract*— Visuomotor control (VMC) is an effective means of achieving basic manipulation tasks such as pushing or pick-and-place from raw images. Conditioning VMC on desired goal states is a promising way of achieving versatile *skill primitives*. However, common conditioning schemes either rely on task-specific fine tuning - e.g. using one-shot imitation learning (IL) - or on sampling approaches using a forward model of scene dynamics i.e. model-predictive control (MPC), leaving deployability and planning horizon severely limited. In this paper we propose a conditioning scheme which avoids these pitfalls by learning the controller and its conditioning in an end-to-end manner. Our model predicts complex action sequences based directly on a dynamic image representation of the robot motion and the distance to a given target observation. In contrast to related works, this enables our approach to efficiently perform complex manipulation tasks from raw image observations without predefined control primitives or test time demonstrations. We report significant improvements in task success over representative MPC and IL baselines. We also demonstrate our model's generalisation capabilities in challenging, unseen tasks featuring visual noise, cluttered scenes and unseen object geometries.**

I. Introduction

With recent advances in deep learning, we can now learn robotic controllers end-to-end, mapping directly from raw video streams into a robot's command space. The promise of these approaches is to build real-time visuomotor controllers without the need for complex pipelines or predefined macro-actions (e.g. for grasping). End-to-end visuomotor controllers have demonstrated remarkable performance in real systems, e.g. learning to pick up a cube and place it in a basket [1], [2]. However, a common drawback of current visuomotor controllers is their limited *versatility* due to an often very narrow task definition. For example, in the controllers of [1], [2], which are unconditioned, putting a red cube into a blue basket is a different task than putting a yellow cube into a green basket. In contrast to that, in this paper we consider a broader definition of task and argue that it should rather be treated as a *skill primitive* (e.g. a policy which can pick up any object and place it anywhere else). Such a policy must thus be conditioned on certain arguments, e.g. specifying the object to be moved and its target.

Several schemes have been proposed to condition visuomotor controllers on a *target image*, e.g. an image depicting how a scene should look like after the robot has executed its task. Established conditioning schemes build on various approaches such as model-predictive control [3], task-embedding [4] or meta-learning [5] and are discussed in greater detail in section II. However, the different methods rely on costly sampling techniques, access to prior demonstrations or task-specific fine-tuning during test time restraining their general applicability.

In contrast to prior work, we propose an efficient end-to-end controller which can be conditioned on a single target image without fine-tuning and regresses directly to motor commands of an actuator without any predefined macro-actions. This allows us to learn general *skill primitives*, e.g. pushing and pick-and-place skills, which are versatile enough to immediately generalise to new tasks, i.e. unseen scene setups and objects to handle. Our model utilises *dynamic images* [6] as a succinct representation of the video dynamics in its observation buffer as well as a visual estimation of the difference between its current observation and the target it is supposed to accomplish. Figure 1 depicts an example execution of our visuomotor controller, its conditioning scheme and intermediate observation representations.

In summary, our contributions are three-fold: Firstly, we propose a novel architecture for visuomotor control which can be efficiently conditioned on a new task with just one single target image. Secondly, we demonstrate our model's efficacy by outperforming representative MPC and IL baselines in pushing and pick-and-place tasks by significant margins. Lastly, we analyse the impact of the dynamic image representation in visuomotor control providing beneficial perception invariances to facilitate controller resilience and generalisation without the need of sophisticated domain randomisation schemes during training.

II. Related Work

The problem of *goal conditioning* constitutes a key challenge in visuomotor control: Given a specific task specification (e.g. putting a red cube onto a blue pad), it needs to be communicated to the robot, which in turn must adapt its control policy in such a way that it can carry out the task. In this paper, we focus on goals which are communicated visually, i.e. through images depicting how objects should be placed on a table. Prior methods which have shown impressive real-world results typically involve dedicated sub-modules for perception and planning [7] or are only loosely goal-conditioned, e.g. on a target shape category [8]. We restrict our survey to end-to-end controllers which can be conditioned on a *single target image* and group related work by their condition schemes and action optimisation methods.

*This work is supported by ERC 677195-IDIU and EPSRC EP/M019918/1.

[1]Department of Engineering Science, University of Oxford, United Kingdom. Corresponding author: ogroth@robots.ox.ac.uk

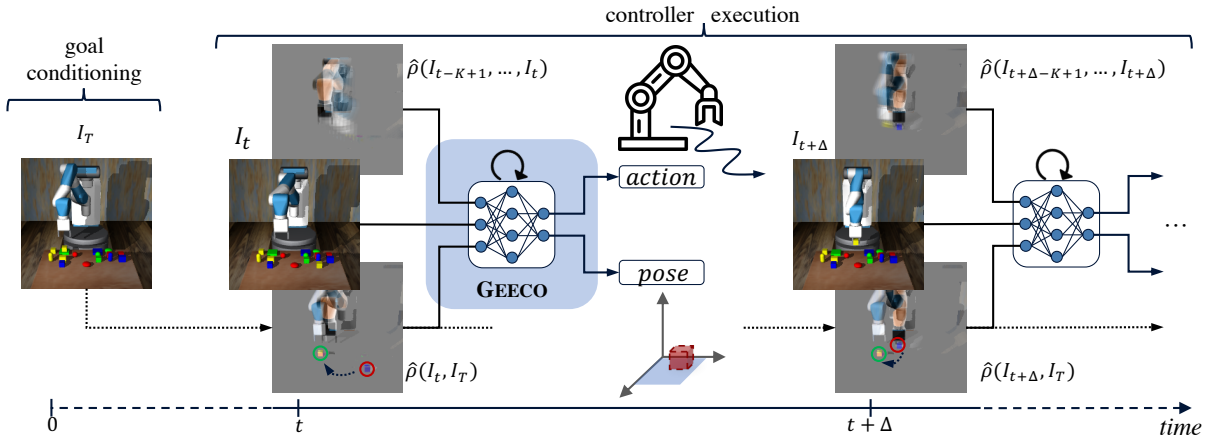


Fig. 1: Our proposed model executes a task given by a target image I_T. In this example, I_T indicates that the small yellow cube from the front right needs to be moved onto the green pad in the back left. Dynamic images are used to (1) represent the difference between the current observation and the target $\hat{\rho}(I_t, I_T)$ and (2) to capture the motion dynamics $\hat{\rho}(I_{t-K+1}, \ldots, I_t)$. Current and target location of the manipulated object are highlighted by red and green circles respectively.

In *visual model-predictive control* one learns a forward model of the world, forecasting the outcome of an action. The learned dynamics model is then explored via sampling or gradient-based methods to compute a sequence of actions which brings the predicted observation closest to a desired goal observation. An established line of work on *Deep Visual Foresight* (VFS) [9], [3], [10], [11] learns action-conditioned video predictors and employs CEM-like [12] sampling methods for trajectory optimisation, successfully applying those models to simulated and real robotic pushing tasks. Instead of low-level video prediction, visual MPC can also be instantiated using higher-level, object-centric models for tasks such as block stacking [13], [14]. Another line of work attempts to learn forward dynamics models in suitable latent spaces. After projecting an observation and a goal image into the latent space, a feasible action sequence can then be computed using gradient-based optimisation methods [15], [16], [17], [18]. Even though MPC approaches have shown promising results in robot manipulation tasks, they are limited by the quality of the forward model and do not scale well due to the action sampling or gradient optimisation procedures required. In contrast to them our model regresses directly to the next command given a buffer of previous observations.

One-Shot Imitation Learning seeks to learn general task representations which are quickly adaptable to unseen setups. MIL [5] is a meta-controller, which requires fine-tuning during test time on one example demonstration of the new task to adapt to it. In contrast to MIL, TecNet [4] learns a task embedding from expert demonstrations and requires at least one demonstration of the new task during test time to modulate its policy according to similar task embeddings seen during training. Additionally, a parallel line of work in that domain operates on discrete action spaces [19], [20] and maps demonstrations of new tasks to known macro actions. Unlike those methods, our model is conditioned on a single target image and does not require any fine-tuning on a new task during test time.

Goal-conditioned reinforcement learning [21] is another established paradigm for learning of control policies. However, due to the unwieldy nature of images as state observations, the use of goal images typically comes with limiting assumptions such as being from a previously observed set [22] or living on the manifold of a learned latent space [23]. Our proposed utilisation of dynamic images for goal conditioning circumvents such limitations and can be seen as complementary to other work which incorporates demonstrations into goal-conditioned policy learning [24], [25] by enabling efficient bootstrapping of a control policy on goal-conditioned demonstrations.

III. Goal-Conditioned Visuomotor Control

In order to build a visuomotor controller which can be efficiently conditioned on a target image and is versatile enough to generalise its learned policy to new tasks immediately, we need to address the following problems: Firstly, we need an efficient way to detect scene changes, i.e. answering the question 'Which object has been moved and where from and to?' Secondly, we want to filter the raw visual observation stream such that we only retain information pertinent to the control task; specifically the motion dynamics of the robot. Drawing inspiration from previous work in VMC and action recognition, we propose *GEECO*, a novel architecture for *goal-conditioned end-to-end control* which combines the idea of *dynamic images* [6] with a robust end-to-end controller network [1] to learn versatile manipulation skill primitives which can be conditioned on new tasks on the fly. We discuss next the individual components.

Dynamic Images. In the domain of action recognition, dynamic images have been developed as a succinct video representation capturing the dynamics of an entire frame sequence in a single image. This enables the treatment of a video with convolutional neural networks as if it was an

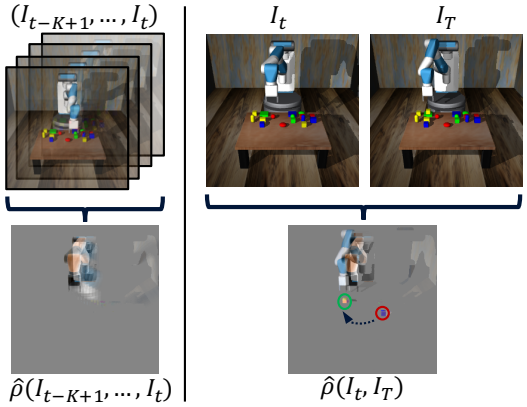


Fig. 2: Utilisation of dynamic images. Left: A dynamic image represents the motion occurring in a sequence of K consecutive RGB observations. Right: A dynamic image represents the changes which occurred between the two images I_t and I_T like the change of object positions as indicated by the red and green circles.

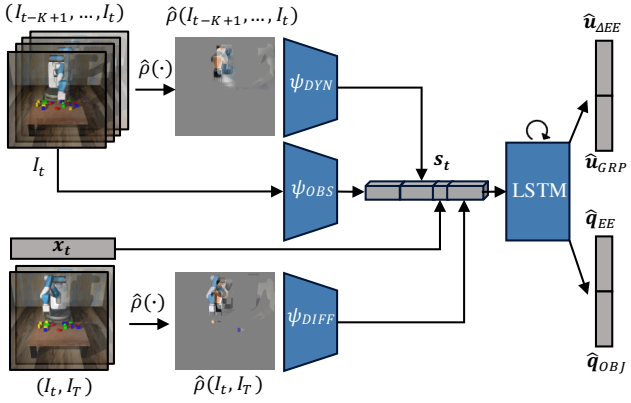


Fig. 3: Network architecture of GEECO-$\mathcal{F}$. The observation buffer $(I_{t-K+1}, \dots, I_T)$ is compressed into a dynamic image via $\hat{\rho}(.)$ and passed through ψ_{DYN}. The current difference to the target frame I_T is also computed via $\hat{\rho}(.)$ and passed through ψ_{DIFF}. Lastly, the current observation I_t is encoded via ψ_{OBS}. All CNNs compute spatial feature maps which are concatenated to the tiled proprioceptive feature $\mathbf{x}_t$. The LSTM's output is decoded into command actions $\hat{\mathbf{u}}_{\Delta EE}$ and $\hat{\mathbf{u}}_{GRP}$ as well as auxiliary pose predictions $\hat{\mathbf{q}}_{EE}$ and $\hat{\mathbf{q}}_{OBJ}$.

ordinary RGB image facilitating dynamics-related feature extraction. The core of the dynamic image representation is a ranking machine which learns to sort the frames of a video temporally [26]. As shown by prior work [6], an approximate linear ranking operator $\hat{\rho}(\cdot)$ can be applied to any sequence of H temporally ordered frames $(I_1, \dots, I_H)$ and any image feature extraction function $\psi(\cdot)$ to obtain a dynamic feature map according to the following eq. (1):

$$\hat{\rho}(I_1, \dots, I_H; \psi) = \sum_{t=1}^{H} \alpha_t \psi(I_t) \qquad (1)$$

$$\alpha_t = 2(H - t + 1) - (H + 1)(\mathcal{H}_H - \mathcal{H}_{t-1}) \qquad (2)$$

Here, $\mathcal{H}_t = \sum_{i=1}^{t} 1/t$ is the t-th Harmonic number and $\mathcal{H}_0 = 0$. Setting $\psi(\cdot)$ to the identity, $\hat{\rho}(I_1, \dots, I_H)$ yields a dynamic image which, after normalisation across all channels, can be treated as a normal RGB image by a downstream network. The employment of dynamic images serves two important purposes in our network, as depicted in fig. 2. Firstly, it compresses a window of the last K RGB observations into one image $\hat{\rho}(I_{t-K+1}, \dots, I_t)$ capturing the current motion of the robot arm. Secondly, given a target image I_T depicting the final state the scene should be in, the dynamic image $\hat{\rho}(I_t, I_T)$ lends itself very naturally to represent the *visual difference* between the current observation I_t and the target state I_T. Another advantage of using dynamic images in these two places is to make the controller network invariant w.r.t. the static scene background and, approximately, the object colour, allowing it to focus on location and geometry of objects involved in the manipulation task.

Observation Buffer. During execution, our network maintains a buffer of most recent K observations as a sequence of pairs $((I_{t-K+1}, \mathbf{x}_{t-K+1}), \dots, (I_t, \mathbf{x_t}))$ where I_t is the RGB frame at time step t and $\mathbf{x_t}$ is the proprioceptive feature of the robot at the same time step represented as a vector of its joint angles. Throughout our experiments we set $K = 4$. The observation buffer breaks long-horizon manipulation trajectories into shorter windows which retain relative independence from each other. This endows the controller with a certain error-correction capacity, e.g. when a grasped object slips from the gripper prematurely, the network can regress back to a pick-up phase.

Goal Conditioning. Before executing a trajectory, our controller is *conditioned* on the task to execute, i.e. moving an object from its initial position to a goal position, via a target image I_T depicting the scene after the task has been carried out. As shown in fig. 2 (right), the dynamic image representation $\hat{\rho}(I_t, I_T)$ helps this inference process by only retaining the two object positions and the difference in the robot pose while cancelling out all static parts of the scene.

Network Architecture. The controller network takes the current observation buffer $((I_{t-K+1}, \mathbf{x}_{t-K+1}), \dots, (I_t, \mathbf{x}_t))$ and the target image I_T as input and regresses to the following two action outputs: (1) The change in Cartesian coordinates of the end effector $\hat{\mathbf{u}}_{\Delta EE}$ and (2) a discrete signal $\hat{\mathbf{u}}_{GRP}$ for the gripper to either open (-1), close $(+1)$ or stay in position (0). Additionally, the controller regresses two auxiliary outputs: the current position of the end effector $\hat{\mathbf{q}}_{EE}$ and of the object to manipulate $\hat{\mathbf{q}}_{OBJ}$, both in absolute Cartesian world coordinates. While the action vectors $\hat{\mathbf{u}}_{\Delta EE}$ and $\hat{\mathbf{u}}_{GRP}$ are directly used to control the robot, the position predictions serve as an auxiliary signal during the supervised training process to encourage the network to learn intermediate representations correlated to the world coordinates. A sketch of our model architecture can be found in fig. 3.

The full model is trained in an end-to-end fashion on

N expert demonstrations of manipulation tasks collected in a simulation environment. Each expert demonstration is a sequence of H time steps indexed by t containing: the RGB frame I_t, the proprioceptive feature $\mathbf{x}_t$, the robot commands $\mathbf{u}^*_{\Delta EE}(t)$, $\mathbf{u}^*_{GRP}(t)$ and the positions of the end effector and the object to manipulate $\mathbf{q}^*_{EE}(t)$, $\mathbf{q}^*_{GRP}(t)$. During training, we minimise the following loss function:

$$\begin{aligned}\mathcal{L} = \sum_{i=1}^{N} \Bigg[\sum_{t=1}^{H-K+1} & \mathrm{MSE}(\hat{\mathbf{u}}_{\Delta EE}(\tau_{i,t}), \mathbf{u}^*_{\Delta EE}(i,t)) \\ & + \mathrm{CCE}(\hat{\mathbf{u}}_{GRP}(\tau_{i,t}), \mathbf{u}^*_{GRP}(i,t)) \\ & + \lambda \Big(\mathrm{MSE}(\hat{\mathbf{q}}_{EE}(\tau_{i,t}), \mathbf{q}^*_{EE}(i,t)) \\ & + \mathrm{MSE}(\hat{\mathbf{q}}_{OBJ}(\tau_{i,t}), \mathbf{q}^*_{OBJ}(i,t)) \Big) \Bigg] \end{aligned} \tag{3}$$

In eq. (3) MSE and CCE are abbreviations of *Mean-Squared Error* and *Categorical Cross-Entropy* respectively. The hyper-parameter λ weighs the auxiliary loss terms for pose prediction. The shorthand notation $\tau_{i,t}$ represents the t-th training window in the i-th expert demonstration of the training dataset comprising of $((I_t, \mathbf{x}_t), \ldots, (I_{t+K-1}, \mathbf{x}_{t+K-1})$; $I_T = I_H)$; $\mathbf{u}^*(i,t)$ and $\mathbf{q}^*(i,t)$ are the corresponding ground truth commands, and $\hat{\mathbf{u}}(\tau_{i,t})$ and $\hat{\mathbf{q}}(\tau_{i,t})$ are shorthand notations for the network predictions on that window. During training we always set the target frame I_T to be the last frame of the expert demonstration I_H.

Model Ablations. We refer to our *full* model as GEECO-$\mathcal{F}$ (or just $\mathcal{F}$ for short) as depicted in fig. 3. However, in order to gauge the effectiveness of our different architecture design decisions, we also consider two ablations of our full model which are briefly described below.

GEECO-$\mathcal{R}$: This ablation has ψ_{DYN} and ψ_{DIFF} removed and ψ_{OBS} is responsible for encoding the current observation I_t and the target image I_T and the feature distance is used for goal conditioning. Thus, the state tensor becomes $\mathbf{s}_t = \psi_{OBS}(I_t) \oplus \mathbf{x}_t \oplus (\psi_{OBS}(I_T) - \psi_{OBS}(I_t))$, where $\oplus$ denotes concatenation along the channel dimension. This *residual* state encoding serves as a baseline for learning meaningful goal distances in the feature space induced by ψ_{OBS}.

GEECO-$\mathcal{D}$: This ablation has only the 'motion branch' ψ_{DYN} removed. The state tensor is comprised of $\mathbf{s}_t = \psi_{OBS}(I_t) \oplus \mathbf{x}_t \oplus \psi_{DIFF}(\hat{\rho}(I_t, I_T))$. This gauges the effectiveness of using an explicitly shaped goal difference function over an implicitly learned one like in GEECO-$\mathcal{R}$.

IV. Experiments

Our experimental design is guided by the following questions: (1) How do the learned skill primitives compare to representative MPC and IL approaches? (2) Can our controller deliver on its aspired *versatility* by transferring its skills, acquired only on simple cubes, to novel shapes, tasks and adverse conditions?

Experimental Setup and Data Collection. We have designed a simulation environment, Goal2Cube2, containing four different tasks to train and evaluate our controller on which are presented in fig. 4. In each task one of the small cubes needs to be moved onto one of the larger target pads. The scenario is designed such that the task is ambiguous and the controller needs to infer the object to manipulate and the target location from a given target image depicting the task to perform. We use the MuJoCo physics engine [27] for simulation. We adapt the Gym environment [28] provided by [29] featuring a model of a *Fetch Mobile Manipulator* [30] with a 7-DoF arm and a 2-point gripper[1]. For each skill, i.e. pushing and pick-and-place, we collect 4,000 unique expert demonstrations successful task completions in simulation according to a pre-computed plan. At the start of each demonstration, all object positions and the joint positions of the arm are randomised. Each demonstration is four seconds long and is recorded as a list of observation-action tuples at 25 Hz resulting in an episode length of $H = 100$.

Baselines for Goal-Conditioned VMC. Our first baseline, VFS [3], is a visual MPC which runs on a video prediction backbone. A CEM-based [12] action sampler proposes command sequences over a short time horizon which are evaluated by the video predictor. The sequence which results in a predicted observation closest to the goal image is executed and the process is repeated until termination. We use SAVP [31] as action-conditioned video predictor and train it on our datasets with the hyper-parameters reported for the BAIR robot-pushing dataset [32]. Since the time horizon of our scenarios is much longer than SAVP's prediction horizon, we estimate an upper-bound of the performance of VFS by providing intermediate goal images.[2] Our second baseline, TecNet [4], is an IL model which is capable of quick adaptation given the demonstration of a new task. The demonstration can be as sparse as start and end image of an executed trajectory making it applicable to our setup. We employ TecNet in its one-shot imitation configuration.

Training and Evaluation Protocol. For each skill, we split the demonstrations into training, validation and test sets with a ratio of 2 : 1 : 1 respectively while keeping the task distributions balanced. We train all models for 300k gradient update steps using the Adam optimiser [33]. After each epoch, we evaluate the controller performance on the validation set and select the best performing model checkpoints for the final evaluation on the test set. During task execution, we monitor the following performance metrics: (1) REACH: If the robot touches the object it is supposed to manipulate at least once during the episode, we count this as a reaching success. (2) PICK: If the palm of the robot's gripper touches the correct object at least once during the episode while the fingers are closed, we count this as a picking success. (3) PUSH / PLACE: If by the end of the episode the correct object sits on the designated goal pad, we count this as a task success in the respective skill. Each evaluation episode is terminated after 200 timesteps (8 seconds).

Basic Manipulation Results. We start our investigation

[1]Despite the robot having a mobile platform, its base is fixed during all experiments.

[2]HVF [10] employs a similar 'ground truth bottleneck' scheme to upper-bound a visual MPC baseline.

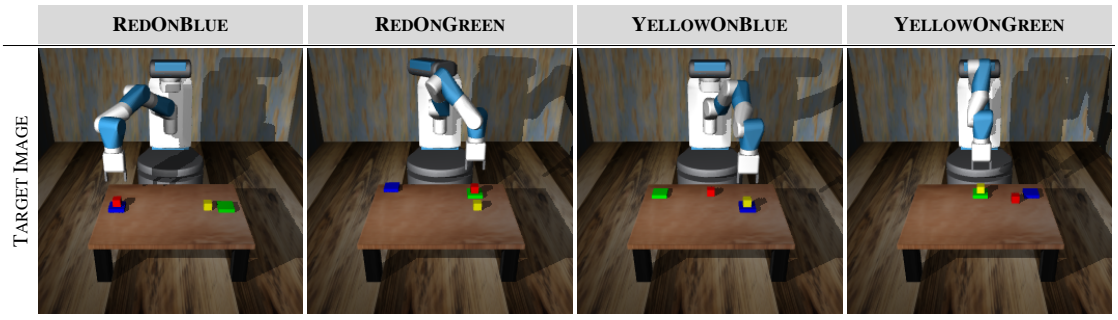


Fig. 4: Basic manipulation tasks in GOAL2CUBE2. In each task, one of the small cubes (red or yellow) needs to be moved onto one of the target pads (blue or green). The tasks can be accomplished via a pushing or pick-and-place manipulation (target pads are reduced to flat textures for pushing).

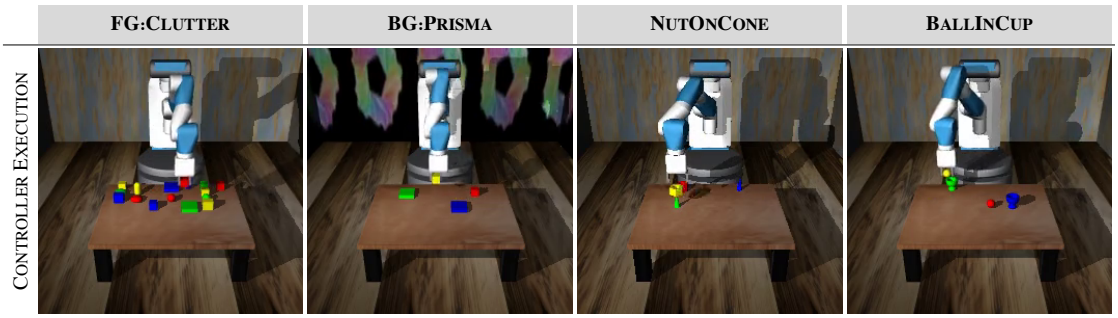


Fig. 5: Generalisation experiments. FG:CLUTTER and BG:PRISMA are variations of GOAL2CUBE2 with severe physical and visual distractions. The background in BG:PRISMA is a video looping rainbow colour patterns. NUTONCONE and BALLINCUP require the application of the pick-and-place skill to novel shapes and target configurations.

by comparing the performance of our proposed model and its ablations to VFS and TECNET in the GOAL2CUBE2 scenario and report the results in table I. We treat the tasks (REACH, PUSH, PICK, PLACE) as Bernoulli experiments and report their mean success rates for 1,000 trials as well as their binomial proportion confidence intervals at 0.95. For pushing tasks we measure nearly perfect reaching and very strong pushing performance for GEECO-$\{\mathcal{R}, \mathcal{D}, \mathcal{F}\}$ outperforming the baselines by up to 80% on final task success. We also observe that $\mathcal{D}$ and $\mathcal{F}$, models which employ the dynamic image representation, perform significantly better than the RGB-only ablation $\mathcal{R}$. Likewise, the general ranking of models also applies to pick-and-place tasks with $\mathcal{D}$ outperforming the baselines by up to 60% task success. The almost complete failure of VFS and TECNET for a multi-stage task like pick-and-place is unsurprising given that both models have been originally designed for reaching and pushing tasks. Qualitatively, we observe that VFS reaches the correct object relatively reliably due to its powerful video predictor but struggles to maintain a firm grasp on an object due to its stochastic action sampling. TECNET fares better in that regard since it is a feedforward regression network like GEECO and can maintain a stable grasp. However, it often approaches the wrong object to manipulate due to the fact that its policy is modulated by the embedding of similar tasks. When confronted with subtle task variations like a colour change in a small object the RGB task embedding becomes less informative and TECNET is prone to inferring the wrong task. An investigation into $\mathcal{F}$'s inferior PLACE performance compared to $\mathcal{D}$ reveals a failure mode of $\mathcal{F}$: The controller sometimes struggles to drop a cube above the target pad presumably due to ambiguity in its depth perception. This suggests that the signal provided by the dynamic frame buffer at relatively motion-less pivot points can be ambiguous without additional treatment. When comparing the versions of GEECO which are trained without auxiliary pose supervision ($\lambda = 0.0$) to their fully supervised counterparts ($\lambda = 1.0$), we observe only mild drops in mean performance of up to 15%. Interestingly, the RGB-only ablation $\mathcal{R}$ is least affected by the pose supervision and even improves performance when trained without auxiliary poses. We hypothesise that this is due to the fact that, in the relatively static simulation, RGB features are very representative of their spatial location. Generally, we conclude from the pose supervision ablation that GEECO's performance is not entirely dependent on accurate pose supervision enabling it to be trained on even fewer data annotations.

Generalisation to New Scenarios. After validating the efficacy of our proposed approach in basic manipulation scenarios which are close to the training distribution, we investigate its robustness and versatility in two additional sets of experiments. In the following trials we take models $\mathcal{R}$, $\mathcal{D}$

	PUSHING		PICK-AND-PLACE		
	REACH [%]	PUSH [%]	REACH [%]	PICK [%]	PLACE [%]
VFS[3] [3]	66.00 ± 9.28	7.00 ± 5.00	87.00 ± 6.59	40.00 ± 9.60	0.00 ± 0.00
TECNET [4]	54.10 ± 3.09	15.70 ± 2.25	32.00 ± 2.89	15.70 ± 2.25	0.80 ± 0.55
$\mathcal{R}, \lambda = 0.0$	98.70 ± 0.70	79.80 ± 2.49	86.20 ± 2.14	58.90 ± 3.05	42.50 ± 3.06
$\mathcal{D}, \lambda = 0.0$	98.70 ± 0.70	***88.50* ± *1.98***	***95.40* ± *1.30***	65.80 ± 2.94	54.20 ± 3.09
$\mathcal{F}, \lambda = 0.0$	98.00 ± 0.87	78.60 ± 2.54	92.70 ± 1.61	71.00 ± 2.81	45.60 ± 3.09
$\mathcal{R}, \lambda = 1.0$	***98.90* ± *0.65***	79.80 ± 2.49	84.90 ± 2.22	50.40 ± 3.10	33.70 ± 2.93
$\mathcal{D}, \lambda = 1.0$	***99.30* ± *0.52***	***86.60* ± *2.11***	**96.20 ± 1.19**	**79.90 ± 2.48**	**61.40 ± 3.02**
$\mathcal{F}, \lambda = 1.0$	**99.80 ± 0.28**	**89.30 ± 1.92**	***94.80* ± *1.38***	***78.40* ± *2.55***	46.30 ± 3.09

TABLE I: Success rates and confidence intervals for pushing and pick-and-place tasks in the GOAL2CUBE2 scenarios. Best mean performances are bold-faced. Models whose CIs overlap with the best-performing one, are additionally italicised.

	FG:CLUTTER			BG:PRISMA		
	REACH [%]	PICK [%]	PLACE [%]	REACH [%]	PICK [%]	PLACE [%]
$\mathcal{R}, \lambda = 1.0$	63.80 ± 2.98	29.40 ± 2.82	15.80 ± 2.26	0.50 ± 0.44	0.10 ± 0.20	0.00 ± 0.00
$\mathcal{D}, \lambda = 1.0$	77.00 ± 2.61	42.10 ± 3.06	20.70 ± 2.51	**94.10 ± 1.46**	**66.00 ± 2.94**	**26.50 ± 2.74**
$\mathcal{F}, \lambda = 1.0$	**85.50 ± 2.18**	**62.60 ± 3.00**	**32.60 ± 2.91**	***93.40* ± *1.54***	***62.40* ± *3.00***	19.30 ± 2.45
	NUTONCONE			BALLINCUP		
	REACH [%]	PICK [%]	PLACE [%]	REACH [%]	PICK [%]	PLACE [%]
$\mathcal{R}, \lambda = 1.0$	32.30 ± 2.90	6.90 ± 1.57	0.40 ± 0.39	21.50 ± 2.55	3.90 ± 1.20	0.10 ± 0.20
$\mathcal{D}, \lambda = 1.0$	***66.60* ± *2.92***	***23.30* ± *2.62***	3.30 ± 1.11	***50.60* ± *3.10***	9.70 ± 1.83	0.20 ± 0.28
$\mathcal{F}, \lambda = 1.0$	**72.20 ± 2.78**	**26.90 ± 2.75**	**6.20 ± 1.49**	**54.60 ± 3.09**	**16.30 ± 2.29**	**1.50 ± 0.75**

TABLE II: Pick-and-place success rates and confidence intervals of GEECO models trained on GOAL2CUBE2 and employed in novel scenarios as depicted in fig. 5. The semantics of bold-faced results are the same as in table I.

and $\mathcal{F}$ which have been trained on pick-and-place tasks of GOAL2CUBE2 and apply them to new scenarios probing different aspects of generalisation. We present examples of the four new scenarios in fig. 5 and present quantitative results for 1,000 trials in table II. FG:CLUTTER and BG:PRISMA evaluate whether the pick-and-place skill learned by GEECO is robust enough to be executed in visually challenging circumstances as well. The results reveal that the employment of dynamic images for target difference ($\mathcal{D}$) and additionally buffer representation ($\mathcal{F}$) significantly improves task success over the RGB-baseline ($\mathcal{R}$) in the cluttered tabletop scenario due to the perceptual invariances afforded by the dynamic image representation. The effect is even more apparent when the colours of the scene background are distorted. This leads to a complete failure of $\mathcal{R}$ (which is now chasing after flickering colour patterns in the background) while $\mathcal{D}$ and $\mathcal{F}$ can still accomplish the task in about 20% of the cases. In the second set of experiments (cf. table II, bottom), we evaluate whether the pick-and-place skill is versatile enough to be immediately applicable to new shapes and target configurations. NUTONCONE requires to drop a nut onto a cone such that the cone is almost entirely hidden. Conversely, BALLINCUP is a coarse-grained insertion task requiring a ball to be dropped into a cup such that only a fraction of its surface remains visible. Besides the handling of unseen object geometries, both tasks also pose novel challenges in terms of task inference because they feature much heavier occlusions than the original GOAL2CUBE2 dataset which the model was trained on. Our full model, $\mathcal{F}$, outperforms the ablations significantly in both new scenarios. Despite the relatively low PLACE success rates on NUTONCONE and BALLINCUP, it is encouraging that the model is still able to handle entirely unseen object geometries and task setups without *any* domain randomisation or fine-tuning on the novel scenarios.

[3]Due to computational constraints, VFS has only been tested on 100 tasks from the test set.

V. CONCLUSIONS

We introduce GEECO, a novel architecture for goal-conditioned end-to-end visuomotor control utilising dynamic images. GEECO can be immediately conditioned on a new task with the input of a single target image. We demonstrate GEECO's efficacy in complex pushing and pick-and-place tasks involving multiple objects. It also generalises well to challenging, unseen scenarios maintaining strong task performance even when confronted with heavy clutter, visual distortions or novel object geometries. Due to its built-in visual invariances, our model can also complement Sim2Real approaches by reducing the dependency on sophisticated randomisation schemes during simulation pre-training of visuomotor controllers. Our results suggest that GEECO can serve as a robust model to efficiently learn coarse-grained manipulation skill primitives like pushing and pick-and-place of rigid bodies from visual demonstrations.

REFERENCES

[1] S. James, A. J. Davison, and E. Johns, "Transferring end-to-end visuomotor control from simulation to real world for a multi-stage task," in *Conference on Robot Learning*, 2017, pp. 334–343.

[2] Y. Zhu, Z. Wang, J. Merel, A. Rusu, T. Erez, S. Cabi, S. Tunyasuvunakool, J. Kramár, R. Hadsell, N. de Freitas, *et al.*, "Reinforcement and imitation learning for diverse visuomotor skills," in *Proceedings of Robotics: Science and Systems*, Pittsburgh, Pennsylvania, June 2018.

[3] F. Ebert, C. Finn, S. Dasari, A. Xie, A. Lee, and S. Levine, "Visual foresight: Model-based deep reinforcement learning for vision-based robotic control," *arXiv preprint arXiv:1812.00568*, 2018.

[4] S. James, M. Bloesch, and A. J. Davison, "Task-embedded control networks for few-shot imitation learning," in *Conference on Robot Learning*, 2018, pp. 783–795.

[5] C. Finn, T. Yu, T. Zhang, P. Abbeel, and S. Levine, "One-shot visual imitation learning via meta-learning," in *Conference on Robot Learning*, 2017, pp. 357–368.

[6] H. Bilen, B. Fernando, E. Gavves, and A. Vedaldi, "Action recognition with dynamic image networks," *IEEE transactions on pattern analysis and machine intelligence*, vol. 40, no. 12, pp. 2799–2813, 2017.

[7] Y. Labbé, S. Zagoruyko, I. Kalevatykh, I. Laptev, J. Carpentier, M. Aubry, and J. Sivic, "Monte-carlo tree search for efficient visually guided rearrangement planning," *IEEE Robotics and Automation Letters*, vol. 5, no. 2, pp. 3715–3722, 2020.

[8] A. Pashevich, I. Kalevatykh, I. Laptev, and C. Schmid, "Learning visual policies for building 3d shape categories," *arXiv preprint arXiv:2004.07950*, 2020.

[9] C. Finn and S. Levine, "Deep visual foresight for planning robot motion," in *2017 IEEE International Conference on Robotics and Automation (ICRA)*, May 2017, pp. 2786–2793.

[10] S. Nair and C. Finn, "Hierarchical foresight: Self-supervised learning of long-horizon tasks via visual subgoal generation," *arXiv preprint arXiv:1909.05829*, 2019.

[11] A. Xie, F. Ebert, S. Levine, and C. Finn, "Improvisation through physical understanding: Using novel objects as tools with visual foresight," *arXiv preprint arXiv:1904.05538*, 2019.

[12] R. Y. Rubinstein and D. P. Kroese, *The Cross Entropy Method: A Unified Approach To Combinatorial Optimization, Monte-Carlo Simulation (Information Science and Statistics)*. Berlin, Heidelberg: Springer-Verlag, 2004.

[13] Y. Ye, D. Gandhi, A. Gupta, and S. Tulsiani, "Object-centric forward modeling for model predictive control," in *Conference on Robot Learning*, 2020, pp. 100–109.

[14] M. Janner, S. Levine, W. T. Freeman, J. B. Tenenbaum, C. Finn, and J. Wu, "Reasoning about physical interactions with object-oriented prediction and planning," in *International Conference on Learning Representations*, 2019.

[15] M. Watter, J. Springenberg, J. Boedecker, and M. Riedmiller, "Embed to control: A locally linear latent dynamics model for control from raw images," in *Advances in neural information processing systems*, 2015, pp. 2746–2754.

[16] A. Byravan, F. Lceb, F. Meier, and D. Fox, "Se3-pose-nets: Structured deep dynamics models for visuomotor control," in *2018 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2018, pp. 1–8.

[17] A. Srinivas, A. Jabri, P. Abbeel, S. Levine, and C. Finn, "Universal planning networks," in *International Conference on Machine Learning (ICML)*, 2018.

[18] T. Yu, G. Shevchuk, D. Sadigh, and C. Finn, "Unsupervised visuomotor control through distributional planning networks," in *Proceedings of Robotics: Science and Systems*, Freiburg im Breisgau, Germany, June 2019.

[19] D. Xu, S. Nair, Y. Zhu, J. Gao, A. Garg, L. Fei-Fei, and S. Savarese, "Neural task programming: Learning to generalize across hierarchical tasks," in *2018 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2018, pp. 1–8.

[20] D.-A. Huang, S. Nair, D. Xu, Y. Zhu, A. Garg, L. Fei-Fei, S. Savarese, and J. C. Niebles, "Neural task graphs: Generalizing to unseen tasks from a single video demonstration," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2019, pp. 8565–8574.

[21] L. P. Kaelbling, "Learning to achieve goals," in *IJCAI*. Citeseer, 1993, pp. 1094–1099.

[22] D. Warde-Farley, T. V. de Wiele, T. Kulkarni, C. Ionescu, S. Hansen, and V. Mnih, "Unsupervised control through non-parametric discriminative rewards," in *International Conference on Learning Representations*, 2019. [Online]. Available: https://openreview.net/forum?id=r1eVMnA9K7

[23] A. V. Nair, V. Pong, M. Dalal, S. Bahl, S. Lin, and S. Levine, "Visual reinforcement learning with imagined goals," in *Advances in Neural Information Processing Systems*, 2018, pp. 9191–9200.

[24] A. Nair, B. McGrew, M. Andrychowicz, W. Zaremba, and P. Abbeel, "Overcoming exploration in reinforcement learning with demonstrations," in *2018 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2018, pp. 6292–6299.

[25] Y. Ding, C. Florensa, P. Abbeel, and M. Phielipp, "Goal-conditioned imitation learning," in *Advances in Neural Information Processing Systems*, 2019, pp. 15 324–15 335.

[26] B. Fernando, E. Gavves, M. José Oramas, A. Ghodrati, and T. Tuytelaars, "Modeling video evolution for action recognition," in *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2015, pp. 5378–5387.

[27] E. Todorov, T. Erez, and Y. Tassa, "Mujoco: A physics engine for model-based control," in *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, 2012, pp. 5026–5033.

[28] G. Brockman, V. Cheung, L. Pettersson, J. Schneider, J. Schulman, J. Tang, and W. Zaremba, "Openai gym," 2016.

[29] Y. Duan, M. Andrychowicz, B. Stadie, O. J. Ho, J. Schneider, I. Sutskever, P. Abbeel, and W. Zaremba, "One-shot imitation learning," in *Advances in neural information processing systems*, 2017, pp. 1087–1098.

[30] M. Wise, M. Ferguson, D. King, E. Diehr, and D. Dymesich, "Fetch & freight: Standard platforms for service robot applications," 2018. [Online]. Available: https://fetchrobotics.com/wp-content/uploads/2018/04/Fetch-and-Freight-Workshop-Paper.pdf

[31] A. X. Lee, R. Zhang, F. Ebert, P. Abbeel, C. Finn, and S. Levine, "Stochastic adversarial video prediction," *arXiv preprint arXiv:1804.01523*, 2018.

[32] F. Ebert, C. Finn, A. X. Lee, and S. Levine, "Self-supervised visual planning with temporal skip connections," in *Conference on Robot Learning*, 2017.

[33] 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. [Online]. Available: http://arxiv.org/abs/1412.6980

Appendix

5.A GEECO Hyperparameters

In this section, we present additional details regarding the architecture and training hyper-parameters of GEECO and all its ablations.

Observation Buffer. The observation buffer consists of pairs $(I_j, \mathbf{x}_j),\ j \in [t - K + 1, \ldots, t]$ of images I_j and proprioceptive features $\mathbf{x}_j$ representing the K most recent observations of the model up to the current time step t. The images are RGB with a resolution of 256×256 and the proprioceptive feature is a vector of length seven containing the angles of the robot's seven joints at the respective time step. We have experimented with frame buffer sizes $K \in \{2, 4, 6, 8\}$. Buffer sizes smaller than four result in too coarse approximations of dynamics (because velocities have to be inferred from just two time steps) and consequently in lower controller performance. However, controller performance also does not seem to improve with buffer sizes greater than four. We assume that in our scenarios, four frames are sufficient to capture the robot's motions accurately enough, which is in line with similar experiments in prior work [42]. Therefore, we keep the buffer hyper-parameter $K = 4$ fixed in all our experiments. At the start of the execution of the controller, we pad the observation buffer to the left with copies of the oldest frame, if there are less than K pairs in the buffer assuming that the robot is always starting from complete rest.

Convolutional Encoder. All convolutional encoders used in the GEECO architecture have the same structure, which is outlined in table 5.1. However, the parameters between the convolutional encoders are not shared. The rationale behind

this decision is that the different stacks of convolutions are processing semantically different inputs: ψ_{OBS} processes raw RGB observations, ψ_{DYN} processes dynamic images representing the motion captured in the observation buffer and ψ_{DIFF} processes the dynamic image difference between the current observation and the target image.

Layer	Filters	Kernel	Stride	Activation
Conv1	32	3	1	ReLU
Conv2	48	3	2	ReLU
Conv3	64	3	2	ReLU
Conv4	128	3	2	ReLU
Conv5	192	3	2	ReLU
Conv6	256	3	2	ReLU
Conv7	256	3	2	ReLU
Conv8	256	3	2	ReLU

Table 5.1: The convolutional encoders used in GEECO all share the same structure of eight consecutive layers of 2D convolutions. They take as inputs RGB images with a resolution of 256×256 and return spatial feature maps with a shape of $2 \times 2 \times 256$.

LSTM Decoder. The spatial feature maps $\psi_{OBS}(I_t)$, $\psi_{DYN}(\hat{\rho}(I_{t-K+1}, \ldots, I_t))$, $\psi_{DIFF}(\hat{\rho}(I_t, I_T))$ obtained from the convolutional encoders are concatenated to the proprioceptive feature $\mathbf{x}_t$ containing the current joint angles for the robot's 7 DoF. This concatenated tensor forms the state representation $\mathbf{s}_t$, which, in the full model GEECO-$\mathcal{F}$, has a shape of $2 \times 2 \times (256 + 256 + 7 + 256)$. The state is subsequently fed into an LSTM. The LSTM has a hidden state $\mathbf{h}$ of size 128 and produces an output vector $\mathbf{o}_t$ of the same dimension at each time step. As shown in prior work [42], maintaining an internal state in the network is crucial for performing multi-stage tasks such as pick-and-place.

At the beginning of each task, i.e. when the target image I_T is set and before the first action is executed, the LSTM state is initialised with a zero vector. The output $\mathbf{o}_t$ at each timestep is passed through a fully connected layer $\phi(\cdot)$ with 128 neurons and a ReLU activation function. This last-layer feature $\phi(\mathbf{o}_t)$ is finally passed through four parallel, fully-connected decoding heads without an activation

function to obtain the command vectors and the auxiliary position estimates for the object and the end effector as described in table 5.2.

Head	Units	Output
$\hat{\mathbf{u}}_{\Delta EE}$	3	change in EE position $(\Delta x, \Delta y, \Delta z)$
$\hat{\mathbf{u}}_{GRP}$	3	logits for {open, noop, close}
$\hat{\mathbf{q}}_{EE}$	3	absolute EE position (x, y, z)
$\hat{\mathbf{q}}_{POS}$	3	absolute OBJ position (x, y, z)

Table 5.2: The output heads of the LSTM decoder regressing to the commands and auxiliary position estimates.

Training Details. We train all versions of GEECO with a batch size of 32 for 300k gradient steps using the Adam optimiser [47] with a start learning rate of 1e-4. One training run takes approximately 48 hours to complete using a single NVIDIA GTX 1080 Ti with 11 GB of memory.

Execution Time. Running one simulated trial with an episode length of eight seconds takes about ten seconds for any version of GEECO using a single NVIDIA GTX 1080 Ti. This timing includes the computational overhead for running and rendering the physics simulation resulting in a lower-bound estimate of GEECO's control frequency at 20 Hz. This indicates that our model is nearly real-time capable of continuous control without major modifications.

5.B GEECO Ablation Details

GEECO-$\mathcal{R}$ Our first ablation, which is presented in fig. 5.1, uses a naïve *residual* target encoding to represent the distance to a given target observation in feature space. The residual feature is the difference $\psi_{OBS}(I_T) - \psi_{OBS}(I_j),\ j \in [t-K+1, \ldots, t]$ and should tend towards zero as the observation I_j approaches the target image I_T. Since the same encoder ψ_{OBS} is used for observation and target image, this architecture should encourage the formation of a feature space which captures the difference between an observation and the target image in a semantically meaningful

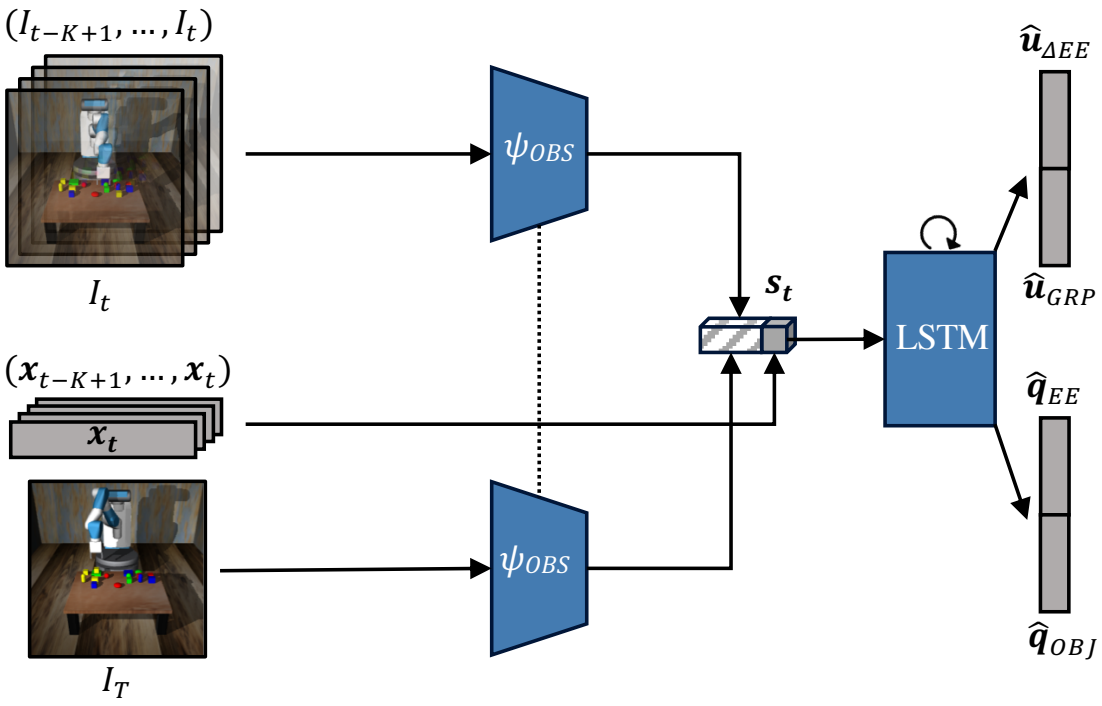


Figure 5.1: Model architecture of GEECO-$\mathcal{R}$. The same encoder ψ_{OBS} is used for RGB observations I_t and the target frame I_T. For each observed image I_t, the residual feature w.r.t. to the target image I_T is computed as $\psi_{OBS}(I_T) - \psi_{OBS}(I_t)$, indicated by the striped box in $\mathbf{s}_t$.

way. Since the observation buffer is not compressed into a dynamic image via $\hat{\rho}(\cdot)$, it is processed slightly differently in order to retain information about the motion dynamics. For each pair $(I_j, \mathbf{x}_j),\ j \in [t-K+1, \ldots, t]$ containing an observed image and a proprioceptive feature at time step j, the corresponding state representation $\mathbf{s}_j$ is computed and fed into the LSTM which, in turn, updates its state. However, only after all K pairs of the observation buffer have been fed, the command outputs are decoded from the LSTM's last output vector. This delegates the task of inferring motion dynamics to the LSTM as it processes the observation buffer.

GEECO-$\mathcal{D}$ Our second ablation, which is presented in fig. 5.2, uses the dynamic image operator $\hat{\rho}(\cdot)$ to compute the difference between each observed frame $I_j,\ j \in [t-K+1, \ldots, t]$ and the target image I_T as opposed to GEECO-$\mathcal{R}$ which represents the difference only in feature space. Since the *dynamic difference* $\hat{\rho}(I_t, I_T)$ is semantically different from a normal RGB observation, it is processed with a

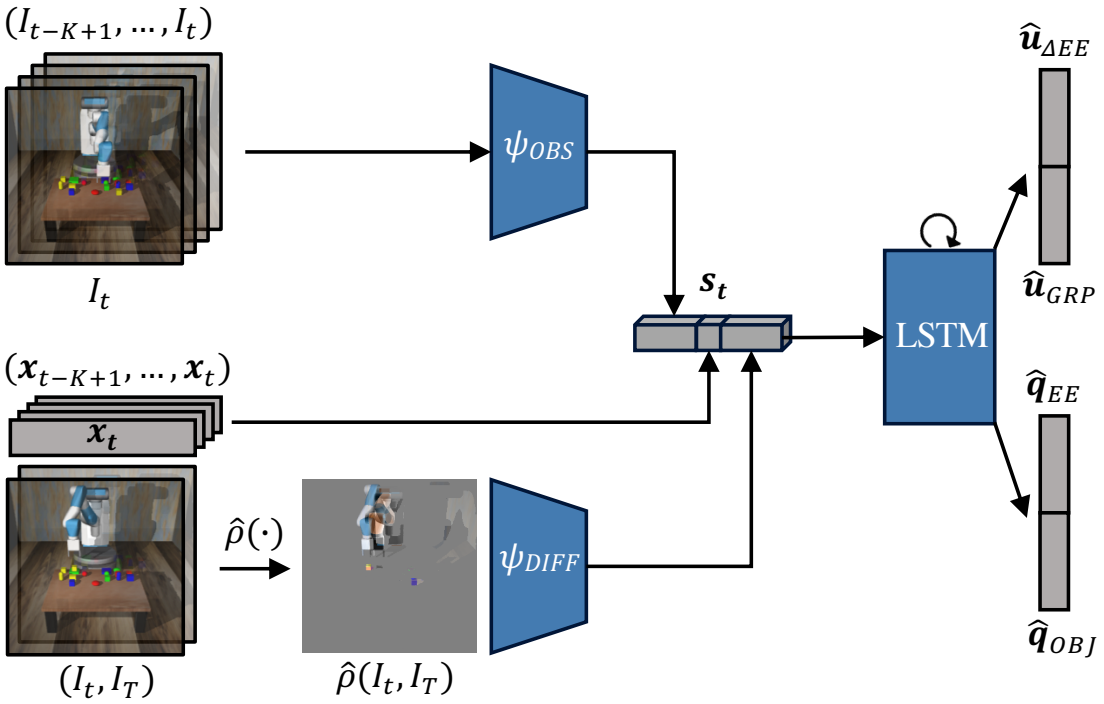


Figure 5.2: Model architecture of GEECO-$\mathcal{D}$. For each image I_t in the observation buffer, the *dynamic difference* to the target image I_T is computed using $\hat{\rho}(\cdot)$. The difference image $\hat{\rho}(I_t, I_T)$ is encoded with ψ_{DIFF} before being concatenated to $\mathbf{s}_t$.

dedicated convolutional encoder ψ_{DIFF} and the resulting feature is concantenated to the state representation $\mathbf{s}_t$. In order to also capture motion dynamics, the observation buffer is processed sequentially like in GEECO-$\mathcal{R}$ before a control command is issued.

5.C E2EVMC Baseline

We compare GEECO to E2EVMC [42], an unconditioned visuomotor controller, which we have implemented according to the original paper. We have re-created a similar environment like in the original paper featuring only a red cube and a blue target pad which we call Goal1Cube1. This dataset also consists of 4,000 demonstrations per skill and is split into training, validation and test sets with a ratio of 2 : 1 : 1. Training and testing on this scenario is done to ensure the correct functionality of the model architecture and verify that GEECO performs at least as well as an unconditioned controller in an unambiguous scenario. Even though the task is always the same, i.e. the red cube always goes on top of the blue pad,

we still provide GEECO with the target image in every trial. The unconditioned baseline, E2EVMC, runs without a target image since it is trained to perform only one task. We train E2EVMC exactly like GEECO (cf. section 5.A: Training Details) and select the best model snapshots according to the task performance on the respective validation sets. We present experimental results for 1,000 trials on the test set of GOAL1CUBE1 in table 5.3.

MODEL	PUSHING		PICK-AND-PLACE		
	REACH [%]	PUSH [%]	REACH [%]	PICK [%]	PLACE [%]
E2EVMC [42]	95.20 ± 1.32	58.60 ± 3.05	**96.10 ± 1.20**	72.00 ± 2.78	**67.60 ± 2.90**
$\mathcal{R}$	**98.80 ± 0.67**	43.70 ± 3.07	**95.30 ± 1.31**	73.00 ± 2.75	60.70 ± 3.03
$\mathcal{D}$	**99.50 ± 0.44**	**87.40 ± 2.06**	**95.80 ± 1.24**	77.40 ± 2.59	**64.90 ± 2.96**
$\mathcal{F}$	**99.20 ± 0.55**	72.90 ± 2.75	**95.90 ± 1.23**	**83.30 ± 2.31**	61.20 ± 3.02

Table 5.3: Comparison of pushing and pick-and-place performance of all versions of GEECO with E2EVMC on GOAL1CUBE1. Best task performances are bold-faced.

We observe that GEECO-$\mathcal{D}$ and -$\mathcal{F}$ perform commensurately with E2EVMC for both pushing and pick-and-place tasks. Both E2EVMC and GEECO reach the red cube nearly perfectly with at least 95% success rate. GEECO-$\mathcal{D}$ performs best on this dataset even outperforming E2EVMC by almost 30% mean success rate for pushing tasks. Again, GEECO-$\mathcal{F}$ sometimes exhibits its failure mode at the pivot point between moving and dropping phase presumably due to ambiguous or uninformative signals from the motion representation around this phase.

5.D Visual Foresight Baseline

In this section, we explain all hyper-parameters which have been used during training and evaluation of the Visual Foresight baseline [14].

Video Predictor. We use the official implementation[1] of Stochastic Adversarial Video Prediction (SAVP) [59] as the video prediction backbone of Visual Foresight. We have not been able to fit the model at a resolution of 256×256 on a single

[1] `https://github.com/alexlee-gk/video_prediction`

GPU with 11 GB of memory. Hence, we adjusted the image resolution of the video predictor to 128×128 pixels. We use SAVP's hyper-parameter set which is reported for the BAIR robot pushing dataset [15] since those scenarios resemble our training setup most closely. We report the hyper-parameter setup in table 5.4.

Parameter	Value	Description
`scale_size`	128	image resolution
`use_state`	True	use action conditioning
`sequence_length`	13	prediction horizon
`frame_skip`	0	use entire video
`time_shift`	0	use original frame rate
`l1_weight`	1.0	use L_1 reconstruction loss
`kl_weight`	0.0	make model deterministic
`state_weight`	1e-4	weight of conditioning loss

Table 5.4: Hyper-parameter setup of SAVP. Hyper-parameters not listed here are kept at their respective default values.

Training Details. We train SAVP with a batch size of 11 for 300k gradient steps using the Adam optimiser [47] with a start learning rate of 1e-4. One training run takes approximately 72 hours to complete using a single NVIDIA GTX 1080 Ti with 11 GB of memory.

Action Sampling. We use CEM [80] as in the original VFS paper [14] to sample actions which bring the scene closer to a desired target image under the video prediction model. We set the *planning horizon* of VFS to the prediction length of SAVP, $P = 13$. The action space is identical to the one used in GEECO and consists of a continuous vector representing the position change in the end effector $\mathbf{u}_{\Delta EE} \in \mathbb{R}^3$ and a discrete command for the gripper $\mathbf{u}_{GRP} \in \{-1, 0, 1\}$. Once a target image has been set, we sample action sequences of length P according to the following eqs. (5.1) and (5.2):

$$\mathbf{u}_{\Delta EE}^{1:\,P} \sim \mathcal{N}(\mu, \Sigma) \tag{5.1}$$

$$\mathbf{u}_{GRP}^{1:\,P} \sim \mathcal{U}\{-1, 0, 1\} \tag{5.2}$$

where $\mathcal{N}(\mu, \Sigma)$ is a multi-variate Gaussian distribution and $\mathcal{U}\{-1, 0, 1\}$ is a uniform distribution over the gripper states. For each planning step, we run CEM for four iterations drawing 200 samples at each step and re-fit the distributions to the ten best action sequences according to the video predictor, i.e. the action sequences which transform the scene closest to the next goal image. Finally, we execute the best action sequence yielded from the last CEM iteration and re-plan after P steps.

Goal Distance. We use L_2 distance in image space to determine the distance between an image forecast by the video predictor and a target image (cf. [14]). Since this goal distance is dominated by large image regions (e.g. the robot arm), it is ill suited to capture position differences of the comparatively small objects on the table or provide a good signal when a trajectory is required which is not a straight line. Therefore, we resort to a 'ground truth bottleneck' scheme [68] for a fairer comparison. Instead of providing just a single target image from the end of an expert demonstration, we give the model ten *intermediate target frames* taken every ten steps during the expert demonstration. This breaks down the long-horizon planning problem into multiple short-horizon ones with approximately straight-line trajectories between any two intermediate targets. This gives an upper-bound estimate of VFS's performance, if it had access to a perfect keyframe predictor splitting the long-horizon problem. An example execution of VFS being guided along intermediate target frames is presented in fig. 5.3.

Execution Time. To account for VFS's sampling-based nature and the guided control process using intermediate target images, we give VFS some additional time to execute a task during test time. We set the total test episode length to 400 time steps as opposed to 200 used during the evaluation of GEECO. VFS is given 40 time steps to 'complete' each sub-goal presented via the ten intermediate target images. However, the intermediate target image is updated to the next sub-goal strictly every 40 time steps, irrespective of how 'close' the controller has come to achieving the previous sub-goal. Running one simulated trial with an episode length of 16 seconds takes about ten minutes using a single NVIDIA GTX 1080 Ti. This

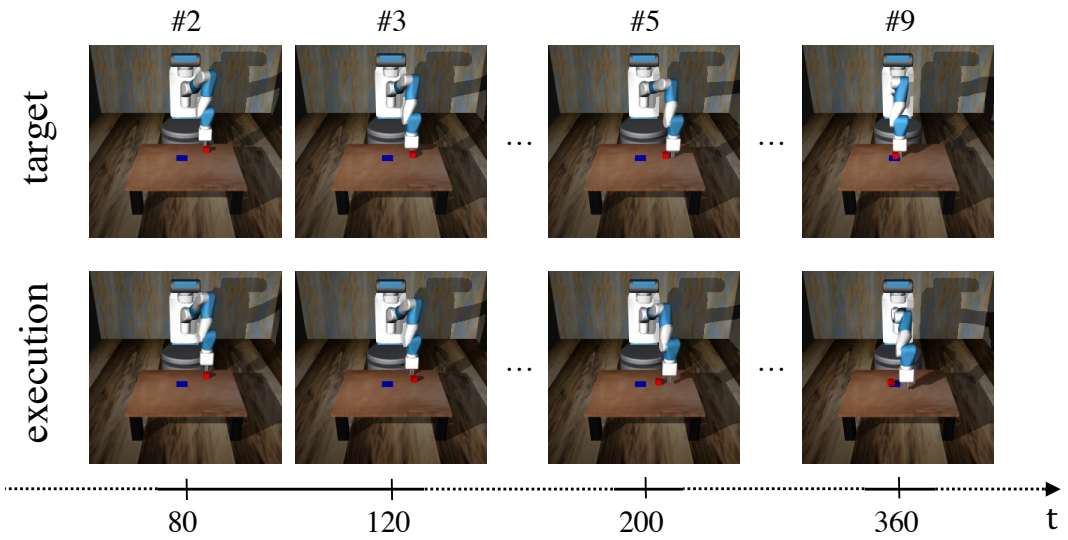


Figure 5.3: An execution of VFS with the 'ground truth bottleneck' scheme. The top row depicts intermediate target images from an expert demonstration. The bottom row shows the corresponding state of execution via VFS at time step t.

timing includes the computational overhead for running and rendering the physics simulation. While this results in an effective control frequency of 0.7 Hz, a like-for-like comparison between VFS and GEECO can not be made in that regard because we have not tuned VFS for runtime efficiency in our scenarios. Potential speedups can be gained from lowering the image resolution and frame rate of the video predictor, predicting shorter time horizons and pipelining the re-planning procedure in a separate thread. However, the fundamental computational bottlenecks of visual MPC can not be overcome with hyper-paramter tuning: Action-conditioned video prediction remains an expensive operation for dynamics forecasting although pixel-level prediction accuracy is presumably not needed to control a robot. Additionally, the action sampling process is a separate part of the model which requires tuning and trades off accuracy versus execution time. In contrast to that, GEECO provides a compelling alternative by reducing the action computation to a single forward pass through the controller network.

5.E TecNet Baseline

We use the official implementation of TecNet for all our experiments[2]. In table 5.5 we provide a comprehensive list of all hyper-parameters used in our experiments with TecNet.

Parameter	Value	Description
iterations	300000	number of gradient updates
batch_size	64	batch size
lr	5e-4	start learning rate of Adam optimiser
img_shape	(125, 125, 3)	image resolution
support	1	k-shot support of new task
query	5	query examples per task during training
embedding	20	size of task embedding vector
activation	elu	layer activation function
filters	16,16,16,16	number of filters in each conv-layer of the embedding network
kernels	5,5,5,5	filter size in each conv-layer of the embedding network
strides	2,2,2,2	stride size in each conv-layer of the embedding network
fc_layers	200,200,200	neurons of the fc-layers of the control network
lambda_embedding	1.0	weight of embedding loss
lambda_support	0.1	weight of support loss
lambda_query	0.1	weight of query loss
margin	0.1	margin of the hinge rank loss
norm	layer	using Layer-Norm throughout the network

Table 5.5: Hyper-parameter setup of TecNet. Hyper-parameters not listed here are kept at their respective default values.

Training Details. We train TecNet in the one-shot imitation setup and provide one 'demonstration' before the start of the controller execution consisting of only the first observation and the target image. During training and evaluation, we resize all images fed into TecNet to 125×125 pixels as per the original paper. We train TecNet with a batch size of 64 for 300k gradient update steps using the Adam optimiser [47] with a start learning rate of 5e-4. One training run takes about 72 hours to complete using a single NVIDIA GTX 1080 Ti with 11 GB of memory.

Execution Time. Running one simulated trial with TecNet with an episode length of eight seconds takes about eight seconds using a single NVIDIA GTX 1080

[2] https://github.com/stepjam/TecNets

Ti. This timing includes the computational overhead for running and rendering the physics simulation resulting in a lower-bound estimate of TecNet's control frequency at 25 Hz. This makes TecNet also a viable option for real-time visuomotor control without any system modifications.

Statement of Authorship for joint/multi-authored papers for PGR thesis

To appear at the end of each thesis chapter submitted as an article/paper

The statement shall describe the candidate's and co-authors' independent research contributions in the thesis publications. For each publication there should exist a complete statement that is to be filled out and signed by the candidate and supervisor **(only required where there isn't already a statement of contribution within the paper itself).**

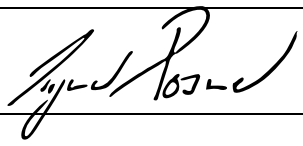
Title of Paper	Goal-Conditioned End-to-End Visuomotor Control for Versatile Skill Primitives
Publication Status	x Published □ Accepted for Publication □ Submitted for Publication □ Unpublished and unsubmitted work written in a manuscript style
Publication Details	Oliver Groth, Chia-Man Hung, Andrea Vedaldi, Ingmar Posner. "Goal-Conditioned End-to-End Visuomotor Control for Versatile Skill Primitives". In: *IEEE International Conference on Robotics and Automation (ICRA)*. June 2021.

Student Confirmation

Student Name:	Chia-Man Hung		
Contribution to the Paper	- implemented the backbone Visuomotor Control model and reproduced experiment results in simulation from the original paper - integrated Visual Foresight (VFS) baseline - designed and implemented the pushing environment - collected part of expert demonstrations for model training - trained models and performed hyperparameter search - designed and drew network architecture figures - contributed to the writing of the paper, specifically: Related Work, Experiments		
Signature		Date	29/09/2022

Supervisor Confirmation

By signing the Statement of Authorship, you are certifying that the candidate made a substantial contribution to the publication, and that the description described above is accurate.

Supervisor name and title: Prof. Ingmar Posner		
Supervisor comments *I confirm that the above is an accurate reflection of the candidate's contributions.*		
Signature 	Date	4/10/2022

This completed form should be included in the thesis, at the end of the relevant chapter.

6

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

In this chapter, we propose a novel approach for motion planning in manipulation. Dissimilar to visuomotor control work in which we collect expert demonstrations of entire tasks for training, only randomly sampled robot poses are needed, rendering the training data collection process easy. We train a latent-variable generative model of robot poses and plan paths through the structured latent space. We demonstrate that such a model achieves commensurate performance against established optimisation-based and sampling-based planners while having the advantage of smoother paths and shorter planning time in certain scenarios. This work was presented at IEEE International Conference on Robotics and Automation (ICRA) in May 2022 and is published as:

Chia-Man Hung, Shaohong Zhong, Walter Goodwin, Oiwi Parker Jones, Martin Engelcke, Ioannis Havoutis, Ingmar Posner. "Reaching Through Latent Space: From Joint Statistics to Path Planning in Manipulation". In: *IEEE Robotics and Automation Letters (RA-L)*. Feb. 2022.

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]

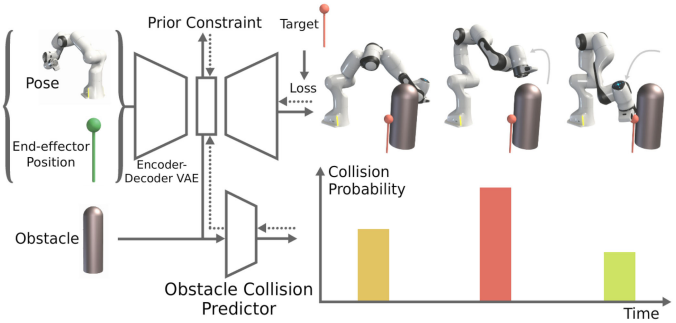


Fig. 1. A VAE is trained to produce a latent representation $\mathbf{z}$ of the joint states and corresponding end-effector positions and an obstacle collision predictor learns the probability of collision. Once trained, gradients through the VAE decoder and collision predictor enable optimisation in 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.

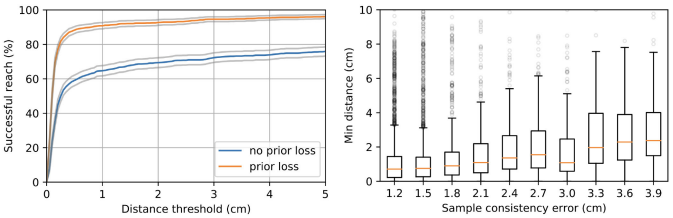


Fig. 2. Left: Target reaching success vs reaching distance threshold, evaluated on 1,000 scenarios. Grey lines are the 95% confidence interval of Wilson score [36]. Adding prior loss in AM objective function improves reaching success rate. Right: Minimum distance between the end-effector and the target 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 $\mathcal{L}^{AM}_t = \|\hat{\boldsymbol{e}}, \boldsymbol{e}_{\text{target}}\|_2 + \lambda^{t+1}_{\text{prior}}(-\log p(\mathbf{z})) + \lambda^{t+1}_{\text{obs}} \sum_i(-\log(1 - p_\vartheta(\mathbf{z}, \mathbf{o}_i)))$;
  update $\mathbf{z}_{t+1} = \mathbf{z}_t - \alpha_{\text{AM}} \nabla \mathcal{L}^{\text{AM}}_t$;
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

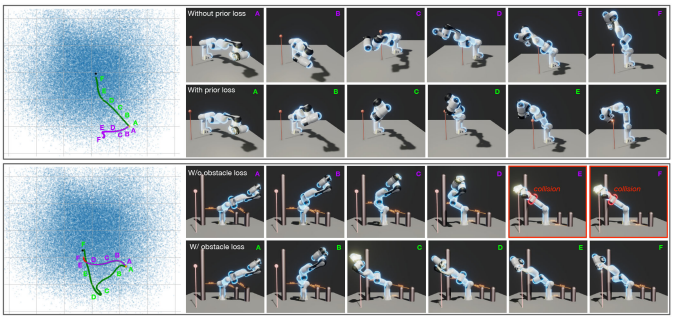


Fig. 3. We project the latent space down to 2D via PCA to visualise AM with and without prior loss (top)/ with and without obstacle loss (bottom). The blue region is the encoding of the training distribution. The green and the purple curves are the robot trajectories from AM. The black dot is the latent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

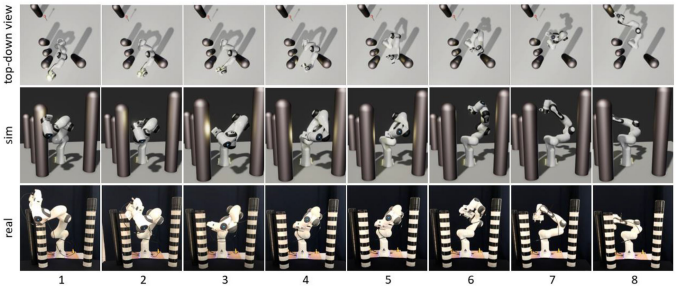


Fig. 4. Real-world experiments: a rollout of a trajectory using LSPP. Our latent-space approach operates in state space and therefore trivially transfers to the real world.

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

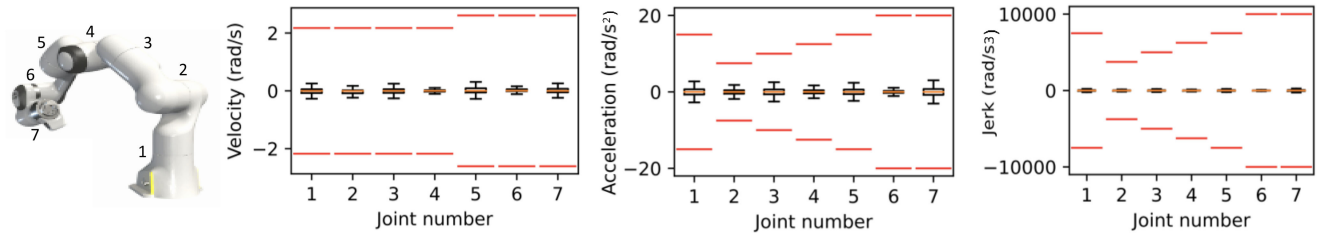


Fig. 5. Dynamic feasibility of motion plans for Panda arm over 1,000 trajectories. No LSPP motion plans violate the joint limits, indicated by the red segments. Left: angular velocity. Middle: angular acceleration. Right: angular jerk.

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.

Appendix

6.A Training Details

6.A.1 LSPP Hyperparameters

A grid search is run on the following hyperparameter values. The final values are chosen by sample consistency.

Parameter	Value
`Number of hidden layers`	2, **4**, 8
`Units per layer`	64, 128, 256, 1024, **2048**
`Latent dimension`	**7**, 10, 20, 32, 64
`GECO reconstruction target` τ	0.0001, 0.0002, 0.0004, 0.0006, **0.0008** 0.001, 0.0012
`Learning rate (GECO)`	0.001, 0.002, 0.003, 0.004, **0.005**, 0.006, 0.007 0.008, 0.009, 0.01, 0.02, 0.05, 0.1
`Learning rate (VAE)`	**0.0001**, 0.0002, 0.0003, 0.0005, 0.001, 0.01

Table 6.1: Training hyperparameters for grid search. Bold font indicates the values chosen.

6.A.2 Choice of Hyperparameters

We discuss the effects of some of the hyperparameters on model training and performance.

Number of hidden layers and units per layer The number of fully connected hidden layers and the number of units per layer in the neural network affect its capacity, which is important for the VAE in modelling the kinematics relationship and for the obstacle classifier in predicting collision. However, having a network that is too large (e.g. eight hidden layers) is found to lead to instabilities in training

and to having diminishing returns in terms of performance. Thus, a grid search is conducted on the size of the neural network. For memory efficiency, we choose to perform the grid search on the number of hidden units in powers of two starting from two hidden layers and 64 units per layer.

Latent dimension The number of latent dimensions determines the capacity of the latent space to capture the correlation between the joint angles and the end-effector position. As the expressive power of the decoder is finite, having a small number of latent dimensions is found to create an information bottleneck that prevents the VAE from generating accurate reconstructions. The information preference problem [104] may also be created if we employ a large number of latent dimensions, which means that many of the latent dimensions may not capture any useful information, which in turn encourages the decoder to ignore the latent encoding. This is also not desirable given our motion planning pipeline is based on latent traversal. Thus, we perform a grid search for the number of latent dimensions, and find that a dimension of seven (i.e. the same as the number of DoFs of the robot) achieves the best performance in terms of our metrics.

GECO reconstruction target The GECO reconstruction target τ imposed as a constraint via a Lagrange multiplier mechanism in GECO [78] is found to be important for the performance of the VAE model. If the goals are strict (i.e. perfect reconstruction), as the size of the neural network is limited and thus limiting its inference and generation capacity, the MSE term in ELBO overwhelms the KL regulariser due to the Lagrange multiplier, leading to overfitting and a mismatch between the posterior and the prior. On the other hand, if the goals are loose, the reconstruction accuracy becomes poor, leading to worse sample consistency. Thus, we use a grid search on the parameter τ.

Learning rate (GECO) The learning rate for the GECO Lagrange multiplier determines the responsiveness of the λ parameter to the violation of the GECO reconstruction target. The higher the learning rate, the more responsive the λ

parameter becomes in adjusting the relative weights of the two terms in ELBO in training the VAE. However, a high GECO learning rate leads to instabilities in training. The optimal value is then found through a grid search.

6.B Planning Details

6.B.1 Modification of GECO

The following algorithm is applied to update the individual λ parameters using a modification of the GECO algorithm [78]. The algorithm is applied to different pairs of loss terms $(-\log p(\mathbf{z}), \|\hat{\boldsymbol{e}}, \boldsymbol{e}_{\text{target}}\|_2)$ and $(\sum_i(-\log(1-p_\vartheta(\mathbf{z}, \mathbf{o}_i))), \|\hat{\boldsymbol{e}}, \boldsymbol{e}_{\text{target}}\|_2)$ to compute λ_{prior} and λ_{obs} at each update step.

Algorithm 1: Update GECO λ

read current λ^t;
read loss terms (l_1^t, l_2^t);
compute constraint violation $C^t = l_1^t - \tau_{goal}$;
if t=0 **then**
    initialise moving average $C_{ma}^0 = C^0$;
else
    $C_{ma}^t = \alpha_{ma} C_{ma}^{t-1} + (1-\alpha_{ma}) C^t$;
end if
compute update step $\kappa^t = \exp(\alpha_{GECO} C_{ma}^t)$;
update $\lambda^{t+1} = \kappa^t \lambda^t$

6.B.2 Planning Hyperparameters

A grid search on the planning hyperparameters is run on a validation dataset of obstacle scenarios. The values chosen are given in Table 6.2.

6.C Choice of Baselines

MoveIt [9] is the most widely used framework for robot manipulation. The Open Motion Planning Library (OMPL) [88] is a collection of sampling-based motion planning algorithms and is the default planner in MoveIt. In our experiments, seven MoveIt path planning algorithms are chosen for comparison. In the following, we provide a summary (mostly condensed from the OMPL documentation) and the rationale behind our choice.

Parameter	Value
Learning rate (AM) α_{AM}	0.03
Learning rate (GECO) α_{GECO}	0.01
Max number of planning steps T	300
Reaching distance threshold γ	0.01
GECO prior loss target τ_{goal}^{prior}	0.4, 0.6, 0.7, 0.8, **0.9**, 1, 1.2, 1.5, 2
Moving average factor α_{ma}^{prior} for prior loss	0.8, 0.9, **0.95**
GECO obstacle loss target τ_{goal}^{obs}	0.5, 0.6, **0.7**, 0.8, 0.9, 1.0, 1.5
Moving average factor α_{ma}^{obs} for obstacle loss	0.1, 0.2, 0.3, **0.4**, 0.6, 0.8, 0.9, 0.95

Table 6.2: Planning hyperparameters. Some hyperparameters are fixed. Bold font indicates the values chosen.

RRTConnect [54] is one of the default planners in MoveIt. It grows two RRTs [58], one from the start and one from the goal, and attempts to connect them. It is an improved version of RRT and is probabilistic complete, ensuring a solution will be found if one exists, given sufficient runtime. It is commonly used and best known for its fast convergence, even in high-dimensional spaces.

LBKPIECE [87] is the other default planner in MoveIt. KPIECE, a sampling-based path planning algorithm designed specifically for planning in high-dimensional spaces, uses a discretisation to guide the exploration of the continuous space. It offers computational advantages by employing projections from the searched space to lower-dimensional Euclidean spaces for estimating exploration coverage. LBKPIECE is a bi-directional variant of KPIECE with lazy collision checking and one level of discretisation. It is also commonly used and known for its planning efficiency.

RRT* [44] is an asymptotically optimal incremental sampling-based path planning algorithm. It is an optimal variant of RRT and converges to an optimal solution in terms of path length after infinite time. This baseline is insightful in comparing path length.

LazyPRM* [7] is another asymptotically optimal sampling-based planner. The Probabilistic Roadmap Method (PRM) constructs a roadmap and checks whether a path exists in the roadmap between a start and goal state. PRM* gradually

increases the number of connection attempts as the roadmap grows in a way that provides convergence to the optimal path. LazyPRM* is a variant of PRM* with lazy state validity checking. This is another useful baseline for path length.

FMT* [43] stands for Fast Marching Tree. It is another asymptotically optimal sampling-based planner. The algorithm is specifically aimed at solving complex motion planning problems in high-dimensional configuration spaces, by performing a lazy dynamic programming recursion on a set of probabilistically-drawn samples to grow a tree of paths.

BIT* [25] stands for Batch Informed Trees. It is an anytime asymptotically optimal sampling-based planner that uses heuristics to prioritise expansion towards the goal and high-quality paths. It has been shown to outperform existing sampling-based planning algorithms, e.g. RRT* and FMT*, in terms of computational cost to find equivalent results.

CHOMP [73] stands for covariant Hamiltonian optimisation for motion planning. It is a gradient-based trajectory optimisation procedure, and uses two objective functions: an obstacle term that captures obstacle avoidance and a smoothness term that captures the dynamics of the trajectory. It is able to avoid obstacles in most cases, but it can fail if it gets stuck in a local minimum due to a bad initial guess for the trajectory. OMPL can be used to generate collision-free seed trajectories for CHOMP to mitigate this issue. Thus, in our experiments, we use OMPL with the default RRTConnect planner for an initial guess and use CHOMP as a post-processor. CHOMP is efficient, produces smooth paths and is the most commonly used optimisation-based approach.

6.D Analysis on Latent Space Representation

The latent space of our VAE encodes pairs of joint position $\boldsymbol{q}$ and end-effector position $\boldsymbol{e}$. How accurate is the $\boldsymbol{q}$ -to-$\boldsymbol{e}$ mapping in the latent space? Figure 6.1

(left) presents a histogram of sample consistency errors from 10,000 prior samples. The peak is centred at around 2.5mm and over 95% of the samples fall below 1cm. What is the completeness of the latent space, i.e. does the latent space cover all the valid joint space or does it miss parts of it? To answer this question, 10,000 samples are drawn from the prior and decoded to $\{\hat{\boldsymbol{q}}, \hat{\boldsymbol{e}}\}$. In fig. 6.1 (right), we plot the sample consistency errors at different end-effector positions $\hat{\boldsymbol{e}}$. We observe that all the valid joint space is well covered and there is no obvious correlation between the sample consistency error and the end-effector position.

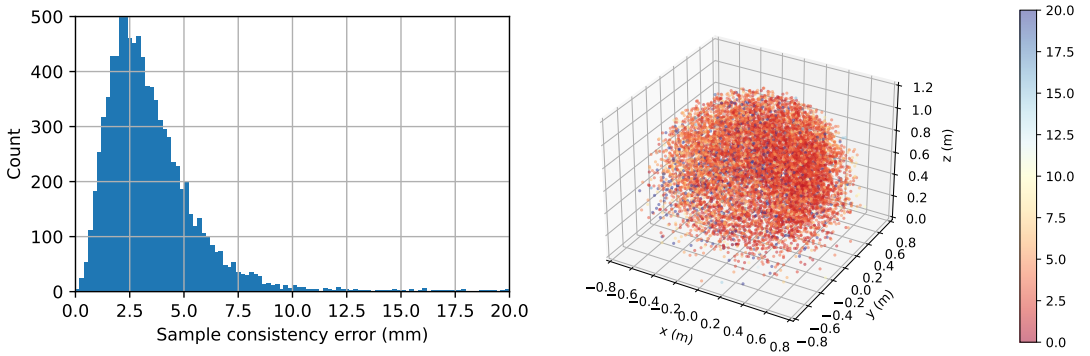


Figure 6.1: Analysis on latent space representation. Left: histogram of sample consistency errors from 10,000 prior samples. Right: 3d scatter of sample consistency errors at different end-effector positions.

7

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

In this chapter, we extend previous work from chapter 6 and probe into the potential of incorporating scene observation from sensory data into our generative model for motion planning. This empowers our model to generalise to unseen scenes in the real world. We illustrate its ability to handle both open and closed-loop planning, especially useful in adapting to dynamic constraints. Moreover, a constraint on the orientation of the end-effector is integrated to our model and demonstrated in reaching pre-grasp poses. Overall, our approach achieves better planning time and commensurate success rate to established sampling-based planners in both simulated environment and real world. This work has been submitted to IEEE International Conference on Robotics and Automation (ICRA) 2023 as:

Jun Yamada*, Chia-Man Hung*, Jack Collins, Ioannis Havoutis, Ingmar Posner. "Leveraging Scene Embeddings for Gradient-Based Motion Planning in Latent Space". Under review at: *IEEE International Conference on Robotics and Automation (ICRA)*. June 2023. * Equal contribution.

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

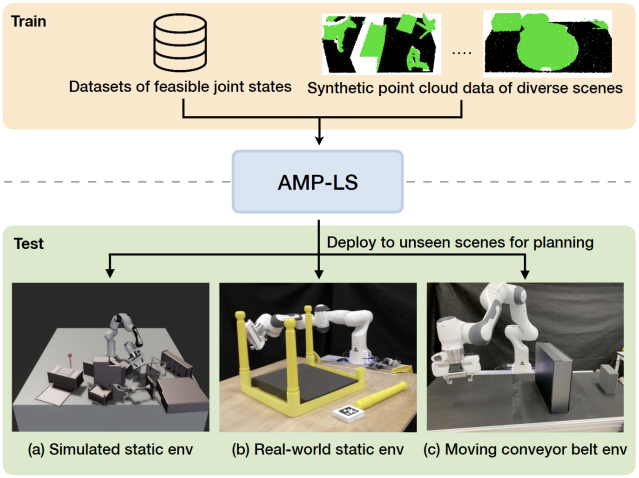


Fig. 1: **Problem setup.** AMP-LS generates a collision-free trajectory via gradient-based optimisation by leveraging scene embeddings. Our model is trained on kinematically feasible robot joint states and synthetic point cloud of diverse scenes. For evaluation, our method is deployed to unseen scenes 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 named 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 $\{\mathbf{q}_0, \ldots, \mathbf{q}_T\}$ 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) $\mathcal{L}^{\mathrm{ELBO}} \leq p(\mathbf{x})$:

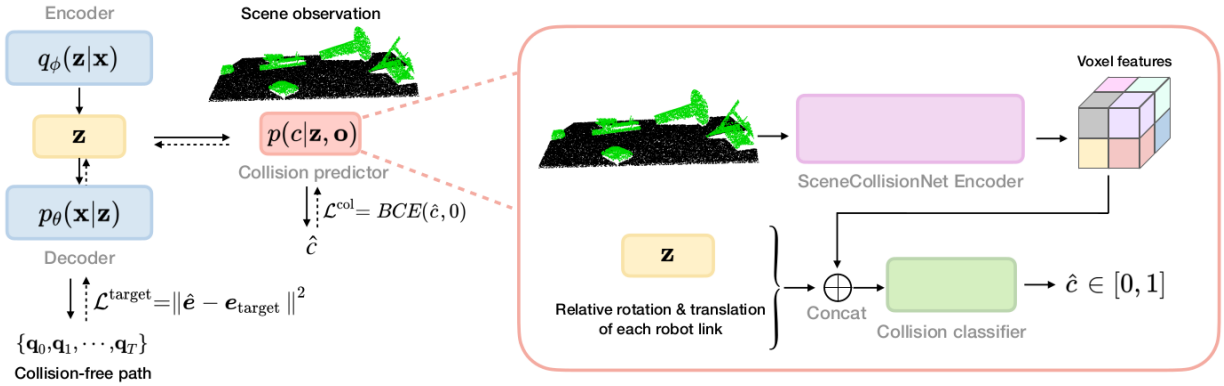


Fig. 2: **Our method overview.** A VAE (**blue**) is trained using feasible robot states $\mathbf{x}$ consisting of joint states, end-effector position, and end-effector orientation to learn structured latent representations $\mathbf{z}$ (**yellow**). Then, freezing the weights of the pre-trained encoder 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 $d(\hat{\boldsymbol{e}}_t, \boldsymbol{e}_{\text{target}}) < \gamma$ **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:

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

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

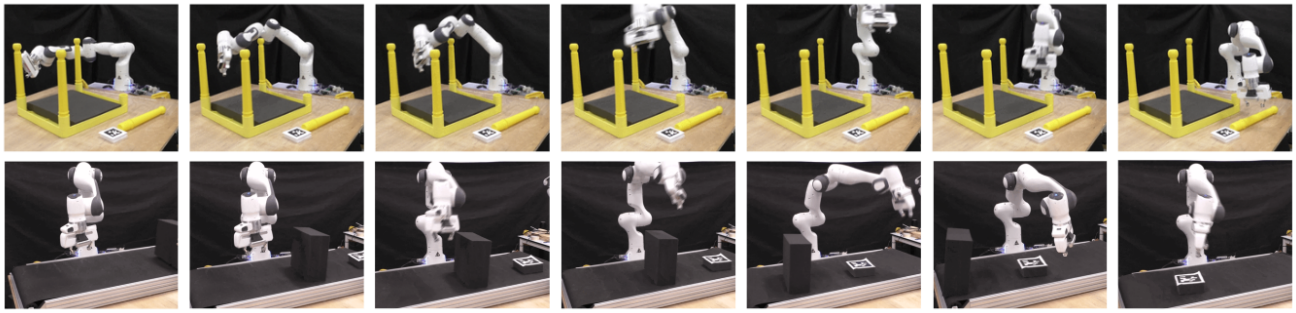


Fig. 3: Visualisation of real-world experiments. **Top**: Our method successfully plans a collision-free trajectory in a complex real-world scene from an impeded start configuration to a pre-grasp goal configuration. By training a collision predictor on diverse synthetic scenes, our method can readily 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)	**0.75 ± 0.08**	**0.26 ± 0.13**	5.25 ± 2.33
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].

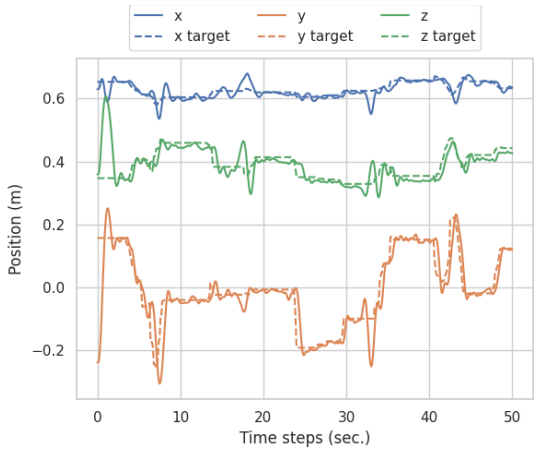


Fig. 4: **Coordinates of end-effector and moving targets in closed-loop settings.** To verify the ability of closed-loop planning in our method, we deploy our method to the real-world robot arm to reach a moving target.

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.

Statement of Authorship for joint/multi-authored papers for PGR thesis

To appear at the end of each thesis chapter submitted as an article/paper

The statement shall describe the candidate's and co-authors' independent research contributions in the thesis publications. For each publication there should exist a complete statement that is to be filled out and signed by the candidate and supervisor **(only required where there isn't already a statement of contribution within the paper itself).**

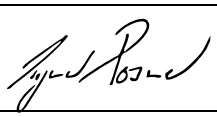
Title of Paper	Leveraging Scene Embeddings for Gradient-Based Motion Planning in Latent Space
Publication Status	□ Published □ Accepted for Publication x Submitted for Publication □ Unpublished and unsubmitted work written in a manuscript style
Publication Details	Jun Yamada*, Chia-Man Hung*, Jack Collins, Ioannis Havoutis, Ingmar Posner. "Leveraging Scene Embeddings for Gradient-Based Motion Planning in Latent Space". Under review at: *IEEE International Conference on Robotics and Automation (ICRA)*. June 2023. *Equal contribution.

Student Confirmation

Student Name:	Chia-Man Hung		
Contribution to the Paper	- designed and set up a simulated environment in Gazebo and ROS - integrated MoveIt motion planning baselines - investigated different orientation representations for moving the robot end-effector to a certain orientation - trained models and performed hyperparameter search - collected data of task scenarios - built Unity demo for visualisation, collision check, and evaluation metrics - evaluated our trained models and baselines in simulation (both static and dynamic) - proposed new ideas: adding a safety margin for the colliders of the robot arm, changing the loss function to enforce the collision constraint as a hard constraint, interpolating feasible states and adding intermediate steps by optimising collision loss - contributed to the writing of the paper: part of every section		
Signature		Date	29/09/2022

Supervisor Confirmation

By signing the Statement of Authorship, you are certifying that the candidate made a substantial contribution to the publication, and that the description described above is accurate.

Supervisor name and title: Prof. Ingmar Posner			
Supervisor comments *I confirm that the above is an accurate reflection of the candidate's contributions.*			
Signature		Date	04/10/2022

8 Discussion

Solving robot manipulation tasks requires putting together several puzzle pieces as pointed out in the background in chapter 2. Specifically, among the learning challenges in robot manipulation, this thesis focuses on improving different aspects of learning a skill policy for manipulation and a fundamental problem – motion planning. In the previous chapters of this thesis, we have introduced visuomotor control and latent space planning and investigated their conduciveness in various manipulation scenarios. In section 8.1, we summarise the key contributions of the thesis and revisit the guiding questions in chapter 1. In section 8.2, we reflect on the limitations of the approaches that we proposed. Finally, in section 8.3, we comment on future research directions to address the identified shortcomings.

8.1 Key Contributions

In chapter 1, we have discussed the main strengths and limitations of visuomotor control. To address the limitation of distribution shift rooted in the nature of standard behavioural cloning, in chapter 4, we suggest a mechanism to detect potential failures due to the compounding errors from distribution shift and recover from failure before the end of a policy rollout. In particular, we propose Introspective VMC, a method extending E2EVMC [42] by monitoring policy uncertainty to recover

from potential failures. In essence, it turns a VMC module into a Bayesian VMC module that comes with an epistemic uncertainty. We verify that this policy uncertainty is inversely correlated with task success rate and can thus be used to detect potential task failures. In addition, we propose a recovery strategy based on following the action with the minimum uncertainty. Empirically, we show that our proposed recovery strategy outperforms the original VMC without recovery and several baseline recovery strategies in pushing, pick-and-place, and pick-and-reach manipulation tasks. Although our approach is built on top of E2EVMC [42], this framework is potentially also applicable to other deterministic policies for failure recovery.

Another limitation of visuomotor control is the lack of versatility in task definition. If we are given a skill policy of putting a red cube into a blue basket, ideally we would like to reuse this skill to put a yellow cube into a green basket rather than training a new skill policy from scratch. In chapter 5, we introduce GEECO, a goal-conditioned visuomotor control policy trained in an end-to-end manner. By leveraging dynamic images representation to encode the motion of the robot arm between the current frame and the target frame, we are able to focus on the relevant parts, such as location and object geometries, that are involved in the task. A new task can then be specified as a single target image and can be solved without any further fine-tuning at test time. We compare the performance of our proposed model against two representative baselines of visual MPC and one-shot imitation learning and show its usefulness in goal-conditioned pushing and pick-and-place tasks. Moreover, we demonstrate GEECO's robustness by successfully learning versatile skills from visual demonstrations and generalising to challenging, unseen scenes with visual distortions or novel object geometries.

Visuomotor control requires a significant amount of demonstration trajectories that may be difficult to collect. E2EVMC hand-designed an expert policy for pick-and-place in simulation and applied domain randomisation techniques to transfer from sim to real. For more sophisticated manipulation tasks, it may be hard to design an expert policy in simulation or time-consuming to manually

collect demonstrations in the real world. In chapter 6, we propose latent space planning for robot manipulation of which the training data simply comes from random motor-babbling on a real platform or sampling kinematically feasible robot states in simulation. For applications, we focus on motion planning, which is a fundamental skill in manipulation. Our proposed method LSPP learns a structured latent space of a generative model that captures a model of forward and inverse kinematics. Constraints for reaching and obstacle avoidance all operate on this latent space, rendering motion planning as an optimisation process in the same space. In comparison against several traditional sampling and optimisation-based motion planning baselines, LSPP bypasses the traditional computational challenges while achieving commensurate performance in terms of reaching success, planning time and path length.

One of the main drawbacks of LSPP is that it requires complete knowledge of the scene as a state-based representation consisting of position and size of primitive shapes, making it hard to be applied to complex or dynamic scenes. In chapter 7, we propose AMP-LS, a significant extension to LSPP, by introducing a collision predictor leveraging scene embeddings, orientation constraints, and explicit collision checking and automatic rescaling for obstacle avoidance. In contrast to LSPP, the collision predictor of AMP-LS incorporates environmental constraints by encoding point cloud observations from a third-person mounted camera pointing towards the scene. We adapt SceneCollisionNet [11] for encoding scene observations, resulting in a collision predictor that can reason about the occluded region in the scene. Adding explicit collision checking during the planning phase intuitively reduces collision. Empirically, we demonstrate that AMP-LS handles both open and closed-loop planning in challenging environments, such as complex cluttered static scene and dynamic scene with a moving obstacle and moving target.

8.2 Limitations

In the previous section, we have recapitulated the key contributions made in this thesis and explained how each of them addressed the guiding questions posed in

chapter 1. We believe the empirical evidence from chapters 4 to 7 supports the idea of learning visuomotor control policies and planning in latent space as an effective means of solving robot manipulation tasks. However, our work would not be complete without putting it into perspective with its limitations to specify to which extent it is applicable and to identify future research directions to remedy its shortcomings (cf. section 8.3).

First, a visuomotor control policy is not task-agnostic and displays limited ability of generalisation beyond its training domains. To alleviate this, in chapter 5, GEECO leverages dynamic image representation to include within-task and across-task variations, such as novel colours or object geometries. However, it is still impossible to exhibit further generalisation if it is not explicitly mitigated by the model design. For example, if a skill policy is solely trained on pick-and-place demonstrations, it is unlikely for it to accomplish a pushing task. In fact, the limitation of generalisability is inherent to all supervised learning approaches and this goes back to the discussion of policy structures in chapter 2 regarding model representational power and degree of generalisation.

Second, learning a successful visuomotor control policy is highly dependant on the quality and quantity of demonstration trajectories. Collecting the required amount of training data on a real physical platform is often time-consuming or even infeasible. While in chapters 4 and 5, the demonstrations are easily obtainable in procedurally generated simulations through a hand-designed expert policy, the domain transfer from sim to real still requires significant additional effort. It should be noted that no existing physics simulator can fully capture the physics dynamics in the real world and unlike in simulation, real sensor observations inevitably contain noise. While one or few-shot imitation learning [13] holds promise in learning from very few demonstrations of any given task and generalising to new situations of the same task after little fine-tuning, the training phase of the generic neural network to be fine-tuned still requires the same or even more effort. Although E2EVMC [42] successfully employs domain randomisation techniques to facilitate the domain transfer, it can still be a daunting process to replicate the exact real-world scene

setup in simulation. For these reasons, improving sample efficiency or developing simple efficient domain adaptation techniques remains crucial when deploying a visuomotor control policy onto a real physical platform.

Next, latent space planning is by design an optimisation-based local algorithm, and is therefore subject to local minima. During planning, multiple loss terms from different constraints are being optimised simultaneously. Unlike a standard optimisation problem, e.g., training a neural network, in which we only care about the final outcome, when planning in the latent space, every step needs to decode to a kinematically feasible collision-free joint state for a trajectory to be considered a success. In chapter 6, it has been observed that sometimes the optimisation process reaches a local minimum, translating to the robot not reaching the target. It is also common to see that when close to the boundary of an obstacle, the collision predictor still predicts low collision probability as it is supposed to, resulting in the optimisation process not prioritising the obstacle loss term with GECO automatically adjusting the coefficients, causing the robot to collide with an obstacle. To mitigate this, in chapter 7, we propose explicit collision checking during planning to reduce the chance of collision using the collision predictor. However, the rescaling of the coefficients of the loss terms after a collision is predicted sometimes yields detours in the generated motion plans, which could potentially be further improved.

Finally, robot manipulation tasks are inherently underactuated systems. Even if the robot is fully actuated, in order to move the inanimate objects in the environment, the robot needs to first move to a state in which it can interact with the objects and change the state of the objects, e.g., the robot can first open its gripper and move into a pre-grasp pose, and then close to gripper to pick up an object. This characteristic of underactuation has some undesirable effects. For instance, in chapter 4, this makes reverting back to the most certain state in the recovery strategy not always possible. However, in our experiments, even if the back trace of the state is not fully completed, changing to a different state rather than staying in an uncertain state can still be beneficial to accomplishing a task. On the ground of underactuation, in the design of latent space planning for manipulation in chapters

6 and 7, we could not include the state of the objects in the input and output of the generative model to learn the robot state and the state of the objects jointly. Such adjacent latent representations would not always decode to feasible adjacent states. For example, one AM step to reduce the distance between the end-effector of the robot and the target object might result in both of them moving towards each other, while the inanimate object cannot move on its own. This phenomenon has been observed when designing and carrying out early experiments.

8.3 Future Work

Reflecting upon the current limitations investigated in the previous section, several lines of future research directions can be established. Regarding the issues related to limited generalisation and sample efficiency, one possible avenue of future work would be to innovate the model architecture to induce enough representational power to express different manipulation skills while encoding the underlying task structure as an architectural prior. CNNs and MLPs as currently used in the architecture of visuomotor control may be too general. *Object-centric representation* [16, 17, 94], based on the structural assumption that the world is made up of objects and for manipulation tasks, the objective is to modify some attribute of a set of objects, may facilitate generalisation and improve sample efficiency by considering objects as equivalent through the use of an abstract representation. *Neuro-symbolic reasoning* [26, 82] marry two powerful ideas: neural networks and high-level symbolic reasoning. One fundamental difference between the two is that for symbolic systems, representations are discrete and in theory understandable by a human, while for neural networks, representations are learned during training and often continuous. Recently, such a fundamental difference challenge has been overcome in various contexts [61, 99]. Combining learning and reasoning has been shown more effective than purely symbolic or neural approaches [70, 71]. These are thus prospective avenues of future work that can potentially benefit visuomotor control and manipulation tasks in general.

If improving sample efficiency proves to be too challenging and a significant amount of training data is still required, an alternative approach would be to explore the potential of a simpler data acquisition process. For visuomotor control, collecting demonstrations in simulation instead of on a real robotic platform, has the inevitable undesirable consequence of needing to replicate the real-world setup in simulation and to rely on domain adaptation techniques as stated earlier. To mitigate this, one possible future research direction lies in collecting demonstrations in the real world using crowd-sourcing or an imperfect policy, e.g., using a trained reinforcement learning agent. An existing line of work focuses on improving the performance of imitation learning given a small amount of sub-optimal training data. For instance, to learn from imperfect demonstrations, Wu et al. [96] propose to use confidence scores to describe the quality of demonstrations in combination with GAIL [36].

Due to the underactuated nature of manipulation tasks, current latent space planning for manipulation in chapters 6 and 7 has been designed to solve motion planning, which does not involve interaction with the inanimate objects in the scene. A natural question of extending it beyond motion planning to other manipulation tasks involving interaction arises. One naïve approach would be to consider motion planning as a skill primitive and combine it with other skill primitives, e.g., opening or closing the gripper, to solve sequential tasks. However, this would make it lose the appeal of elegance and simplicity. If the challenge of designing an end-to-end latent space planning algorithm for manipulation could be overcome, this approach could be leveraged as an important task-agnostic framework for learning a general skill policy for robot manipulation.

References

[1] Pulkit Agrawal, Ashvin V Nair, Pieter Abbeel, Jitendra Malik, and Sergey Levine. "Learning to poke by poking: Experiential learning of intuitive physics". In: *Advances in neural information processing systems*. 2016, pp. 5074–5082.

[2] Peter K Allen, Aleksandar Timcenko, Billibon Yoshimi, and Paul Michelman. "Automated tracking and grasping of a moving object with a robotic hand-eye system". In: *IEEE Transactions on Robotics and Automation* 9.2 (1993), pp. 152–165.

[3] Ershad Banijamali, Rui Shu, Hung Bui, and Ali Ghodsi. "Robust locally-linear controllable embedding". In: *International Conference on Artificial Intelligence and Statistics*. PMLR. 2018, pp. 1751–1759.

[4] Dmitry Berenson, Siddhartha S. Srinivasa, Dave Ferguson, and James J. Kuffner. "Manipulation planning on constraint manifolds". In: *2009 IEEE International Conference on Robotics and Automation*. Vol. 5. 4. IEEE, May 2009, pp. 625–632. eprint: 2008.03787. URL: http://ieeexplore.ieee.org/document/5152399/.

[5] Aude Billard and Danica Kragic. "Trends and challenges in robot manipulation". In: *Science* 364.6446 (2019), eaat8414.

[6] Botond Bócsi, Duy Nguyen-Tuong, Lehel Csató, Bernhard Schoelkopf, and Jan Peters. "Learning inverse kinematics with structured prediction". In: *2011 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE. 2011, pp. 698–703.

[7] Robert Bohlin and Lydia 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.

[8] Francois Chaumette, Patrick Rives, and Bernard Espiau. "Positioning of a robot with respect to an object, tracking it and estimating its velocity by visual servoing". In: *ICRA*. 1991, pp. 2248–2253.

[9] Sachin Chitta, Ioan Sucan, and Steve Cousins. "MoveIt!" In: *IEEE Robotics & Automation Magazine* 19.1 (2012), pp. 18–19.

[10] Silvia Cruciani, Balakumar Sundaralingam, Kaiyu Hang, Vikash Kumar, Tucker Hermans, and Danica Kragic. "Benchmarking in-hand manipulation". In: *IEEE Robotics and Automation Letters* 5.2 (2020), pp. 588–595.

[11] Michael Danielczuk, Arsalan Mousavian, Clemens Eppner, and Dieter Fox. "Object rearrangement using learned implicit collision functions". In: *2021 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE. 2021, pp. 6010–6017.

[12] Tom Drummond and Roberto Cipolla. “Visual tracking and control using lie algebras”. In: *Proceedings. 1999 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (Cat. No PR00149)*. Vol. 2. IEEE. 1999, pp. 652–657.

[13] Yan Duan, Marcin Andrychowicz, Bradly Stadie, OpenAI Jonathan Ho, Jonas Schneider, Ilya Sutskever, Pieter Abbeel, and Wojciech Zaremba. “One-shot imitation learning”. In: *Advances in neural information processing systems*. 2017, pp. 1087–1098.

[14] Frederik Ebert, Chelsea Finn, Sudeep Dasari, Annie Xie, Alex Lee, and Sergey Levine. “Visual foresight: Model-based deep reinforcement learning for vision-based robotic control”. In: *arXiv preprint arXiv:1812.00568* (2018).

[15] Frederik Ebert, Chelsea Finn, Alex X Lee, and Sergey Levine. “Self-supervised visual planning with temporal skip connections”. In: *Conference on Robot Learning*. 2017.

[16] Martin Engelcke, Adam R Kosiorek, Oiwi Parker Jones, and Ingmar Posner. “GENESIS: Generative Scene Inference and Sampling with Object-Centric Latent Representations”. In: *International Conference on Learning Representations (ICLR)* (2020).

[17] Martin Engelcke, Oiwi Parker Jones, and Ingmar Posner. “Genesis-v2: Inferring unordered object representations without iterative refinement”. In: *Advances in Neural Information Processing Systems* 34 (2021), pp. 8085–8094.

[18] Dumitru Erhan, Yoshua Bengio, Aaron Courville, and Pascal Vincent. *Visualizing Higher-Layer Features of a Deep Network*. Tech. rep. 1341. Also presented at the ICML 2009 Workshop on Learning Feature Hierarchies, Montréal, Canada. University of Montreal, June 2009.

[19] Chelsea Finn, Pieter Abbeel, and Sergey Levine. “Model-agnostic meta-learning for fast adaptation of deep networks”. In: *International conference on machine learning*. PMLR. 2017, pp. 1126–1135.

[20] Chelsea Finn and Sergey Levine. “Deep visual foresight for planning robot motion”. In: *2017 IEEE International Conference on Robotics and Automation (ICRA)*. May 2017, pp. 2786–2793.

[21] Chelsea Finn, Tianhe Yu, Tianhao Zhang, Pieter Abbeel, and Sergey Levine. “One-Shot Visual Imitation Learning via Meta-Learning”. In: *Conference on Robot Learning*. 2017, pp. 357–368.

[22] Fabrizio Flacco, Torsten Kröger, Alessandro De Luca, and Oussama Khatib. “A depth space approach to human-robot collision avoidance”. In: *2012 IEEE International Conference on Robotics and Automation*. IEEE. 2012, pp. 338–345.

[23] Yarin Gal. “Uncertainty in deep learning”. In: (2016).

[24] Yarin Gal and Zoubin Ghahramani. “Dropout as a bayesian approximation: Representing model uncertainty in deep learning”. In: *international conference on machine learning*. 2016, pp. 1050–1059.

[25] Jonathan D Gammell, Timothy D Barfoot, and Siddhartha S Srinivasa. “Batch Informed Trees (BIT*): Informed asymptotically optimal anytime search”. In: *The International Journal of Robotics Research* 39.5 (2020), pp. 543–567.

[26] Artur d'Avila Garcez, Sebastian Bader, Howard Bowman, Luis C Lamb, Leo de Penning, BV Illuminoo, Hoifung Poon, and COPPE Gerson Zaverucha. "Neural-symbolic learning and reasoning: A survey and interpretation". In: *Neuro-Symbolic Artificial Intelligence: The State of the Art* 342 (2022), p. 1.

[27] Jakob Gawlikowski, Cedrique Rovile Njieutcheu Tassi, Mohsin Ali, Jongseok Lee, Matthias Humt, Jianxiang Feng, Anna Kruspe, Rudolph Triebel, Peter Jung, Ribana Roscher, et al. "A survey of uncertainty in deep neural networks". In: *arXiv preprint arXiv:2107.03342* (2021).

[28] Oliver Groth, Chia-Man Hung, Andrea Vedaldi, and Ingmar Posner. "Goal-conditioned end-to-end visuomotor control for versatile skill primitives". In: *2021 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE. 2021, pp. 1319–1325.

[29] Danijar Hafner, Kuang-Huei Lee, Ian Fischer, and Pieter Abbeel. "Deep Hierarchical Planning from Pixels". In: *arXiv preprint arXiv:2206.04114* (2022).

[30] Danijar Hafner, Timothy Lillicrap, Jimmy Ba, and Mohammad Norouzi. "Dream to Control: Learning Behaviors by Latent Imagination". In: *International Conference on Learning Representations*. 2019.

[31] Danijar Hafner, Timothy Lillicrap, Ian Fischer, Ruben Villegas, David Ha, Honglak Lee, and James Davidson. "Learning latent dynamics for planning from pixels". In: *International Conference on Machine Learning*. PMLR. 2019, pp. 2555–2565.

[32] Gregory D Hager, Wen-Chung Chang, and A Stephen Morse. "Robot hand-eye coordination based on stereo vision". In: *IEEE Control Systems Magazine* 15.1 (1995), pp. 30–39.

[33] Negin Heravi, Ayzaan Wahid, Corey Lynch, Pete Florence, Travis Armstrong, Jonathan Tompson, Pierre Sermanet, Jeannette Bohg, and Debidatta Dwibedi. "Visuomotor Control in Multi-Object Scenes Using Object-Aware Representations". In: *arXiv preprint arXiv:2205.06333* (2022).

[34] John Hill. "Real time control of a robot with a mobile camera". In: *9th Int. Symp. on Industrial Robots, 1979*. 1979, pp. 233–246.

[35] Noriaki Hirose, Fei Xia, Roberto Martín-Martín, Amir Sadeghian, and Silvio Savarese. "Deep visual mpc-policy learning for navigation". In: *IEEE Robotics and Automation Letters* 4.4 (2019), pp. 3184–3191.

[36] Jonathan Ho and Stefano Ermon. "Generative adversarial imitation learning". In: *Advances in neural information processing systems*. 2016, pp. 4565–4573.

[37] Chia-Man Hung, Li Sun, Yizhe Wu, Ioannis Havoutis, and Ingmar Posner. "Introspective visuomotor control: exploiting uncertainty in deep visuomotor control for failure recovery". In: *2021 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE. 2021, pp. 6293–6299.

[38] Chia-Man Hung, Shaohong Zhong, Walter Goodwin, Oiwi Parker Jones, Martin Engelcke, Ioannis Havoutis, and Ingmar Posner. "Reaching Through Latent Space: From Joint Statistics to Path Planning in Manipulation". In: *IEEE Robotics and Automation Letters* 7.2 (2022), pp. 5334–5341.

[39] Brian Ichter and Marco Pavone. "Robot Motion Planning in Learned Latent Spaces". In: *IEEE Robotics and Automation Letters* 4.3 (July 2019), pp. 2407–2414. arXiv: 1807.10366. URL: https://ieeexplore.ieee.org/document/8653875/.

[40] Auke Jan Ijspeert, Jun Nakanishi, Heiko Hoffmann, Peter Pastor, and Stefan Schaal. "Dynamical movement primitives: learning attractor models for motor behaviors". In: *Neural computation* 25.2 (2013), pp. 328–373.

[41] Divye Jain, Andrew Li, Shivam Singhal, Aravind Rajeswaran, Vikash Kumar, and Emanuel Todorov. "Learning Deep Visuomotor Policies for Dexterous Hand Manipulation". In: *2019 International Conference on Robotics and Automation (ICRA)*. IEEE. 2019, pp. 3636–3643.

[42] Stephen James, Andrew J Davison, and Edward Johns. "Transferring End-to-End Visuomotor Control from Simulation to Real World for a Multi-Stage Task". In: *Conference on Robot Learning*. 2017, pp. 334–343.

[43] Lucas Janson, Edward Schmerling, Ashley Clark, and Marco Pavone. "Fast marching tree: A fast marching sampling-based method for optimal motion planning in many dimensions". In: *The International journal of robotics research* 34.7 (2015), pp. 883–921.

[44] Sertac Karaman and Emilio Frazzoli. "Sampling-based algorithms for optimal motion planning". In: *International Journal of Robotics Research* 30.7 (2011), pp. 846–894.

[45] Maximilian Karl, Maximilian Soelch, Justin Bayer, and Patrick Van der Smagt. "Deep variational bayes filters: Unsupervised learning of state space models from raw data". In: *arXiv preprint arXiv:1605.06432* (2016).

[46] Oussama 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.

[47] Diederik P Kingma and Jimmy 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*. Ed. by Yoshua Bengio and Yann LeCun. 2015. URL: http://arxiv.org/abs/1412.6980.

[48] Diederik P Kingma and Max Welling. "Auto-Encoding Variational Bayes". In: *2nd International Conference on Learning Representations, ICLR 2014, Banff, AB, Canada, April 14-16, 2014, Conference Track Proceedings*. Ed. by Yoshua Bengio and Yann LeCun. 2014.

[49] Diederik P Kingma and Max Welling. "An introduction to variational autoencoders". In: *Foundations and Trends® in Machine Learning* 12.4 (2019), pp. 307–392.

[50] Yoram Koren and Johann Borenstein. "Potential field methods and their inherent limitations for mobile robot navigation." In: *ICRA*. Vol. 2. 1991, pp. 1398–1404.

[51] Petar Kormushev, Sylvain Calinon, and Darwin G Caldwell. "Imitation learning of positional and force skills demonstrated via kinesthetic teaching and haptic input". In: *Advanced Robotics* 25.5 (2011), pp. 581–603.

[52] Danica Kragic and Henrik I Christensen. "Survey on visual servoing for manipulation". In: *Computational Vision and Active Perception Laboratory, Fiskartorpsv* 15 (2002), p. 2002.

[53] Oliver Kroemer, Scott Niekum, and George Konidaris. "A review of robot learning for manipulation: Challenges, representations, and algorithms". In: *The Journal of Machine Learning Research* 22.1 (2021), pp. 1395–1476.

[54] James J Kuffner and Steven M LaValle. "RRT-connect: An efficient approach to single-query path planning". In: *Proceedings 2000 ICRA. Millennium Conference. IEEE International Conference on Robotics and Automation. Symposia Proceedings (Cat. No. 00CH37065)*. Vol. 2. IEEE. 2000, pp. 995–1001.

[55] Vikash Kumar and Emanuel Todorov. "MuJoCo HAPTIX: A virtual reality system for hand manipulation". In: *Humanoid Robots (Humanoids), 2015 IEEE-RAS 15th International Conference on*. IEEE. 2015, pp. 657–663.

[56] Balaji Lakshminarayanan, Alexander Pritzel, and Charles Blundell. "Simple and scalable predictive uncertainty estimation using deep ensembles". In: *Advances in neural information processing systems*. 2017, pp. 6402–6413.

[57] Steven M LaValle. *Planning algorithms*. Cambridge university press, 2006.

[58] Steven M Lavalle. *Rapidly-Exploring Random Trees: A New Tool for Path Planning*. Tech. rep. Iowa State University, 1998.

[59] Alex X Lee, Richard Zhang, Frederik Ebert, Pieter Abbeel, Chelsea Finn, and Sergey Levine. "Stochastic Adversarial Video Prediction". In: *arXiv preprint arXiv:1804.01523* (2018).

[60] Sergey Levine, Chelsea Finn, Trevor Darrell, and Pieter Abbeel. "End-to-end training of deep visuomotor policies". In: *The Journal of Machine Learning Research* 17.1 (2016), pp. 1334–1373.

[61] Qing Li, Siyuan Huang, Yining Hong, Yixin Chen, Ying Nian Wu, and Song-Chun Zhu. "Closed loop neural-symbolic learning via integrating neural perception, grammar parsing, and symbolic reasoning". In: *International Conference on Machine Learning*. PMLR. 2020, pp. 5884–5894.

[62] Kevin M Lynch and Frank C Park. *Modern robotics*. Cambridge University Press, 2017.

[63] Ezio Malis, Francois Chaumette, and Sylvie Boudet. "Positioning a coarse-calibrated camera with respect to an unknown object by 2D 1/2 visual servoing". In: *Proceedings. 1998 IEEE International Conference on Robotics and Automation (Cat. No. 98CH36146)*. Vol. 2. IEEE. 1998, pp. 1352–1359.

[64] Alireza Mehrtash, William M Wells, Clare M Tempany, Purang Abolmaesumi, and Tina Kapur. "Confidence calibration and predictive uncertainty estimation for deep medical image segmentation". In: *IEEE transactions on medical imaging* 39.12 (2020), pp. 3868–3878.

[65] Josh Merel, Arun Ahuja, Vu Pham, Saran Tunyasuvunakool, Siqi Liu, Dhruva Tirumala, Nicolas Heess, and Greg Wayne. "Hierarchical Visuomotor Control of Humanoids". In: *International Conference on Learning Representations*. 2018.

[66] Alexander L Mitchell, Martin Engelcke, Oiwi Parker Jones, David Surovik, Siddhant Gangapurwala, Oliwier Melon, Ioannis Havoutis, and Ingmar Posner. "First Steps: Latent-Space Control with Semantic Constraints for Quadruped Locomotion". In: *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)* (2020), pp. 5343–5350.

[67] Edward F Moore. *Sequential machines: Selected papers*. Addison-Wesley Longman Ltd., 1964.

[68] Suraj Nair and Chelsea Finn. "Hierarchical Foresight: Self-Supervised Learning of Long-Horizon Tasks via Visual Subgoal Generation". In: *International Conference on Learning Representations*. 2019.

[69] Yunpeng Pan, Ching-An Cheng, Kamil Saigol, Keuntak Lee, Xinyan Yan, Evangelos Theodorou, and Byron Boots. "Agile Autonomous Driving using End-to-End Deep Imitation Learning". In: *Robotics: science and systems*. 2018.

[70] H Leo H de Penning, Artur S d'Avila Garcez, Luís C Lamb, and John-Jules C Meyer. "A neural-symbolic cognitive agent for online learning and reasoning". In: *Twenty-Second International Joint Conference on Artificial Intelligence*. 2011.

[71] L de Penning, AS d'Avila Garcez, Luís C Lamb, and JJ Meyer. "An integrated neural-symbolic cognitive agent architecture for training and assessment in simulators". In: *Proceedings of the 6th International Workshop on Neural-Symbolic Learning and Reasoning, NeSy10, held at AAAI-2010*. 2010.

[72] Carl Edward Rasmussen. "Gaussian processes in machine learning". In: *Summer School on Machine Learning*. Springer. 2003, pp. 63–71.

[73] Nathan Ratliff, Matt Zucker, J Andrew Bagnell, and Siddhartha Srinivasa. "CHOMP: Gradient optimization techniques for efficient motion planning". In: *2009 IEEE International Conference on Robotics and Automation*. IEEE. 2009, pp. 489–494.

[74] Nathan D Ratliff, Jan Issac, Daniel Kappler, Stan Birchfield, and Dieter Fox. "Riemannian motion policies". In: *arXiv preprint arXiv:1801.02854* (2018).

[75] Siddharth Reddy, Anca D Dragan, and Sergey Levine. "SQIL: Imitation Learning via Reinforcement Learning with Sparse Rewards". In: *International Conference on Learning Representations*. 2019.

[76] Douglas A Reynolds. "Gaussian mixture models." In: *Encyclopedia of biometrics* 741.659-663 (2009).

[77] Danilo Jimenez Rezende, Shakir Mohamed, and Daan Wierstra. "Stochastic Backpropagation and Approximate Inference in Deep Generative Models". In: *International Conference on Machine Learning (ICML)*. PMLR. 2014, pp. 1278–1286.

[78] Danilo Jimenez Rezende and Fabio Viola. "Taming VAEs". In: *arXiv preprint arXiv:1810.00597* (2018).

[79] Stéphane Ross, Geoffrey Gordon, and Drew Bagnell. "A reduction of imitation learning and structured prediction to no-regret online learning". In: *Proceedings of the fourteenth international conference on artificial intelligence and statistics*. JMLR Workshop and Conference Proceedings. 2011, pp. 627–635.

[80] Reuven Y Rubinstein and Dirk P Kroese. *The Cross Entropy Method: A Unified Approach To Combinatorial Optimization, Monte-Carlo Simulation (Information Science and Statistics)*. Berlin, Heidelberg: Springer-Verlag, 2004.

[81] Lars Ruthotto and Eldad Haber. "An introduction to deep generative modeling". In: *GAMM-Mitteilungen* 44.2 (2021), e202100008.

[82] Md Kamruzzaman Sarker, Lu Zhou, Aaron Eberhart, and Pascal Hitzler. "Neuro-symbolic artificial intelligence: Current trends". In: *arXiv preprint arXiv:2105.05330* (2021).

[83] Sooyoon Shin, Trinity B Crapse, J Patrick Mayo, and Marc A Sommer. "Visuomotor Integration". In: *Encyclopedia of Neuroscience* (2009), pp. 4354–4359.

[84] Rahul Shome, Wei N Tang, Changkyu Song, Chaitanya Mitash, Hristiyan Kourtev, Jingjin Yu, Abdeslam Boularias, and Kostas E Bekris. "Towards robust product packing with a minimalistic end-effector". In: *2019 International Conference on Robotics and Automation (ICRA)*. IEEE. 2019, pp. 9007–9013.

[85] Aravind Srinivas, Allan Jabri, Pieter Abbeel, Sergey Levine, and Chelsea Finn. "Universal planning networks: Learning generalizable representations for visuomotor control". In: *International Conference on Machine Learning*. PMLR. 2018, pp. 4732–4741.

[86] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. "Dropout: a simple way to prevent neural networks from overfitting". In: *The journal of machine learning research* 15.1 (2014), pp. 1929–1958.

[87] Ioan A Şucan and Lydia E Kavraki. "Kinodynamic motion planning by interior-exterior cell exploration". In: *Algorithmic Foundation of Robotics VIII*. Springer, 2009, pp. 449–464.

[88] Ioan A Şucan, Mark Moll, and Lydia E Kavraki. "The open motion planning library". In: *IEEE Robotics & Automation Magazine* 19.4 (2012), pp. 72–82.

[89] Faraz Torabi, Garrett Warnell, and Peter Stone. "Behavioral cloning from observation". In: *Proceedings of the 27th International Joint Conference on Artificial Intelligence*. 2018, pp. 4950–4957.

[90] Ulrich Viereck, Andreas Pas, Kate Saenko, and Robert Platt. "Learning a visuomotor controller for real world robotic grasping using simulated depth images". In: *Conference on Robot Learning*. 2017, pp. 291–300.

[91] Jack Wang, Aaron Hertzmann, and David J Fleet. "Gaussian process dynamical models". In: *Advances in neural information processing systems* 18 (2005).

[92] Manuel Watter, Jost Springenberg, Joschka Boedecker, and Martin Riedmiller. "Embed to control: A locally linear latent dynamics model for control from raw images". In: *Advances in neural information processing systems*. 2015, pp. 2746–2754.

[93] William J Wilson, CC Williams Hulls, and Graham S Bell. "Relative end-effector control using cartesian position based visual servoing". In: *IEEE Transactions on Robotics and Automation* 12.5 (1996), pp. 684–696.

[94] Yizhe Wu, Oiwi Parker Jones, Martin Engelcke, and Ingmar Posner. "APEX: Unsupervised, object-centric scene segmentation and tracking for robot manipulation". In: *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE. 2021, pp. 3375–3382.

[95] Yizhe Wu, Sudhanshu Kasewa, Oliver Groth, Sasha Salter, Li Sun, Oiwi Parker Jones, and Ingmar Posner. "Imagine That! Leveraging Emergent Affordances for 3D Tool Synthesis". In: *arXiv preprint arXiv:1909.13561* (2020).

[96] Yueh-Hua Wu, Nontawat Charoenphakdee, Han Bao, Voot Tangkaratt, and Masashi Sugiyama. "Imitation learning from imperfect demonstration". In: *International Conference on Machine Learning*. PMLR. 2019, pp. 6818–6827.

[97] Patrick Wunsch and Gerd Hirzinger. "Real-time visual tracking of 3D objects with dynamic handling of occlusion". In: *Proceedings of International Conference on Robotics and Automation*. Vol. 4. IEEE. 1997, pp. 2868–2873.

[98] Annie Xie, Frederik Ebert, Sergey Levine, and Chelsea Finn. "Improvisation through physical understanding: Using novel objects as tools with visual foresight". In: *arXiv preprint arXiv:1904.05538* (2019).

[99] Kexin Yi, Jiajun Wu, Chuang Gan, Antonio Torralba, Pushmeet Kohli, and Josh Tenenbaum. "Neural-symbolic vqa: Disentangling reasoning from vision and language understanding". In: *Advances in neural information processing systems* 31 (2018).

[100] Hanna Yousef, Mehdi Boukallel, and Kaspar Althoefer. "Tactile sensing for dexterous in-hand manipulation in robotics—A review". In: *Sensors and Actuators A: physical* 167.2 (2011), pp. 171–187.

[101] Tianhe Yu, Gleb Shevchuk, Dorsa Sadigh, and Chelsea Finn. "Unsupervised Visuomotor Control through Distributional Planning Networks". In: *Proceedings of Robotics: Science and Systems*. Freiburg im Breisgau, Germany, June 2019.

[102] Andy Zeng, Shuran Song, Kuan-Ting Yu, Elliott Donlon, Francois R Hogan, Maria Bauza, Daolin Ma, Orion Taylor, Melody Liu, Eudald Romo, et al. "Robotic pick-and-place of novel objects in clutter with multi-affordance grasping and cross-domain image matching". In: *The International Journal of Robotics Research* 41.7 (2022), pp. 690–705.

[103] Tianhao Zhang, Zoe McCarthy, Owen Jow, Dennis Lee, Xi Chen, Ken Goldberg, and Pieter Abbeel. "Deep imitation learning for complex manipulation tasks from virtual reality teleoperation". In: *2018 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE. 2018, pp. 5628–5635.

[104] Shengjia Zhao, Jiaming Song, and Stefano Ermon. "InfoVAE: Information Maximizing Variational Autoencoders". In: *CoRR* abs/1706.02262 (2017). arXiv: 1706.02262. URL: http://arxiv.org/abs/1706.02262.

[105] Yuke Zhu, Ziyu Wang, Josh Merel, Andrei Rusu, Tom Erez, Serkan Cabi, Saran Tunyasuvunakool, János Kramár, Raia Hadsell, Nando de Freitas, et al. "Reinforcement and Imitation Learning for Diverse Visuomotor Skills". In: *Proceedings of Robotics: Science and Systems*. Pittsburgh, Pennsylvania, June 2018.