Generative and Self-Supervised Learning for Robotics Problems

by

Oliver Limoyo

A thesis submitted in conformity with the requirements
for the degree of Doctor of Philosophy
Graduate Department of Institute for Aerospace Studies
University of Toronto

Abstract

Generative and Self-Supervised Learning for Robotics Problems

Oliver Limoyo
Doctor of Philosophy
Graduate Department of Institute for Aerospace Studies
University of Toronto
2024

There is a tremendous amount of information available in the form of unlabelled data from robotic systems. Without proper models and algorithms, tapping into this trove of data is not possible. In this dissertation, we apply generative and self-supervised learning as a framework to solve robotics-specific problems. Traditionally, supervised learning relies on human-annotated labels as signal for the training process. Generative, self-supervised learning is a promising approach that works with unlabelled data directly. A model that is trained to generate a large amount of unlabelled but structured data with a significantly smaller number of parameters learns to acquire an efficient internal representation of the data itself. We present models and algorithms that leverage generative self-supervision in multiple robotic contexts. First, we solve the classical robotic problems of system identification and inverse kinematics by formulating them as generative modelling problems. Second, we leverage self-supervision in combination with grasp planning, impedance control, and tactile sensing to automatically generate robotic manipulation placement demonstration data in contact-constrained environments. Finally, we directly integrate generative models of vision and language, trained solely via self-supervision, with the perspective-n-point algorithm to produce a language-guided robotic photography framework. We demonstrate our approaches on simulated and real-world datasets as well as various real-world robotic manipulation tasks.

To Monica.

Acknowledgements

This dissertation would not have been possible without the support and encouragement of many incredible people.

Thank you Jon for your guidance and trust in me. Your sense of humour and open-minded approach to advising were greatly appreciated. Angela and Gabe, thank you for providing constructive feedback on my work as my committee members. To my colleagues in STARS Laboratory, your friendships were truly enriching and made this journey much more fun.

To my mom, dad, and popo, your sacrifices and unwavering belief in me have been the foundation of my achievements. I am also grateful to my aunts, uncles, and cousins for always being there for me—I could not have asked for a more supportive family. Finally, to Monica, my partner in all things, thank you for your love, patience, and encouragement. I couldn't have done this without you.

Contents

Notation

x	:	Symbols in this font are real scalars.
$\mathbf{x}$	:	Symbols in this font are real column vectors.
$\mathbf{X}$	:	Symbols in this font are real matrices.
$\mathcal{N}(\boldsymbol{\mu}, \mathbf{R})$	:	Normally distributed with mean $\boldsymbol{\mu}$ and covariance matrix $\mathbf{R}$.
$\mathbb{E}[\cdot]$	:	The expectation operator.
X	:	Random variable corresponding to $\mathbf{x}$
$p(\mathbf{y} \mid \mathbf{x})$	:	Probability that random variable Y takes the value $\mathbf{y}$ given that random variable X takes the value $\mathbf{x}$.
$\underrightarrow{\mathcal{F}}_a$	:	A reference frame in three dimensions.
$\mathbf{0}$	:	The zero vector.
$\mathbf{I}$	:	The identity matrix.
$\mathbf{p}_a^b$	:	Point (a 3×1 vector) b, denoted by the superscript, expressed in $\underrightarrow{\mathcal{F}}_a$, denoted by the subscript.
$\mathbf{R}_{ab}$	:	The 3×3 rotation matrix that transforms vectors from $\underrightarrow{\mathcal{F}}_b$ to $\underrightarrow{\mathcal{F}}_a$: $\mathbf{p}_a = \mathbf{R}_{ab}\mathbf{p}_b$.
$\mathbf{T}_{ab}$	:	The 4×4 transformation matrix that transforms homogeneous points from $\underrightarrow{\mathcal{F}}_b$ to $\underrightarrow{\mathcal{F}}_a$: $\boldsymbol{p}_a = \mathbf{T}_{ab}\boldsymbol{p}_b$.

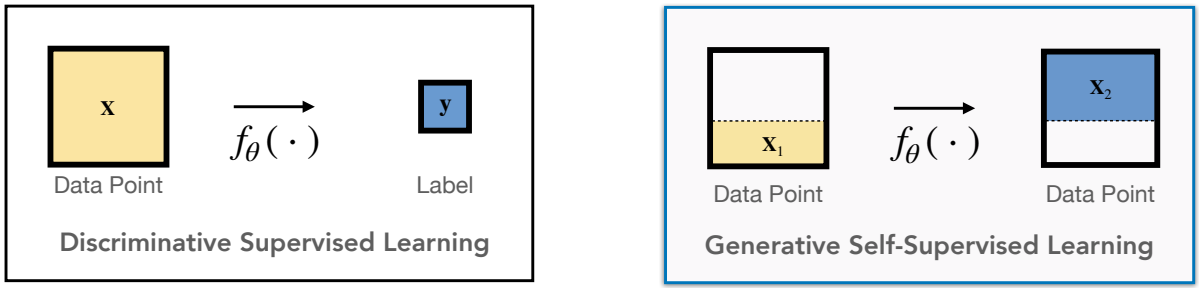
Chapter 1

Introduction

There are myriad ways to stratify and interpret problems in robotics, machine learning, and at the intersection thereof. This dissertation presents a collection of approaches that leverage the generative and self-supervised learning (SSL) paradigm to solve robotics-specific problems.

While supervised learning (SL) has been the basis for many early breakthroughs in deep learning such as ImageNet (Deng et al., 2009), it is fundamentally limited in the amount of signal that can be extracted from data. A single label can only provide a few bits of information at most per data point and often requires significant human labour. A typical example of SL and SSL with an abstract data point $\mathbf{x}$ is shown in Fig. 1.1. In the SSL case, a subset of the data is predicted using the rest and no labels are involved; alternatively, the full datapoint can be predicted without conditioning on any part of the data itself. By predicting a large subset of high-dimensional data, millions of bits of information can be provided per sample. For example, we can predict the future or the past with time-varying data (Srivastava et al., 2015). Or, as commonly done in the SSL community, we can also predict the spatial location of image patches (Doersch et al., 2015), masked images or text (Zhou et al., 2021a; Kenton and Toutanova, 2019) and minimize embedding distances of similar images (He et al., 2020).

The main idea behind self-supervision is to produce a dense amount of labels automatically (i.e., without human labour) by creating a SL task with the patterns and structure of the data itself. By doing so, we can extract an order of magnitude larger signal than with sparse labels. Unlabelled data contain a



(a) **(b)**

Figure 1.1: (a) A datapoint $\mathbf{x}$ is classified with a label $\mathbf{y}_1$ or $\mathbf{y}_2$. (b) Turning a single data point into a generative SSL problem. The function f_θ learns how to predict $\mathbf{x}_2$ based on $\mathbf{x}_1$.

large amount of information. Consider a uniform and random sampling of RGB values in image space. The resulting images will almost always look like gibberish. On the other hand, a robot equipped with a camera will see something completely different: images filled with spatial and temporal patterns (Chapter 3). This idea has also been widely used in language modelling, where a corpus of text provides self-generated labels that can be used for next word prediction given a past sequence. A core hypothesis behind SSL is that a model that learns how to generate a large amount of structured data with a significantly smaller number of parameters must acquire some efficient internal representation of the data itself in order to generate it. For example, a language model may learn both the grammar and the semantic meaning of individual words. Or, a generative model of images of animals could learn useful semantic visual features associated with, for example, posture and various body parts. Beyond just images, videos, and text, signal for SSL can exist more abstractly in the structure and symmetry inherent to robotic problems (Chapter 4, Chapter 5).

Much of this dissertation deals with *generative* self-supervised models, which are a specific class of probabilistic self-supervised models. A generative model learns the distribution of the data itself and can provide information about how likely a given datapoint is. Broadly speaking, all generative models can be considered to be self-supervised. However, not all self-supervised models are generative. For example, some self-supervised models leverage a *contrastive* objective, which involves comparing and contrasting datapoints (Chen et al., 2020) as opposed to generating datapoints.

Generative SSL shares a common probabilistic root with robotics, and we argue forms a natural match. In fact, we show that many robotic problems can be directly cast as generative modelling problems, where the solutions are samples from a learned distribution. Moreover, the probabilistic perspective provides a formalism to describe multiple outcomes and uncertainties in the context of learning, both of which are crucial to robotics. Finally, learning to generate data often leads to useful intermediate representations or features. Generative models have been used in robotics to represent policy classes (Chi et al., 2023), solve inverse kinematics (Lembono et al., 2021), generate goals in reinforcement learning (Nair et al., 2018), and to learn intermediate representations (Sermanet et al., 2018). The same training objectives used to train generative models have also been used to optimize policies in reinforcement learning (Ho and Ermon, 2016)

The approaches that we develop in this dissertation leverage generative self-supervision in four different contexts (Fig. 1.2). We consider the classical robotic problems of system identification with state space models and inverse kinematics through the lens of generative SSL (Chapter 3, Chapter 4), leverage self-supervision to automatically generate training data in the real world (Chapter 5), and, finally, directly integrate trained self-supervised modules into robotic systems (Chapter 6).

1.1 Structure and Contributions

The dissertation is structured as follows. We first provide background material in Chapter 2 that is required to understand the work in the subsequent chapters. Then, in Chapters 3 to 6, we cover the main original contributions of this dissertation. We provide a brief summary of each of these contributions

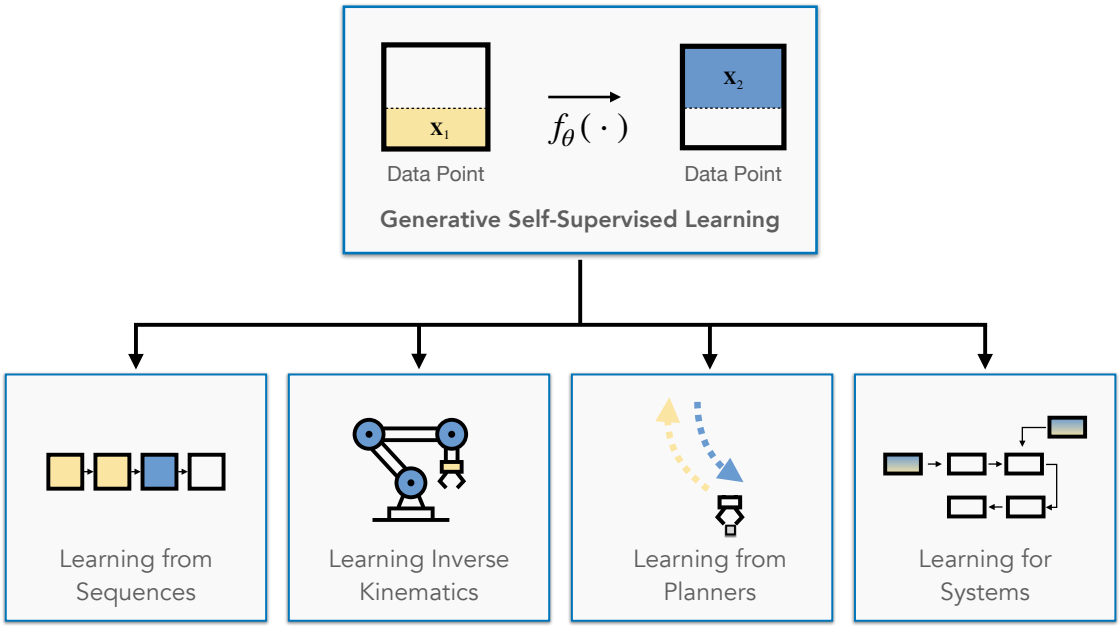


Figure 1.2: We investigate generative self-supervised models in the context of robotics with sequences, kinematics, planners and within overall systems.

on a per-chapter basis below.

1. **Learning Latent State Space Models from Observations**
   In Chapter 3, we present two improvements to learning with state space models (SSMs), where a latent state representation and SSM are learned directly from observations by formulating the problem as generative modelling with sequential data. First, we introduce a method that makes learned SSMs more robust to out-of-distribution inputs. We do this by composing learned deep neural networks with a structured linear Gaussian SSM. We then use the likelihood of the trained generative model to scale the measurement uncertainties of the linear Gaussian SSM. Second, we demonstrate how to extend learning with SSMs to multiple sensing modalities using a principled probabilistic formulation.

2. **Generative Graphical Inverse Kinematics**
   Chapter 4 presents a framework named generative graphical inverse kinematics (GGIK). With GGIK, we formulate inverse kinematics as generative modelling and apply SSL to predict joint angles given end-effector poses. By using a probabilistic generative formulation, we are able to quickly approximate the full solution set of the inverse kinematics problem with a single parallel forward pass of a neural network. Critically, our learned solver is able to generalize across multiple manipulator geometries by using a graph neural network based solver.

3. **Learning to Place by Picking**
   Chapter 5 applies the concept of self-supervision in the pick-and-place context with robot manipulators. We demonstrate how modern grasp planners, tactile sensing and compliant control

can be leveraged to generate expert demonstrations for robotic placement policies in contact-constrained environments. We introduce a method called *placing via picking* (PvP). PvP exploits the symmetric structure of the pick and place problems to generate expert demonstrations of robotic placement in a self-supervised manner.

4. **Language-Guided Robotic Photography**
   In Chapter 6, we show how generative models of vision and language, trained solely via self-supervision, can be integrated to improve classical robotic systems and algorithms. We demonstrate how the intermediate features learned by a generative model of images can be used as useful semantic features, and how language models provide a natural user interface to robotic systems. We present PhotoBot, a framework for fully automated photo acquisition based on an interplay between high-level human language guidance and a robot photographer. At its core, PhotoBot is a system that combines the classical Perspective-n-Point algorithm (Fischler and Bolles, 1981a) with modern self-supervised language and vision models.

Finally, in Chapter 7, we conclude with a summary of the novel contributions of the dissertation and propose several future research directions.

1.2 Associated Publications

This dissertation is comprised of work from the following first-authored papers:

1. Limoyo, O., Chan, B., Maric, F., Wagstaff, B., Mahmood, R., and Kelly, J. (2020b). Heteroscedastic uncertainty for robust generative latent dynamics. *IEEE Robotics and Automation Letters*, 5(4):6654–6661

2. Limoyo, O., Ablett, T., and Kelly, J. (2023a). Learning sequential latent variable models from multimodal time series data. In *Intelligent Autonomous Systems 17*, volume 577 of *Lecture Notes in Networks and Systems*, pages 511–528. Best Paper Finalist

3. Limoyo, O., Maric, F., Giamou, M., Alexson, P., Petrovic, I., and Kelly, J. (2023d). Generative graphical inverse kinematics. *IEEE Transactions on Robotics*. To Appear.

4. Limoyo, O., Maric, F., Giamou, M., Alexson, P., Petrovic, I., and Kelly, J. (2023c). Euclidean equivariant models for generative graphical inverse kinematics. In *Proceedings of the Robotics: Science and Systems (RSS) Workshop on Symmetries in Robot Learning*, Daegu, Republic of Korea

5. Limoyo, O., Konar, A., Ablett, T., Kelly, J., Hogan, F. R., and Dudek, G. (2024a). Working backwards: Learning to place by picking. In *IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS)*, Abu Dhabi, United Arab Emirates. Submitted

6. Limoyo, O., Li, J., Rivkin, D., Kelly, J., and Dudek, G. (2024c). Photobot: Reference-guided interactive photography via natural language. In *IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS)*, Abu Dhabi, United Arab Emirates. Submitted

7. Limoyo, O., Li, J., Rivkhin, D., Kelly, J., and Dudek, G. (2024b). Reference-guided robotic photography through natural language interaction. In *Proceedings of the ACM/IEEE International Conference on Human Robot Interaction (HRI) Workshop on Human—Large Language Model Interaction: The Dawn of a New Era or the End of it All?*, Boulder, Colorado, USA

Chapter 2

Background

In this chapter, we briefly cover the key concepts and relevant background material required to understand the contributions described in this dissertation. We begin by providing a primer on probability and statistics. Then, we introduce relevant theoretical machinery, focusing on probabilistic machine learning. Finally, we consider the deep learning architectures that are essential to the contributions presented herein. The notation and conventions used throughout this dissertation borrow heavily from (Murphy, 2012a) and (Barfoot, 2024).

2.1 Probability and Statistics

We first review fundamental definitions in probability and statistics and introduce some of the probability distributions that we use. Then, we briefly discuss parameter estimation and maximum likelihood estimation (MLE). We also introduce KL divergence as a way to quantify the similarity of two probability distributions.

2.1.1 Preliminaries

A *probability density function*, or PDF, $p(x)$ for the continuous random variable X is a non-negative function defined over an interval $[a, b]$ called the *sample space*. We can calculate the probability that X takes a value in the interval $[c, d]$, for $a \leq c$ and $d \leq b$, as

$$\Pr(c \leq X \leq d) = \int_c^d p(x)dx, \tag{2.1}$$

and where $p(x)$ satisfies the following property

$$\int_a^b p(x)dx = 1. \tag{2.2}$$

The *cumulative distribution function* or CDF of random variable X is defined as

$$P(x) \triangleq \Pr(X \leq x), \tag{2.3}$$

which is the probability that X takes a value no larger than some value x. The CDF is the antiderivative of the PDF, and

$$p(x) \triangleq \frac{d}{dx} P(x). \tag{2.4}$$

We can also evaluate the probability of X taking a value in some finite interval $[c, d]$ with use of the CDF by computing the following

$$\Pr(c \leq X \leq d) = \int_c^d p(x)dx = P(c) - P(d), \tag{2.5}$$

which follows from the fundamental theorem of calculus.

The univariate PDF can be generalized to the joint, D-dimensional density of continuous variables $\mathbf{x} = (x_1, \cdots, x_D)$, where

$$p(\mathbf{x}) = p\left(x_1, x_2, \cdots, x_D\right). \tag{2.6}$$

The joint density of two multidimensional variables $\mathbf{x}$ and $\mathbf{y}$ is sometimes written as

$$p(\mathbf{x}, \mathbf{y}). \tag{2.7}$$

Given the joint density, we can define the marginal density of the random variable $\mathbf{y}$ as

$$p(\mathbf{y}) = \int_a^b p(\mathbf{x}, \mathbf{y})d\mathbf{x}, \tag{2.8}$$

where we sum over all possible $\mathbf{x}$ values. This is often called the *sum rule*. We can also write the joint density as

$$p(\mathbf{x}, \mathbf{y}) = p(\mathbf{y} \mid \mathbf{x})p(\mathbf{x}), \tag{2.9}$$

which is called the *product rule*. If the variables are independent, then we can represent the joint density as the product of two marginals

$$p(\mathbf{x}, \mathbf{y}) = p(\mathbf{y})p(\mathbf{x}). \tag{2.10}$$

An immediate consequence of the product rule is Bayes' rule,

$$p(\mathbf{x} \mid \mathbf{y}) = \frac{p(\mathbf{y} \mid \mathbf{x})p(\mathbf{x})}{p(\mathbf{y})}. \tag{2.11}$$

Bayes' rule can be interpreted as a formula for calculating the PDF of an unknown or hidden random variable $\mathbf{x}$, given some observed data $\mathbf{y}$.

The first moment of a PDF is the *mean*, $\boldsymbol{\mu}$,

$$\boldsymbol{\mu} = \mathbb{E}[\mathbf{x}] = \int_a^b \mathbf{x} p(\mathbf{x}) d\mathbf{x}, \tag{2.12}$$

where $\mathbb{E}[\cdot]$ is the expectation operator. Similarly, the second moment of a PDF, or the covariance, is

$$\boldsymbol{\Sigma} = \mathbb{E}\left[(\mathbf{x} - \boldsymbol{\mu})(\mathbf{x} - \boldsymbol{\mu})^T\right] = \int_a^b (\mathbf{x} - \boldsymbol{\mu})(\mathbf{x} - \boldsymbol{\mu})^T p(\mathbf{x}) d\mathbf{x}. \tag{2.13}$$

2.1.2 Useful Distributions

In this section, we introduce some of the important distributions used in the upcoming chapters. We write the conditioning parameters $\boldsymbol{\theta}$ of the distribution as follows

$$p(\mathbf{x} \mid \boldsymbol{\theta}). \tag{2.14}$$

Similarly, if the parameters of the distribution are themselves parametrized by a neural network, we write

$$p_{\boldsymbol{\phi}}(\mathbf{x} \mid \boldsymbol{\theta}), \tag{2.15}$$

where the neural network parameters are $\boldsymbol{\phi}$ and the distribution parameters are $\boldsymbol{\theta}$. That is, a network parametrized by $\boldsymbol{\phi}$ outputs $\boldsymbol{\theta}$. For the sake of clarity, in many cases we drop the explicit distribution conditioning parameters in the PDF and use $p(\mathbf{x})$ or even $p_{\boldsymbol{\theta}}(\mathbf{x})$ directly as a shorthand when the distribution parameters $\boldsymbol{\theta}$ are themselves based on the parameters of a neural network.

Multivariate Gaussian (Normal) Distribution

The multivariate Gaussian distribution is often used because it is mathematically convenient and the Gaussian assumption is reasonable in many situations. The multivariate Gaussian PDF for a random variable $\mathbf{x} \in \mathbb{R}^D$ is

$$p(\mathbf{x} \mid \boldsymbol{\mu}, \boldsymbol{\Sigma}) = \frac{1}{\sqrt{(2\pi)^D \det(\boldsymbol{\Sigma})}} \exp\left(-\frac{1}{2}(\mathbf{x} - \boldsymbol{\mu})^T \boldsymbol{\Sigma}^{-1} (\mathbf{x} - \boldsymbol{\mu})\right). \tag{2.16}$$

We use the following notation to indicate that $\mathbf{x}$ is normally- or Gaussian-distributed,

$$\mathbf{x} \sim \mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma}). \tag{2.17}$$

Mixture of Gaussians Distribution

We can consider a convex combination of simpler distributions and create a more complex probability distribution called a mixture distribution or model. The general form of the mixture model PDF is

$$p(\mathbf{x} \mid \boldsymbol{\theta}) = \sum_{k=1}^{K} \pi_k \, p_k(\mathbf{x}), \tag{2.18}$$

for K mixture distributions p_k and mixture weights π_k that satisfy $0 \leq \pi_k \leq 1$ and $\sum_{k=1}^{K} \pi_k = 1$. A Gaussian mixture model is defined as follows

$$p(\mathbf{x} \mid \boldsymbol{\theta}) = \sum_{k=1}^{K} \pi_k \, \mathcal{N}\left(\mathbf{x} \mid \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k\right). \tag{2.19}$$

A Gaussian mixture model is capable of representing distributions with multiple modes, which makes the model useful for many learning applications. In fact, a Gaussian mixture model is a universal approximator of an arbitrary density. Any smooth density can be approximated to an arbitrary accuracy by a Gaussian mixture with a sufficient number of components (Goodfellow et al., 2016).

2.1.3 Maximum Likelihood Estimation

Maximum likelihood estimation (MLE) is a common approach to parameter estimation. With MLE, we find the parameters that maximize the probability of the training data. The MLE optimization problem is

$$\boldsymbol{\theta}^*_{mle} \triangleq \operatorname*{argmax}_{\boldsymbol{\theta}} p(\mathcal{D} \mid \boldsymbol{\theta}), \tag{2.20}$$

where $\mathcal{D}$ is the training data. As commonly done, we assume that the training data are independently sampled from the same distribution (or iid for "independent and identically distributed"). The likelihood of the training data (for a supervised learning problem) is then

$$p(\mathcal{D} \mid \boldsymbol{\theta}) = \prod_{n=1}^{N} p(\mathbf{y}_n \mid \mathbf{x}_n, \boldsymbol{\theta}), \tag{2.21}$$

where $\mathbf{y}_n$ is a label and $\mathbf{x}_n$ is a datapoint. The corresponding log likelihood is given by

$$\log p(\mathcal{D} \mid \boldsymbol{\theta}) = \sum_{n=1}^{N} \log p(\mathbf{y}_n \mid \mathbf{x}_n, \boldsymbol{\theta}). \tag{2.22}$$

In practice we often use the negative of the log likelihood to make the optimization problem a minimization problem. In the self-supervised case, we are interested in finding a set of model parameters

(a) Forward KL divergence. **(b)** Reverse KL divergence.

Figure 2.1: Comparison of forward and reverse KL divergence for two Gaussian distributions p and q. The distribution p is denoted in blue and q in orange. Figure from (Ghosh, 2018).

that maximizes the likelihood of the datapoint itself. The log likelihood then becomes

$$\log p(\mathcal{D} \mid \boldsymbol{\theta}) = \sum_{n=1}^{N} \log p(\mathbf{x}_n \mid \boldsymbol{\theta}). \tag{2.23}$$

2.1.4 KL divergence

We will often be interested in measuring a notion of similarity between two probability distributions. Given distributions p and q, we can define a divergence measure $D(p, q)$ that provides a quantifiable measure of how far p is from q. A divergence measure should satisfy the following properties: $D(p, q) \geq 0$, and $D(p, q) = 0$ iff $p = q$. However, such measures are usually not metrics since they do not satisfy the triangle inequality and are not symmetric (i.e., we may not get the same value if we swap p and q). Many divergence measures exist, but we focus on the Kullback-Leibler divergence or KL divergence, which measures the *information gain* or *relative entropy* between two distributions. The KL divergence is defined as

$$\mathrm{KL}(p \,||\, q) \triangleq \int p(\mathbf{x}) \log \frac{p(\mathbf{x})}{q(\mathbf{x})} d\mathbf{x}. \tag{2.24}$$

The KL divergence can sometimes be written in closed form, although this is not true in the general case. As an example, if p and q are both Gaussians, then the KL divergence is given by

$$\begin{aligned} &\mathrm{KL}(\mathcal{N}(\mathbf{x} \mid \boldsymbol{\mu}_1, \boldsymbol{\Sigma}_1) \,||\, \mathcal{N}(\mathbf{x} \mid \boldsymbol{\mu}_2, \boldsymbol{\Sigma}_2)) \\ &= \frac{1}{2}\left[\mathrm{tr}\left(\boldsymbol{\Sigma}_2^{-1}\boldsymbol{\Sigma}_1\right) + (\boldsymbol{\mu}_2 - \boldsymbol{\mu}_1)^T \boldsymbol{\Sigma}_2^{-1}(\boldsymbol{\mu}_2 - \boldsymbol{\mu}_1) - C + \log\left(\frac{\det(\boldsymbol{\Sigma}_2)}{\det(\boldsymbol{\Sigma}_1)}\right)\right], \end{aligned} \tag{2.25}$$

where C is a scalar that is the dimension of $\mathbf{x}$. In the absence of a closed-form expression, we can use a sampling procedure to approximate the KL divergence.

We note that two variants of the KL divergence exist, the forward KL divergence as shown in Eq. (2.24) and the reverse KL divergence, written as

$$\mathrm{KL}(q \,||\, p) \triangleq \int q(\mathbf{x}) \log \frac{q(\mathbf{x})}{p(\mathbf{x})} d\mathbf{x}. \tag{2.26}$$

Since the KL divergence is not a metric, $\mathrm{KL}(q \,||\, p) \neq KL(p \,||\, q)$. The behaviour of the forward and reverse KL divergence is significantly different. The forward KL divergence is zero-avoiding and mean-seeking, which results in a "stretched" q distribution, as shown in Fig. 2.1a (if p and q are both Gaussian distributions). On the other hand, the reverse KL divergence is mode-seeking; the q distribution covers only one of the modes as shown in Fig. 2.1b. In the methods introduced in Chapter 3 and Chapter 4, we use the reverse KL divergence. For many applications, the mode-seeking behaviour is preferable to the "averaging" behaviour of the mean-seeking forward KL divergence. Additionally, in many cases, both p and q are learnable or trainable, and we purposely choose p to be a multimodal distribution with the capacity to represent the q distributions of multiple datapoints.

2.2 Generative Self-Supervised Learning

In this chapter, we provide background theory on the self-supervised variant and focus on problems of the form $p(\mathbf{x} \mid \boldsymbol{\theta})$. We are interested in fitting or learning parameters $\boldsymbol{\theta}$ of the distribution $p(\mathbf{x} \mid \boldsymbol{\theta})$ given a finite sample of data points $\mathbf{x}$, which could be images or texts, for example. A good model will assign high probability to actual data samples and, in turn, implicitly assign lower probabilities to regions where data are sparse or absent. Once trained, the learned distribution can be used to generate new samples.

There are multiple ways to model $p(\mathbf{x} \mid \boldsymbol{\theta})$. For example, we could decompose the distribution over its dimensions directly in an auto-regressive manner by using the product rule or chain rule of probabilities:

$$p(\mathbf{x} \mid \boldsymbol{\theta}) = \prod p(x_1 \mid \boldsymbol{\theta}) p(x_2 \mid x_1, \boldsymbol{\theta}) p(x_3 \mid x_1, x_2, \boldsymbol{\theta}) \cdots p(x_D \mid x_{<D}, \boldsymbol{\theta}). \tag{2.27}$$

We can then fit the parametrized conditional distribution based on the data. In many of the methods presented in this dissertation, we focus on another class of generative models named latent variable models, which we cover in more details in the next section.

2.2.1 Latent Variable Model

We can introduce an unobserved, lower-dimensional *latent variable* $\mathbf{z} \in \mathbb{R}^M$ and express $p(\mathbf{x} \mid \boldsymbol{\theta})$ as

$$p(\mathbf{x} \mid \boldsymbol{\theta}) = \int p(\mathbf{x}, \mathbf{z} \mid \boldsymbol{\theta}) d\mathbf{z} = \int p(\mathbf{x} \mid \mathbf{z}, \boldsymbol{\theta}) p(\mathbf{z} \mid \boldsymbol{\theta}) d\mathbf{z}. \tag{2.28}$$

We drop the explicit distribution parameters $\boldsymbol{\theta}$ for clarity, which results in

$$p(\mathbf{x}) = \int p(\mathbf{x}, \mathbf{z}) d\mathbf{z} = \int p(\mathbf{x} \mid \mathbf{z})\, p(\mathbf{z}) d\mathbf{z}. \tag{2.29}$$

At the core of latent variable models is the assumption that there are underlying factors of variation that explain the data. These underlying factors of variation exist in a lower-dimensional subspace. For example, given a dataset of images of dogs, factors of variation could include the size, shape,

colour, posture, and background. The latent state can be interpreted as a 'bottleneck' through which the information content of the data needs to pass; the latent state therefore also acts as a compressed representation of the data. Using Bayes' rule, we can determine the posterior distribution as

$$p(\mathbf{z} \mid \mathbf{x}) = \frac{p(\mathbf{x} \mid \mathbf{z})\, p(\mathbf{z})}{p(\mathbf{x})}. \tag{2.30}$$

There are many types of latent variable models. If we use a simpler linear function, we recover the model commonly used in factor analysis (Roweis and Ghahramani, 1999), which has the form

$$\mathbf{z} \sim \mathcal{N}(0, \mathbf{I}), \qquad \mathbf{x} \sim \mathcal{N}(\mathbf{W}\mathbf{z} + \boldsymbol{\mu}, \boldsymbol{\Sigma}). \tag{2.31}$$

In this case, the distribution is parametrized by the following set of variables

$$\{\mathbf{W} \in \mathbb{R}^{D \times M}, \boldsymbol{\mu} \in \mathbb{R}^{D}, \boldsymbol{\Sigma} \in \mathbb{R}^{D \times D}\}. \tag{2.32}$$

Moreover, if we set $\boldsymbol{\Sigma} = \sigma^2 \mathbf{I}$, we arrive at the probabilistic principal component analysis (PPCA) model (Tipping and Bishop, 1999). It can be shown that $\mathbf{z}$ converges to the principal components recovered using principal component analysis (PCA) as σ approaches zero. With PPCA, we can conveniently calculate all of the distributions analytically. The true posterior has the closed form expression

$$p(\mathbf{z} \mid \mathbf{x}) = \mathcal{N}\left(\mathbf{M}^{-1}\mathbf{W}^T(\mathbf{x} - \boldsymbol{\mu}), \sigma^2 \mathbf{M}^{-1}\right), \tag{2.33}$$

where $\mathbf{M} = \mathbf{W}^T\mathbf{W} + \sigma^2\mathbf{I}$.

We can also generalize the above to nonlinear factor analysis through a nonlinear transformation of the latent variable model, as shown below by

$$\mathbf{z} \sim \mathcal{N}(0, \mathbf{I}) \qquad \mathbf{x} \sim p(f_{\boldsymbol{\phi}}(\mathbf{z})), \tag{2.34}$$

where $f_{\boldsymbol{\phi}}(\cdot)$ is a nonlinear function parametrized by $\boldsymbol{\phi}$ that outputs the parameters of the distribution p. One particular choice for f is a deep neural network. This problem formulation then becomes one of *deep generative modelling*. We note that the prior distribution of $\mathbf{z}$ does not necessarily need to be constrained to be Gaussian in deep generative modelling. We focus on this class of problems for most of the contributions in this dissertation. With general nonlinear latent variable models, we unfortunately can no longer compute the MLE or the posterior exactly and approximate methods are required. That is, the integrals shown in Eq. (2.29) and Eq. (2.30) do not have analytical solutions.

There is more than one way to approximate the posterior distribution in Eq. (2.30). For example, techniques such as Markov chain Monte Carlo (Hastings, 1970) could be used to generate samples from the true posterior distribution. In this dissertation, we focus on robotic applications with relatively large datasets and complex models. Having said this, in the next chapter we review an alternate family of approximate inference methods falling under the umbrella of variational inference that have good scaling properties for large datasets and complex models.

2.2.2 Variational Inference

In this section we provide a brief overview of variational inference (VI) as a method for approximate probabilistic inference. The literature on variational methods for probabilistic inference is vast and we refer readers to (Jordan et al., 1999a) for a comprehensive treatment of the subject. VI formulates approximate inference as an optimization problem using a family of variational distributions. Specifically, we can approximate the posterior distribution $p(\mathbf{z} \mid \mathbf{x})$ and derive a lower bound on the marginal likelihood or evidence $p(\mathbf{x})$ for model fitting or training with VI.

The log likelihood is typically a differentiable function of the model parameters $\boldsymbol{\theta}$. We first consider the log likelihood of a general latent variable model for a single datapoint $\mathbf{x}$,

$$\log p(\mathbf{x} \mid \boldsymbol{\theta}) = \log \int p(\mathbf{x} \mid \mathbf{z}, \boldsymbol{\theta})\, p(\mathbf{z} \mid \boldsymbol{\theta}) d\mathbf{z}. \tag{2.35}$$

Determining the log likelihood is intractable without an analytic expression for the integral. In variational inference, we approximate the log likelihood with a surrogate lower bound instead of by sampling. To arrive at a lower bound, we introduce a variational distribution $q(\mathbf{z})$, with variational parameters $\boldsymbol{\phi}$, as follows:

$$\begin{aligned}
\log p(\mathbf{x} \mid \boldsymbol{\theta}) &= \log \int p(\mathbf{x} \mid \mathbf{z}, \boldsymbol{\theta}) p(\mathbf{z} \mid \boldsymbol{\theta}) d\mathbf{z} \\
&= \log \int \frac{p(\mathbf{x} \mid \mathbf{z}, \boldsymbol{\theta}) p(\mathbf{z} \mid \boldsymbol{\theta})}{q(\mathbf{z} \mid \boldsymbol{\phi})} q(\mathbf{z} \mid \boldsymbol{\phi}) d\mathbf{z} \\
&= \log \mathbb{E}_{q(\mathbf{z}|\boldsymbol{\phi})} \left[\frac{p(\mathbf{x} \mid \mathbf{z}, \boldsymbol{\theta}) p(\mathbf{z} \mid \boldsymbol{\theta})}{q(\mathbf{z} \mid \boldsymbol{\phi})} \right] \\
&\geq \underbrace{\mathbb{E}_{q(\mathbf{z}|\boldsymbol{\phi})} \left[\log \frac{p(\mathbf{x} \mid \mathbf{z}, \boldsymbol{\theta}) p(\mathbf{z} \mid \boldsymbol{\theta})}{q(\mathbf{z} \mid \boldsymbol{\phi})} \right]}_{\mathcal{L}(\boldsymbol{\theta}, q)}.
\end{aligned} \tag{2.36}$$

We can swap the logarithm and expectation terms on the last line with use of Jensen's inequality, given that the logarithm function is concave. The lower bound in Eq. (2.36) is called the evidence lower bound (ELBO). We note that the logarithm is inside the expectation term and we can use sampling to obtain an unbiased Monte Carlo estimate. We use the lower bound as our objective function and maximize the bound with respect to the parameters $\boldsymbol{\theta}$ and distribution $q(\mathbf{z} \mid \boldsymbol{\phi})$.

The ELBO coincides with the log likelihood if and only if $q(\mathbf{z} \mid \boldsymbol{\phi})$ is the true posterior distribution $p(\mathbf{z} \mid \mathbf{x}, \boldsymbol{\theta})$. In other words, the best choice for the distribution of $q(\mathbf{z} \mid \boldsymbol{\phi})$ is the posterior distribution

itself. We can demonstrate this by rewriting the ELBO as

$$\begin{aligned}
\mathcal{L}(\boldsymbol{\theta}, q) &= \mathbb{E}_{q(\mathbf{z}|\boldsymbol{\phi})}\left[\log \frac{p(\mathbf{x} \mid \mathbf{z}, \boldsymbol{\theta}) p(\mathbf{z} \mid \boldsymbol{\theta})}{q(\mathbf{z} \mid \boldsymbol{\phi})}\right] \\
&= \mathbb{E}_{q(\mathbf{z}|\boldsymbol{\phi})}\left[\log \frac{p(\mathbf{x}, \mathbf{z} \mid \boldsymbol{\theta})}{q(\mathbf{z} \mid \boldsymbol{\phi})}\right] \\
&= \mathbb{E}_{q(\mathbf{z}|\boldsymbol{\phi})}\left[\log p(\mathbf{x} \mid \boldsymbol{\theta}) p(\mathbf{z} \mid \mathbf{x}, \boldsymbol{\theta}) - \log q(\mathbf{z} \mid \boldsymbol{\phi})\right] \\
&= \log p(\mathbf{x} \mid \boldsymbol{\theta}) - \mathbb{E}_{q(\mathbf{z}|\boldsymbol{\phi})}\left[-p(\mathbf{z} \mid \mathbf{x}, \boldsymbol{\theta}) + \log q(\mathbf{z} \mid \boldsymbol{\phi})\right] \\
&= \log p(\mathbf{x} \mid \boldsymbol{\theta}) - KL(q(\mathbf{z} \mid \boldsymbol{\phi}) \,||\, p(\mathbf{z} \mid \mathbf{x}, \boldsymbol{\theta})).
\end{aligned} \tag{2.37}$$

Rearranging, we obtain,

$$\mathrm{KL}(q(\mathbf{z} \mid \boldsymbol{\phi}) \,||\, p(\mathbf{z} \mid \mathbf{x}, \boldsymbol{\theta})) = \log p(\mathbf{x} \mid \boldsymbol{\theta}) - \mathcal{L}(\boldsymbol{\theta}, q). \tag{2.38}$$

The gap between the marginal loglikelihood $\log p(\mathbf{x} \mid \boldsymbol{\theta})$ and the ELBO $\mathcal{L}(\boldsymbol{\theta}, q)$ is the KL divergence between $q(\mathbf{z} \mid \boldsymbol{\phi})$ and $p(\mathbf{z} \mid \mathbf{x}, \boldsymbol{\theta})$, and the KL divergence is only zero for two identical distributions.

In most cases, we do not know the form of the posterior distribution $p(\mathbf{z} \mid \mathbf{x}, \boldsymbol{\theta})$ and the choice of variational distribution $q(\mathbf{z} \mid \boldsymbol{\phi})$ is a design decision. In practice, the distribution family of $q(\mathbf{z} \mid \boldsymbol{\phi})$ is often restricted to be something that is flexible enough to provide a good approximation of the true posterior while being simple enough to make the ELBO computation feasible. Once the family of the variational distribution is fixed or chosen, maximizing the ELBO with respect to $q(\mathbf{z} \mid \boldsymbol{\phi})$ then becomes a maximization based on the parameters $\boldsymbol{\phi}$. That is, our lower bound objective function can be written as $\mathcal{L}(\boldsymbol{\theta}, \boldsymbol{\phi})$ instead of $\mathcal{L}(\boldsymbol{\theta}, q)$.

Variational inference is an active field of research and multiple solutions have been proposed to maximize the lower bound. One common solution is to combine the coordinate ascent VI algorithm (CAVI) (Bishop, 2006) with a mean-field approximation (Opper and Saad, 2001), where the variational posterior is factorized as

$$q(\mathbf{z} \mid \boldsymbol{\phi}) = \prod_{m=1}^{M} q_m(z_n \mid \boldsymbol{\phi}_m). \tag{2.39}$$

The mean-field approximation reduces the complexity of the problem by allowing the separate optimization of each factor q_j of the mean-field distribution while holding the others fixed. In fact, it can be shown that with a mean-field approximation, a closed form solution exists for the variational parameters $\boldsymbol{\phi}$ as long as the conditional densities of q belong to the exponential family (Blei et al., 2017). Mean-field VI, however, does not scale well to larger datasets that are the primary focus of this dissertation. Moreover, the mean-field assumption is a strong restriction on the variational posterior that ignores correlations between variables.

Stochastic VI (SVI) (Hoffman et al., 2013) uses gradient-based search and stochastic optimization (Robbins and Monro, 1951) to tackle scalability. As opposed to having to use the full dataset, SVI can be used with batches of data to estimate the full ELBO. SVI consists of two main steps. First, the ELBO

is optimized with respect to the variational parameters:

$$\boldsymbol{\phi} \leftarrow \boldsymbol{\phi} + \alpha \nabla_{\boldsymbol{\phi}} \mathbb{E}_{q(\mathbf{z}|\boldsymbol{\phi})} \left[\log p(\mathbf{x}, \mathbf{z} \mid \boldsymbol{\theta}) - \log q(\mathbf{z} \mid \boldsymbol{\phi}) \right]. \tag{2.40}$$

Then, while fixing the variational parameters, the ELBO is optimized with respect to the model parameters:

$$\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} + \alpha \nabla_{\boldsymbol{\theta}} \mathbb{E}_{q(\mathbf{z}|\boldsymbol{\phi})} \left[\log p(\mathbf{x}, \mathbf{z} \mid \boldsymbol{\theta}) - \log q(\mathbf{z} \mid \boldsymbol{\phi}) \right]. \tag{2.41}$$

We can obtain unbiased estimates of the gradient update in Eq. (2.41) via Monte Carlo sampling by moving the gradient operator inside the expectation. On the other hand, the gradient update in Eq. (2.40) is trickier to deal with. The gradient operator cannot be easily moved inside the expectation because the expectation is taken with respect to the distribution that we are differentiating. To evaluate Eq. (2.40), the authors of (Hoffman et al., 2013) calculate analytical gradients with respect to the variational parameters $\boldsymbol{\phi}$ by making strong assumptions regarding the conditional probabilities. With SVI, each datapoint is assigned its own specific set of variational parameters, which are then optimized in the first stage. In turn, the number of variational parameters directly scales with the number of datapoints. It can be unwieldy to have to optimize a set of variational parameters for every new datapoint.

Amortized VI (AVI) is a method introduced in (Rezende et al., 2014a; Kingma and Welling, 2014) that uses a separate parametrized function to predict the variational parameters based on the data. Crucially, we no longer need to track and optimize separate variational parameters for each datapoint because the variational parameters are *amortized* or shared across all data points. The parametrized function is typically a neural network with parameters $\boldsymbol{\phi}$ called the *inference network* or *encoder*. Instead of using $q(\mathbf{z} \mid \boldsymbol{\phi})$, we now use

$$q_{\boldsymbol{\phi}}(\mathbf{z} \mid \mathbf{x}), \tag{2.42}$$

where the distribution parameters of q are outputs of the neural network. Concretely, if $q_{\boldsymbol{\phi}}(\mathbf{z} \mid \mathbf{x})$ is a Gaussian distribution, then we have

$$\boldsymbol{\mu} = NN^1_{\boldsymbol{\phi}}(\mathbf{x}), \qquad \boldsymbol{\Sigma} = NN^2_{\boldsymbol{\phi}}(\mathbf{x}), \tag{2.43}$$

where NN is a function that represents a forward pass through a neural network. Conveniently, with amortization, we can deal with new datapoints without having to first run an expensive optimization procedure based on the ELBO. However, sharing parameters across all datapoints introduces some additional errors when compared to the more traditional approach of having a separate set of optimized variational parameters per datapoint. If we use a neural network to parametrize the distribution $p_{\boldsymbol{\theta}}(\mathbf{x}, \mathbf{z})$ or a *decoder* network, our final lower bound can be written as

$$\mathcal{L}(\boldsymbol{\theta}, \boldsymbol{\phi}) = \underbrace{\mathbb{E}_{q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x})} \left[\log p_{\boldsymbol{\theta}}(\mathbf{x} \mid \mathbf{z}) \right]}_{Reconstruction} - \underbrace{\mathrm{KL}(q_{\boldsymbol{\phi}}(\mathbf{z} \mid \mathbf{x}) \,||\, p_{\boldsymbol{\theta}}(\mathbf{z}))}_{Regularization}, \tag{2.44}$$

where we split the bound (Eq. (2.36)) into a reconstruction term and a regularization term. The reconstruction term encourages both the encoder and decoder to reconstruct the data. The regularization

term encourages the distribution from the encoder not to stray too far from the prior distribution. With this formulation, optimization can be performed in a single step where the encoder parameters $\boldsymbol{\phi}$ and decoder parameters $\boldsymbol{\theta}$ are jointly updated via gradient descent. The authors of (Kingma and Welling, 2014) call this encoder and decoder pair a *variational autoencoder* (VAE).

Calculating the gradient update $\nabla_{\boldsymbol{\phi}} \mathbb{E}_{q(\mathbf{z}|\boldsymbol{\phi})} \left[\log p(\mathbf{x}, \mathbf{z} \mid \boldsymbol{\theta}) - \log q(\mathbf{z} \mid \boldsymbol{\phi})\right]$ from Eq. (2.40) remains a challenge for VAEs. With SVI, simplifying assumptions can be made to calculate analytical gradients, but this is not possible with VAEs which leverage deep neural networks.

A plausible Monte Carlo gradient estimator is

$$\begin{aligned} \nabla_{\boldsymbol{\phi}} \mathbb{E}_{q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x})} \left[f(\mathbf{x}, \mathbf{z})\right] &= \mathbb{E}_{q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x})} \left[f(\mathbf{x}, \mathbf{z}) \nabla_{\boldsymbol{\phi}} \log q_{\boldsymbol{\phi}}(\mathbf{z} \mid \mathbf{x})\right] \\ &\approx \frac{1}{N} \sum_{n=1}^{N} f(\mathbf{x}_n, \mathbf{z}_n) \nabla_{\boldsymbol{\phi}} \log q_{\boldsymbol{\phi}}(\mathbf{z}_n \mid \mathbf{x}_n). \end{aligned} \tag{2.45}$$

This gradient estimator belongs to the general family of score function estimators, which include the REINFORCE algorithm (Williams, 1992). Unfortunately, a gradient estimator of this form exhibits very high variance (Rezende et al., 2014a). Alternatively, the authors of (Rezende et al., 2014a; Kingma and Welling, 2014) propose a low-variance gradient estimator for VAEs using the *reparametrization trick*. If we assume that q is a Gaussian distribution, we can generate random samples from q using

$$\mathbf{z} = g_{\boldsymbol{\phi}}(\boldsymbol{\epsilon}, \mathbf{x}) = \boldsymbol{\mu} + \boldsymbol{\sigma}\boldsymbol{\epsilon}, \tag{2.46}$$

where $\boldsymbol{\epsilon} \sim p(\boldsymbol{\epsilon})$, $p(\boldsymbol{\epsilon})$ is the distribution $\mathcal{N}(\mathbf{0}, \mathbf{I})$, and both $\boldsymbol{\mu}$ and $\boldsymbol{\sigma}$ come from the neural network parametrized by $\boldsymbol{\phi}$. We remove the dependence of the expectation on the parameters $\boldsymbol{\phi}$ and move the randomness to an independent variable $\boldsymbol{\epsilon}$ as shown by

$$\mathbb{E}_{q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x})} \left[f(\mathbf{x}, \mathbf{z})\right] = \mathbb{E}_{p(\boldsymbol{\epsilon})} \left[f(\mathbf{x}, g_{\boldsymbol{\phi}}(\boldsymbol{\epsilon}, \mathbf{x}))\right]. \tag{2.47}$$

By doing so, we are able to calculate gradients with respect to a deterministic function with the chain rule. That is, we can now move the gradient operator inside the expectation,

$$\nabla_{\boldsymbol{\phi}} \mathbb{E}_{q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x})} \left[f(\mathbf{x}, \mathbf{z})\right] = \nabla_{\boldsymbol{\phi}} \mathbb{E}_{p(\boldsymbol{\epsilon})} \left[f(\mathbf{x}, g_{\boldsymbol{\phi}}(\boldsymbol{\epsilon}, \mathbf{x}))\right] = \mathbb{E}_{p(\boldsymbol{\epsilon})} \left[\nabla_{\boldsymbol{\phi}} f(\mathbf{x}, g_{\boldsymbol{\phi}}(\boldsymbol{\epsilon}, \mathbf{x}))\right]. \tag{2.48}$$

The estimator defined by Eq. (2.48) is sometimes called a pathwise derivative estimator (Glasserman, 2004).

We use ancestral sampling with $p_{\boldsymbol{\theta}}(\mathbf{z})$ and $p_{\boldsymbol{\theta}}(\mathbf{x} \mid \mathbf{z})$ respectively to generate samples from a trained model. Unfortunately, we have no control over the data generation process with standard VAEs. In many applications, including robotics, we would often like some control over the samples that are generated. For example, we may want to generate data based on specific control inputs (Chapter 3) or end-effector poses (Chapter 4). The conditional variational autoencoder (CVAE) (Sohn et al., 2015a) is an extension of the VAE that has been developed to tackle this problem. We model the latent variable

and data jointly with the VAE. The CVAE also models the latent variable and data jointly but, crucially, conditioned on an additional random variable $\mathbf{c}$, which results in the following modified lower bound,

$$\log p(\mathbf{x} \mid \mathbf{c}, \boldsymbol{\theta}) \geq \mathbb{E}_{q(\mathbf{z}|\mathbf{c},\boldsymbol{\phi})}\left[\log \frac{p(\mathbf{x} \mid \mathbf{z}, \mathbf{c}, \boldsymbol{\theta}) p(\mathbf{z} \mid \mathbf{c}, \boldsymbol{\theta})}{q(\mathbf{z} \mid \mathbf{c}, \boldsymbol{\phi})}\right]. \tag{2.49}$$

As before, we conditionally sample from $p_{\boldsymbol{\theta}}(\mathbf{z} \mid \mathbf{c})$ and $p_{\boldsymbol{\theta}}(\mathbf{x} \mid \mathbf{z}, \mathbf{c})$ to generate our data. Most important, we can control the sample generation procedure by modifying the value of the input $\mathbf{c}$.

2.3 Deep Learning

Most of the algorithms presented in this dissertation rely on neural networks and methods that are broadly called *deep learning* (Goodfellow et al., 2016). A basic premise of deep learning is that data contains regularity and structure that can be captured with the right *inductive biases* (Mitchell, 2008). Multiple different models with varying architectures and sets of parameters can fit a limited training set equally well. However, each of these models may have drastically different generalization behaviours for inputs not in the training set. Having said this, inductive biases are used to guide the model towards certain, preferred behaviour. Before even considering any learning algorithm or data, the choice of the network architecture implicitly incorporates inductive biases as priors on the representation that will be learned. Priors can include modularity, translation invariance or equivariance, permutation invariance or equivariance, hierarchy, and many others. In this section we cover four network architectures that we use in the dissertation.

2.3.1 Fully Connected Neural Networks

A single layer of a standard fully connected neural network defines a nonlinear transformation that maps an input $\mathbf{z}_l$ to an output $\mathbf{z}_{l+1}$,

$$\mathbf{z}_{l+1} = \mathbf{f}_l(\mathbf{z}_l) = \boldsymbol{\phi}(\mathbf{W}_l \mathbf{z}_l + \mathbf{b}_l), \tag{2.50}$$

where $\mathbf{W}_l \in \mathbb{R}^{D_{l_{out}} \times D_{l_{in}}}$ is the weight matrix, $\mathbf{b}_l \in \mathbb{R}^{D_{l_{out}}}$ is the bias, and $\boldsymbol{\phi}$ is an element-wise nonlinear function or *activation*. In this case, the weight matrix is said to have $D_{l_{out}}$ neurons. A single layer multiplies and sums each input with an extra offset or bias added in before passing through a nonlinearity.

Importantly, without a nonlinearity, the neural network would be equivalent to a linear regressor since the composition of two linear functions, $\mathbf{f}_1$ and $\mathbf{f}_2$, $\mathbf{f} = \mathbf{f}_1 \circ \mathbf{f}_2$ is also linear. A common choice of activation is the rectified linear unit (ReLU),

$$\boldsymbol{\phi}(\mathbf{z}) = \max\{0, \mathbf{z}\}. \tag{2.51}$$

If we consider the one-dimensional case, where $\mathbf{z} \in \mathbb{R}$, the addition of the ReLU activation turns a single layer of a neural network into a continuous, piecewise linear function. The number of neurons

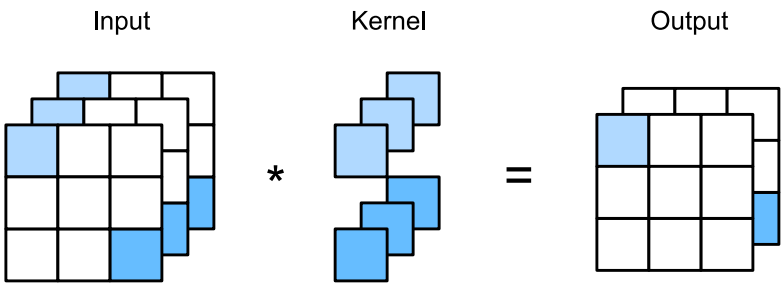


Figure 2.2: A convolutional layer with two 1×1 kernels applied to a two dimensional image with three channels (i.e., $k = 2, w = 1, c = 3$). The output image has two channels since two kernels were used. Figure from (Zhang et al., 2023).

determines the number of piecewise linear segments. The one-dimensional case with a ReLU network provides an intuitive and visual understanding of the universal approximation theorem (Hornik et al., 1989): any one-dimensional continuous function can be approximated to an arbitrary accuracy with enough line segments (i.e., with enough neurons).

A fully connected neural network with L layers is defined by the composition

$$NN_{\boldsymbol{\pi}}(\mathbf{x}) = \mathbf{f}_L \circ \mathbf{f}_{L-1} \circ \cdots \circ \mathbf{f}_1(\mathbf{x}), \tag{2.52}$$

where $f \circ g \triangleq f(g(\cdot))$ is defined as function composition and $\boldsymbol{\pi} = \{\mathbf{W}_l, \mathbf{b}_l\}_{l=1}^{L}$ are the parameters of the network.

2.3.2 Convolutional Neural Networks

The convolutional neural network (CNN) (LeCun et al., 1989) is a network architecture that applies one or more convolutions. CNNs have inductive biases that make them particularly useful for processing high-dimensional image data. Notably, CNNs are *equivariant* to translations. In other words, if an object is translated in an input image, for example, the output features of the CNN will also be equivalently translated. Furthermore, CNNs can process high-dimensional data efficiently by reusing shared weights. Whereas a fully connected network has one parameter per input-output pair, the number of parameters of a kernel from a CNN does not scale up with the input dimension.

The convolution of two functions f and g, denoted as $f * g$, is defined by the integral

$$s(t) = (f * g)(t) = \int_{-\infty}^{\infty} f(s)g(t-s)ds, \tag{2.53}$$

where the function f is the *input*, g is the *kernel*, and the function s is the *output*. The convolution operator can be thought of as the transformation of a continuous signal $f(t)$ into a new signal $s(t)$.

It is also useful to define the discrete version of the convolution operation, which CNN layers actually apply in practice. For example, considering a discrete and finite two-dimensional array $\mathbf{I}_l \in$

$\mathbb{R}^{H\times W\times 1}$ representing a single channel image, a convolution is defined as

$$\mathbf{I}_{l+1}(i,j) = \sum_m \sum_n \mathbf{I}_l(i-m, j-n)\, \mathbf{K}_l(m,n). \tag{2.54}$$

The kernel is represented as a matrix of varying size w in the discrete case (i.e., $\mathbf{K}_l \in \mathbb{R}^{w\times w}$). Typically, a kernel would have size $w = 3$ or $w = 5$. We compute Eq. (2.54), at specific spatial intervals called *strides*. In practice most neural network libraries implement the convolution operation using the related cross-correlation operation defined as

$$\mathbf{I}_{l+1}(i,j) = \sum_m \sum_n \mathbf{I}_l(i+m, j+n)\, \mathbf{K}_l(m,n). \tag{2.55}$$

Lastly, the general convolution operation may have k kernels and be applied to images with c channels, as shown by

$$\mathbf{I}_{l+1}(i,j,k) = \sum_c \sum_m \sum_n \mathbf{I}_l(i+m, j+n, c)\, \mathbf{K}_{l,k}(m,n,c), \tag{2.56}$$

where $\mathbf{K}_{l,k} \in \mathbb{R}^{w\times w\times c}$, $\mathbf{I}_l \in \mathbb{R}^{H\times W\times c}$, and k is the number of kernels chosen by the user. Note that the output image will then have k channels associated with the k kernels. We visualize a simple example of this operation in Fig. 2.2. The goal of a learning algorithm for a CNN is to find the values of $\mathbf{K}_{l,k}$ and potentially an extra bias term, for all k and l.

2.3.3 Recurrent Neural Networks

Recurrent neural networks (RNNs) are a class of neural networks that enable previous outputs to be reused as inputs, while also propagating a hidden or 'memory' state. RNNs are widely used with sequential data of varying length (e.g., text, video, and audio) to capture temporal correlations. The update equation is

$$\mathbf{h}_{l+1} = \mathbf{f}_{\boldsymbol{\pi}}(\mathbf{x}_l, \mathbf{h}_l) = \boldsymbol{\phi}(\mathbf{W}^{hx}\mathbf{x}_l + \mathbf{W}^{hh}\mathbf{h}_l). \tag{2.57}$$

Semantically, we can index the layers in terms of the steps of a temporal sequence

$$\mathbf{h}_{t+1} = \mathbf{f}_{\boldsymbol{\pi}}(\mathbf{x}_t, \mathbf{h}_t) = \boldsymbol{\phi}(\mathbf{W}^{hx}\mathbf{x}_t + \mathbf{W}^{hh}\mathbf{h}_t), \tag{2.58}$$

since each each layer of the RNN usually processes data from one time step in the sequence. The initial state $\mathbf{h}_0$ is typically learned or fixed to be $\mathbf{0}$. If we are interested in regressing a separate variable $\mathbf{y}$, we can include an extra neural network to map the hidden state to $\mathbf{y}$ as

$$\mathbf{y}_t = \mathbf{g}_{\boldsymbol{\pi}}(\mathbf{h}_t) = \boldsymbol{\phi}(\mathbf{W}^{yh}\mathbf{h}_t + \mathbf{b}_y). \tag{2.59}$$

Note that the learnable parameters $\boldsymbol{\pi} = \{\mathbf{W}^{wh}, \mathbf{W}^{hh}, \mathbf{W}^{yh}, \mathbf{b}_y\}$ are typically shared and reused across each layer or time step. The vanilla RNN update shown in Eq. (2.57) is known to produce

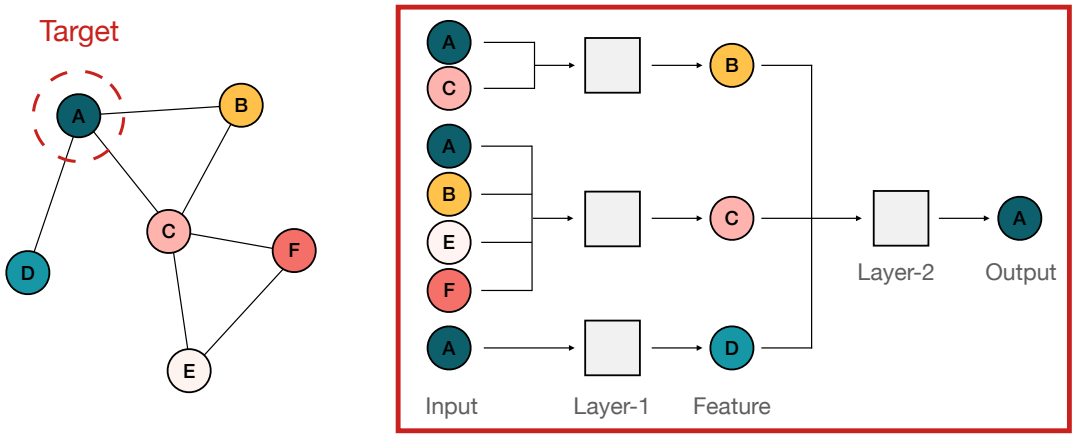


Figure 2.3: A visual depiction of a two-layer GNN and the unrolled computational graph involved in updating node A. The grey boxes represent the combined messaging, aggregation and updating operation shown in Eq. (2.60). Similar to a CNN, the weights of the networks from the same layers are shared. The weights of the networks between different layers are sometimes shared.

vanishing or exploding gradients when backpropagating over longer sequences. The transition function $\mathbf{f}_{\boldsymbol{\pi}}$ needs to be a differentiable function that can also capture long-term dependencies. To address this issue, different functions for $\mathbf{f}_{\boldsymbol{\pi}}$ are often used. This includes the memory cell units used in the long short-term memory (LSTM) network (Hochreiter and Schmidhuber, 1997) and the gated recurrent units (GRU) network (Chung et al., 2014). Memory cell units can pass along and also 'forget' information via a gating mechanism. Gating works by allowing the network to decide how much of the previous hidden state to propagate forward and how much of the new observation to incorporate into the hidden state via elementwise multiplications with weighing variables.

2.3.4 Graph Neural Networks

Graph neural networks (GNNs) are permutation-invariant networks that operate on graph structured data (Bronstein et al., 2021). A graph is a tuple $G = (V, E)$, with a set of nodes $v_i \in V$ and a set of edges $e_{ij} \in E$. A graph models pairwise relationships between nodes. Related pairs of nodes are called edges. During operation, a GNN layer applies three main operations: messaging, aggregating and updating. Following the notation from (Gilmer et al., 2017), the three operations can be written as, respectively,

$$\begin{aligned} \mathbf{m}_{ij} &= \boldsymbol{\phi}_m(\mathbf{h}_i^l, \mathbf{h}_j^l, \mathbf{a}_{ij}) \\ \mathbf{m}_i &= \sum_{j \in \mathcal{N}(i)} \mathbf{m}_{ij} \\ \mathbf{h}_i^{l+1} &= \boldsymbol{\phi}_u(\mathbf{h}_i^l, \mathbf{m}_i), \end{aligned} \tag{2.60}$$

where $\mathbf{h}_i^l \in \mathbb{R}^{D_{node}}$ is the D-dimension embedding vector of a node v_i at layer l, $\mathbf{a}_{ij} \in \mathbb{R}^{D_{edge}}$ is the edge attribute of edge e_{ij}, and $\mathcal{N}(i)$ is the set of neighbours of node v_i. The functions $\boldsymbol{\phi}_m$ and $\boldsymbol{\phi}_u$ are the respective messaging and updating operations. The messaging and updating functions are

typically approximated using neural networks. The aggregation in Eq. (2.60) is carried out by summing up the message from all of the neighbouring nodes. Other aggregation operations are also possible, such as averaging or taking the maximum. A visual depiction of the computations involved to update a single node in a two-layer GNN is shown in Fig. 2.3. Both the CNN and RNN architectures can be considered special cases of the general GNN architecture with specific node connectivity patterns.

Chapter 3

Learning Latent State Space Models from Observations

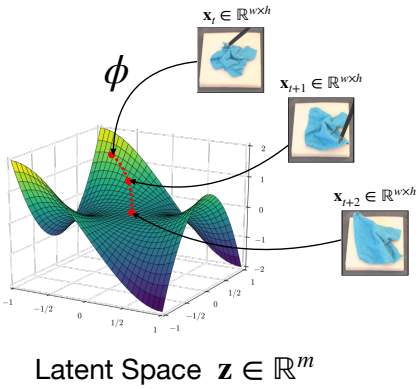
Sequential modelling of temporal data is a general problem that appears in various domains including model-based reinforcement learning, dynamics identification, and learning with language models. In robotics, state space models (SSMs) are used as a standard tool for prediction and inference in the sequential setting. However, for many applications, the task-relevant Lagrangian state (i.e., the minimal state that describes the physics of the world) cannot be directly measured or observed. Or, an *explicit state* that we can interpret symbolically is not readily obvious and an *implicit representation* that is learned but not easily interpretable is preferred. In these cases, it is appealing to use a learned hidden or latent representation directly recovered from observations. Furthermore, for complex systems, hand-engineered models may fail to adequately capture the underlying state transition dynamics.

In this chapter, we formulate SSMs as generative models or, specifically, sequential latent variable (SLV) models, and demonstrate how a latent representation of the state and the accompanying transition dynamics can be learned or identified directly from observations. The temporal structure of the observations provides a rich signal for SSL without the need for any explicit labels. We use the term *latent SSM* and *SLV model* synonymously for this type of model throughout the chapter.

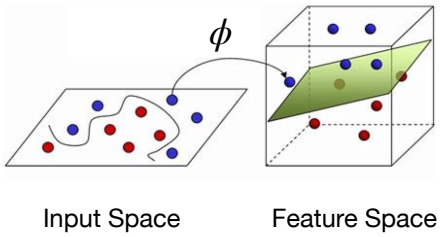
We present two improvements to learning with latent SSMs. To begin, we present a method to make learned SSMs more robust to out-of-distribution (OOD) inputs by adding the structure of the Kalman filter into the learned SLV model. Then, we use the likelihood of the trained generative model to scale measurement uncertainties. Additionally, we demonstrate how to extend learning with SLVs to multiple sensing modalities through a principled, probabilistic formulation.

3.1 Motivation

Latent variable models that leverage the rich representational capacity of deep networks are used to learn from sequential data. These methods embed high-dimensional observations into a lower-dimensional representation space, or *latent space*, often by means of a variational autoencoder (VAE)



(a) Mapping observations to latent space.



(b) Mapping input space to feature space.

Figure 3.1: We identity or learn latent SSMs directly from observations using SSL. (a) With learned SSMs, the function ϕ maps observations (image sequence from (Hoque et al., 2022)) to a manifold where the transition model can be identified. (b) Analogously, with an SVM, the function ϕ learns a mapping from input space to a feature space where the data can be linearly separated.

(Kingma and Welling, 2014; Rezende et al., 2014b), where the (latent) dynamics can be identified in a self-supervised manner (Fig. 3.1). Such models form the backbone of many methods for image-based control, reinforcement learning (Hafner et al., 2021; Rafailov et al., 2021), and motion planning for complex robotic systems (Ichter and Pavone, 2019; Lippi et al., 2020).

When dealing with images as a single modality, latent variable models for sequential data have been shown to be particularly effective in various existing works (Zhang et al., 2019a; Wahlström et al., 2015; Watter et al., 2015; Karl et al., 2017; Fraccaro et al., 2017a). However, existing methods have not yet been adapted for real-world robotic systems. Namely, these systems will need to operate in environments where unexpected conditions (e.g., sensor noise, lighting variations, and unexpected obstructions) may occur. Moreover, in many robotic deployments, images are accompanied by data from multiple additional sensing modalities with varying characteristics, that is, as *multimodal* data. These multimodal data may contain complementary information, patterns, and useful statistical correlations across modalities.

In this chapter, we describe two learning methods. We propose a self-supervised approach to jointly learn a probabilistic state representation and the associated dynamics in a way that makes them amenable for long-term planning and closed-loop control under perceptually difficult conditions. We take inspiration from recent papers that show how the structure of a linear Gaussian state space model (LGSSM) can be composed with one or more neural networks (Karl et al., 2017; Fraccaro et al., 2017b; Chiappa and Piquet, 2019; Johnson et al., 2016).

In addition, we present a novel probabilistic framework for learning latent dynamics from multimodal time series data. Inspired by the multimodal variational autoencoder (MVAE) architecture (Wu and Goodman, 2018), we employ a product of experts (Hinton, 2002) to encode all data modalities into a shared probabilistic latent representation while jointly learning the dynamics in a self-supervised manner with a recurrent neural network (RNN) Cho et al. (2014).

3.2 Related Work

We cover related literature on learning SSMs and novelty detection. We also review previous work that investigates the use of multimodal data in the general context of learning and, more specifically, learning for robotics.

3.2.1 Learning State Space Models

The literature on learned SSMs (Ghahramani and Hinton, 2000; Deisenroth and Rasmussen, 2011) and system identification (Ljung, 1998) is vast and diverse. Bearing in mind our application, we focus our review on prior and proximal work related to learning SSMs from high dimensional observations; we view the problem through the lens of deep generative models or, more broadly, self-supervised representation learning of sequential data.

A significant portion of the existing sequential modelling literature has built upon either the autoencoder framework or its variational alternative (Kingma and Welling, 2014; Rezende et al., 2014b). The authors of (Wahlström et al., 2015) use the bottleneck of an autoencoder as the state of a deep dynamical model parameterized by a feedforward neural network. The model is trained with a reconstruction loss based on a one-step prediction. In (Watter et al., 2015) and (Banijamali et al., 2018), a VAE is used to find both a low-dimensional latent representation of images and a transition model between consecutive image pairs. Extensions to the VAE framework for longer sequences are investigated in (Gregor et al., 2015) and (Chung et al., 2015) with a focus on recurrent neural network architectures. Other lines of work combine structured and interpretable probabilistic graphical models with the flexibility of neural networks. In (Haarnoja et al., 2016), an extended Kalman filter is employed in combination with a convolutional neural network to output feature measurements and associated uncertainties. The Deep Markov Model, introduced in (Krishnan et al., 2017), parametrizes the transition and measurement functions of a SSM with neural networks. Deep variational Bayes filters (DVBF) (Karl et al., 2017) reparametrize the variational inference problem to enable a recognition network to output transition parameters. In this way, reconstruction errors can backpropagate through the transitions directly. Other research (Johnson et al., 2016; Fraccaro et al., 2017b) combines fast and exact inference subroutines with probabilistic graphical models and deep learning components.

Learning SSMs from rich, high-dimensional data such as images has also been investigated in the context of model-based reinforcement learning. Such models have been used for both online planning (Zhang et al., 2019b; Hafner et al., 2019) and for generating synthetic trajectories for model-free policy learning (Ha and Schmidhuber, 2018; Sutton et al., 2008). These approaches emphasize learning forward transitions through a combination of reconstruction and additional auxiliary losses (e.g., reward prediction).

We build on learning methods for sequential data that take advantage of the structure of a SSM (Karl et al., 2017; Fraccaro et al., 2017b; Chiappa and Piquet, 2019; Johnson et al., 2016). However, our work differs from and extends these approaches in the following ways: 1) we combine ideas from the novelty detection literature to enhance robustness to input degradations not seen during training, by

making use of the model's generative capabilities and probabilistic representation, and 2) we provide experimental results from a real-world robotic prediction and control task, demonstrating that our approach can be successfully applied to physical systems.

3.2.2 Novelty Detection

Our work is related to the field of novelty detection (Bishop, 1994b) and uncertainty quantification with Bayesian neural networks (MacKay, 1995). We demonstrate how novelty detection can be integrated into learned SSMs. Both of these fields address the general problem of evaluating the reliability of neural network outputs.

Existing Bayesian methods use Monte Carlo dropout (Gal and Ghahramani, 2016) or ensembles (Lakshminarayanan et al., 2017) to capture the variance within the training data. Bayesian methods have not yet been shown to be able to express accurate variances for regions far from the training distribution, however. Additionally, these networks often require a significant number of evaluations, which may compromise runtime efficiency. Taking into account these limitations, we aim to improve the reliability of network outputs by leveraging techniques from the field of novelty detection. Relevant prior work includes (Richter and Roy, 2017), where an autoencoder forms part of an image-based collision prediction system for a ground vehicle. When the autoencoder reconstruction loss is large, the input images are considered to be OOD and the system reverts to a more conservative (safer) collision prediction algorithm. In (Pomerleau, 1993) and (Amini et al., 2018), similar reconstruction-based approaches to novelty detection are presented for end-to-end learning of a control policy for autonomous driving. We also use a reconstruction-based approach to novelty detection but adapt it to the context of learned SSMs.

3.2.3 Multimodal Machine Learning

In this chapter, we also present an approach to learn SSMs given data from various sensing modalities. Work in the machine learning field has investigated the problem of learning representations and patterns of multimodal data for a variety downstream tasks. A good summary of the existing literature, focused on applications involving multimedia data (e.g., video, text, and audio), is provided in (Baltrusaitis et al., 2019). Probabilistic methods have also been applied to model the joint and conditional distributions of non-sequential multimodal data. Examples of these methods include the Restricted Boltzman Machine (RBM) (Ngiam et al., 2011; Srivastava and Salakhutdinov, 2014) and the VAE (Wu and Goodman, 2018; Suzuki et al., 2016; Vedantam et al., 2018). Our work is most similar to the latter of these two approaches. We build upon the MVAE (Wu and Goodman, 2018), but, critically, we apply and extend the framework to the sequential setting so that it is amenable to capturing a SSM of multimodal data.

3.2.4 Multimodal Learning for Robotics

Availability of multimodal data is a common occurrence in robotics (Limoyo et al., 2018). Our approach to extend learned SSMs to multimodal data is most similar in nature to (Gandhi et al., 2020), where the authors use audio data to augment a deterministic, state-based forward model. However, in (Gandhi et al., 2020), the audio data are taken from a previous random interaction and do not provide causally related information to the forward model—the audio data is simply used to augment the representation. In contrast, we directly learn a SSM based on the observed multimodal data. We also do not assume a relaxed, deterministic state-based setting and instead learn a probabilistic representation from raw multimodal information directly (as opposed to dealing with difficult-to-acquire state labels).

Similar to the approach presented in this chapter, other groups have investigated the use of learned differentiable filters with multimodal measurement models of visual, proprioceptive, and haptic data, relying on ground-truth annotations (Lee et al., 2020). However, in many cases ground-truth labels are expensive or impossible to acquire, which may hinder the scalability of such methods. We leverage recent work in variational inference and devise a self-supervised generative approach to bypass this limitation. We do so by maximizing a proper lower bound of the marginal likelihood of the data itself.

Other works have also leveraged the MVAE architecture for learned localization with multimodal data (Zhou et al., 2021b). Closer to our work, the authors of (Rezaei-Shoshtari et al., 2021) demonstrate a technique to learn a notion of "intuitive physics" in a self-supervised manner by applying the MVAE architecture as a generative model of multimodal sensor measurements. Specifically, future sensor measurements resulting from interaction with objects are decoded based on an encoding of the current sensor measurements. In our work, as opposed to directly decoding future transitions, we learn a SSM based on the compressed latent space; we therefore have the choice to predict while remaining in a low-dimensional space and without having to decode, which saves a significant amount of computation and memory.

3.3 Sequential Latent Variable Model

The task of identifying (i.e., learning) a SSM from observations can be formulated as one of learning a lower-dimensional latent state representation $\{\mathbf{z}_t\}_{t=1}^T$, $\mathbf{z}_k \in \mathbb{R}^n$ and the respective dynamics or state transition function $\mathbf{z}_{t+1} \sim p(\mathbf{z}_{t+1} | \mathbf{z}_t, \mathbf{u}_t)$, with control input or action $\mathbf{u}_t \in \mathbb{R}^m$. We consider a sequence of observations $\mathbf{x}_{1:T} = \{\mathbf{x}_t\}_{t=1}^T$ with respective control inputs $\mathbf{u}_{1:T} = \{\mathbf{u}_t\}_{t=1}^T$. We then introduce latent variables $\mathbf{z}_{1:T} = \{\mathbf{z}_t\}_{t=1}^T$ to create a joint distribution $p_\theta(\mathbf{x}_{1:T}, \mathbf{z}_{1:T} \mid \mathbf{u}_{1:T})$, where θ are the learnable parameters of our distributions, which can be parametrized by neural networks. We factor the joint distribution of the generative process as

$$p_\theta(\mathbf{x}_{1:T}, \mathbf{z}_{1:T} \mid \mathbf{u}_{1:T}) = p_\theta(\mathbf{x}_{1:T} \mid \mathbf{z}_{1:T}, \mathbf{u}_{1:T})\, p_\theta(\mathbf{z}_{1:T} \mid \mathbf{u}_{1:T}), \tag{3.1}$$

where

$$p_\theta(\mathbf{z}_{1:T} \mid \mathbf{u}_{1:T}) = p(\mathbf{z}_1) \prod_{t=2}^{T} p_\theta(\mathbf{z}_t \mid \mathbf{z}_{t-1}, \mathbf{u}_{t-1}), \tag{3.2}$$

and

$$p_\theta(\mathbf{x}_{1:T} \mid \mathbf{z}_{1:T}, \mathbf{u}_{1:T}) = \prod_{t=1}^{T} p_\theta(\mathbf{x}_t \mid \mathbf{z}_t). \tag{3.3}$$

The latent forward or motion model is the distribution $p_\theta(\mathbf{z}_t \mid \mathbf{z}_{t-1}, \mathbf{u}_{t-1})$ and the observation or measurement model is $p_\theta(\mathbf{x}_t \mid \mathbf{z}_t)$. The distribution $p(\mathbf{z}_1)$ is an arbitrary initial distribution with high uncertainty. The goal of learning is to maximize the marginal likelihood of the data or the evidence, given for a single sequence by

$$p(\mathbf{x}_{1:T} \mid \mathbf{u}_{1:T}) = \int p_\theta(\mathbf{x}_{1:T} \mid \mathbf{z}_{1:T}, \mathbf{u}_{1:T})\, p_\theta(\mathbf{z}_{1:T} \mid \mathbf{u}_{1:T})\, d\mathbf{z}_{1:T} \tag{3.4}$$

with respect to the parameters θ. Unfortunately, in the general case with this model, the posterior distribution used for inference, $p(\mathbf{z}_{1:T} \mid \mathbf{x}_{1:T}, \mathbf{u}_{1:T})$, is intractable. A common solution from recent work in variational inference (Kingma and Welling, 2014; Rezende et al., 2014b) is to introduce a recognition or inference model $q_\phi(\mathbf{z}_{1:T} \mid \mathbf{x}_{1:T}, \mathbf{u}_{1:T})$ with parameters ϕ to approximate the intractable posterior. This leads to the following lower bound on the marginal log-likelihood or the evidence lower bound (ELBO)

$$\begin{aligned} \log p(\mathbf{x}_{1:T} \mid \mathbf{u}_{1:T}) \geq \mathbb{E}_{q_\phi(\mathbf{z}_{1:T} \mid \mathbf{x}_{1:T}, \mathbf{u}_{1:T})} \left[\log p_\theta(\mathbf{x}_{1:T} \mid \mathbf{z}_{1:T}, \mathbf{u}_{1:T})\right] \\ - KL(q_\phi(\mathbf{z}_{1:T} \mid \mathbf{x}_{1:T}, \mathbf{u}_{1:T}) \| p_\theta(\mathbf{z}_{1:T} \mid \mathbf{u}_{1:T})). \end{aligned} \tag{3.5}$$

Maximizing this lower bound can be shown to be equivalent to minimizing the KL divergence between the true posterior $p_\theta(\mathbf{z}_{1:T} \mid \mathbf{x}_{1:T}, \mathbf{u}_{1:T})$ and the recognition model $q_\phi(\mathbf{z}_{1:T} \mid \mathbf{x}_{1:T}, \mathbf{u}_{1:T})$. The resulting optimization objective, denoted by Eq. (3.5), is based on an expectation with respect to the distribution $q_\phi(\mathbf{z}_{1:T} \mid \mathbf{x}_{1:T}, \mathbf{u}_{1:T})$, which itself is based on the parameters ϕ. As is typically done, we restrict $q_\phi(\mathbf{z}_{1:T} \mid \mathbf{x}_{1:T}, \mathbf{u}_{1:T})$ to be a Gaussian variational approximation. This enables us to use stochastic gradient descent (i.e., using Monte Carlo estimates of the gradient) via the reparameterization trick (Kingma and Welling, 2014) to optimize the lower bound. The specific choice for the factorization of $q_\phi(\mathbf{z} \mid \mathbf{x}, \mathbf{u})$ varies depending on the application (i.e., prediction, smoothing, or filtering). Given our intended applications of prediction, in this chapter, we choose to only use causal (i.e., current and past) information for inference,

$$q_\phi(\mathbf{z}_{1:T} \mid \mathbf{x}_{1:T}, \mathbf{u}_{1:T}) = \prod_{t=1}^{T} q_\phi(\mathbf{z}_t \mid \mathbf{x}_{\leq t}, \mathbf{u}_{<t}). \tag{3.6}$$

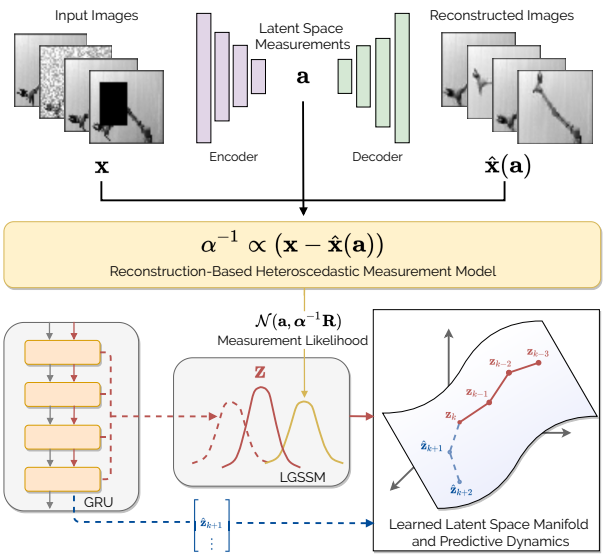


Figure 3.2: We learn a generative latent dynamics model that is robust to novel or out-of-distribution data at test time. To do so, we leverage the structure of a linear Gaussian state space model (LGSSM) with explicit uncertainty terms (e.g., the measurement uncertainty, $\mathbf{R}$), which can be set on a per-input basis depending on the approximated novelty of the data.

3.4 Structured Sequential Latent Variable Model

We take inspiration from recent papers that show how the structure of a linear Gaussian SSM (LGSSM) can be composed with one or more neural networks (Karl et al., 2017; Fraccaro et al., 2017b; Chiappa and Piquet, 2019; Johnson et al., 2016). By doing so, we have a mechanism at test time to mitigate the detrimental effects of observations that are far from the training set distribution. We do this by inflating the latent measurement uncertainty based on the estimated novelty of each observation.

In this section, we first demonstrate how the structure of the LGSSM can be used within the latent state space model framework (Fig. 3.2). We define our transition model as follows,

$$p(\mathbf{z}_t \mid \mathbf{z}_{t-1}, \mathbf{u}_{t-1}) = \mathcal{N}(\mathbf{A}_t \mathbf{z}_{t-1} + \mathbf{B}_t \mathbf{u}_{t-1}, \mathbf{Q}), \tag{3.7}$$

with local transition matrix $\mathbf{A}_t \in \mathbb{R}^{n \times n}$, local control matrix $\mathbf{B}_t \in \mathbb{R}^{n \times m}$, and constant process noise covariance $\mathbf{Q} \in \mathbb{R}^{n \times n}$. The accompanying measurement model is

$$p(\mathbf{a}_t \mid \mathbf{z}_t) = \mathcal{N}(\mathbf{C}_t \mathbf{z}_t, \mathbf{R}), \tag{3.8}$$

where measurement $\mathbf{a}_t \in \mathbb{R}^l \sim q(\mathbf{a}_t \mid \mathbf{x}_t)$ is extracted from image $\mathbf{x}_t$, $\mathbf{C}_t \in \mathbb{R}^{l \times n}$ is the local measurement matrix, and $\mathbf{R} \in \mathbb{R}^{l \times l}$ is the measurement noise covariance. We choose to add the structure of the LGSSM to be able to explicitly reason about uncertainties in an interpretable manner. We explicitly disentangle the recognition network q by having it operate on the measurements or observations, following the work of (Fraccaro et al., 2017b), in order to reduce the dimensions of both (latent) measurement $\mathbf{a}$ and state $\mathbf{z}$ for the LGSSM. The joint probability distribution of a sequence of latent

measurements $\mathbf{a}_{1:T} = \{\mathbf{a}_t\}_{t=1}^T$ and states $\mathbf{z}_{1:T} = \{\mathbf{z}_t\}_{t=1}^T$ can then be defined as

$$\begin{aligned} p(\mathbf{a}_{1:T}, \mathbf{z}_{1:T} \mid \mathbf{u}_{1:T}) &= p(\mathbf{a}_{1:T} \mid \mathbf{z}_{1:T})\, p(\mathbf{z}_{1:T} \mid \mathbf{u}_{1:T}) \\ &= p(\mathbf{z}_1) \prod_{t=1}^{T} p(\mathbf{a}_t \mid \mathbf{z}_t) \prod_{t=2}^{T} p(\mathbf{z}_t \mid \mathbf{z}_{t-1}, \mathbf{u}_{t-1}), \end{aligned} \tag{3.9}$$

where $\mathbf{z}_1 \in \mathbb{R}^n \sim p(\mathbf{z}_1)$ is the initial state from a given fixed distribution $p(\mathbf{z}_1) = \mathcal{N}(\mathbf{0}, \mathbf{\Sigma}_1)$. Note that, as a byproduct of employing a LGSSM, exact inference can be carried out for the posterior $p(\mathbf{z} \mid \mathbf{a}, \mathbf{u})$ using the Kalman filter (KF) (Kalman, 1960) or Rauch-Tung-Striebel (RTS) smoothing equations (Rauch et al., 1965).

We pose the problem of approximating the intractable posterior of the latent measurements given the images, $p(\mathbf{a}_{1:T} \mid \mathbf{x}_{1:T})$ in terms of variational inference. As is commonly done in the variational inference literature, we use a recognition network to output the mean and standard deviation of the (assumed) Gaussian posterior,

$$q_\phi(\mathbf{a}_{1:T} \mid \mathbf{x}_{1:T}) = \prod_{t=1}^{T} q_\phi(\mathbf{a}_t | \mathbf{x}_t). \tag{3.10}$$

In this work, our network is comprised of a CNN followed by a fully connected layer, based on the architecture in (Ha and Schmidhuber, 2018). We use transposed convolution layers in the generative decoder to output the parameters of the assumed Gaussian or Bernoulli conditional distribution,

$$p_\theta(\mathbf{x}_{1:T} \mid \mathbf{a}_{1:T}) = \prod_{t=1}^{T} p_\theta(\mathbf{x}_t \mid \mathbf{a}_t). \tag{3.11}$$

A gated recurrent unit (GRU) network (Cho et al., 2014) is trained to regress the matrices $\mathbf{A}_k, \mathbf{B}_k, \mathbf{C}_k$,

$$\mathbf{A}_t, \mathbf{B}_t, \mathbf{C}_t, \mathbf{h}_{t+1} = g_\psi(\mathbf{z}_{t-1}, \mathbf{u}_{t-1}, \mathbf{h}_t), \tag{3.12}$$

where $\mathbf{h}_t \in \mathbb{R}^v$ is the recurrent hidden state at time step t. The GRU network outputs a linear combination of base matrices that are independent of the input, and are learned globally (see (Karl et al., 2017; Fraccaro et al., 2017b) for a more detailed exposition). In order to learn or identify the parameters $\{\phi, \theta, \psi\}$, we use the reparameterization trick (Kingma and Welling, 2014; Rezende et al., 2014b) to maximize a lower bound (Jordan et al., 1999b) of the marginal likelihood of the data, $p(\mathbf{x}_{1:T} \mid \mathbf{u}_{1:T}) = \int p(\mathbf{x}_{1:T}, \mathbf{a}_{1:T}, \mathbf{z}_{1:T} \mid \mathbf{u}_{1:T})\, d\mathbf{a}_{1:T}\, d\mathbf{z}_{1:T}$. We first consider the log-likehood of the data

$$\log p(\mathbf{x}_{1:T} \mid \mathbf{u}_{1:T}) = \log \int p(\mathbf{x}_{1:T}, \mathbf{a}_{1:T}, \mathbf{z}_{1:T} \mid \mathbf{u}_{1:T})\, d\mathbf{a}_{1:T}\, d\mathbf{z}_{1:T}, \tag{3.13}$$

and derive a lower bound using Jensen's inequality

$$\log p(\mathbf{x}_{1:T} \mid \mathbf{u}_{1:T}) \geq \int \log p(\mathbf{x}_{1:T}, \mathbf{a}_{1:T}, \mathbf{z}_{1:T} \mid \mathbf{u}_{1:T})\, d\mathbf{a}_{1:T}\, d\mathbf{z}_{1:T}. \tag{3.14}$$

Next, we introduce the recognition network or variational distribution $q_\phi(\mathbf{a}_{1:T}, \mathbf{z}_{1:T} \mid \mathbf{x}_{1:T}, \mathbf{u}_{1:T})$,

$$\begin{aligned}\log p(\mathbf{x}_{1:T} \mid \mathbf{u}_{1:T}) &\geq \int \frac{p(\mathbf{x}_{1:T}, \mathbf{a}_{1:T}, \mathbf{z}_{1:T} \mid \mathbf{u}_{1:T}) q(\mathbf{a}_{1:T}, \mathbf{z}_{1:T} \mid \mathbf{x}_{1:T}, \mathbf{u}_{1:T})}{q(\mathbf{a}_{1:T}, \mathbf{z}_{1:T} \mid \mathbf{x}_{1:T}, \mathbf{u}_{1:T})} d\mathbf{a}_{1:T}\, d\mathbf{z}_{1:T} \\ &= \mathbb{E}_{q(\mathbf{a}_{1:T}, \mathbf{z}_{1:T} \mid \mathbf{x}_{1:T}, \mathbf{u}_{1:T})} \left[\frac{p(\mathbf{x}_{1:T}, \mathbf{a}_{1:T}, \mathbf{z}_{1:T} \mid \mathbf{u}_{1:T})}{q(\mathbf{a}_{1:T}, \mathbf{z}_{1:T} \mid \mathbf{x}_{1:T}, \mathbf{u}_{1:T})} \right] \\ &= \mathbb{E}_{q(\mathbf{a}_{1:T}, \mathbf{z}_{1:T} \mid \mathbf{x}_{1:T}, \mathbf{u}_{1:T})} \left[\frac{p_\theta(\mathbf{x}_{1:T} \mid \mathbf{a}_{1:T}) p_\psi(\mathbf{a}_{1:T} \mid \mathbf{z}_{1:T}) p_\psi(\mathbf{z}_{1:T} \mid \mathbf{u}_{1:T})}{q(\mathbf{a}_{1:T}, \mathbf{z}_{1:T} \mid \mathbf{x}_{1:T}, \mathbf{u}_{1:T})} \right].\end{aligned} \tag{3.15}$$

Exploiting the LGSSM structure, we can factorize the variational distribution as

$$q(\mathbf{a}_{1:T}, \mathbf{z}_{1:T} \mid \mathbf{x}_{1:T}, \mathbf{u}_{1:T}) = q_\phi(\mathbf{a}_{1:T} \mid \mathbf{x}_{1:T}) p_\psi(\mathbf{z}_{1:T} \mid \mathbf{a}_{1:T}, \mathbf{u}_{1:T}), \tag{3.16}$$

where, crucially, we can compute $p_\psi(\mathbf{z}_{1:T} \mid \mathbf{a}_{1:T}, \mathbf{u}_{1:T})$ analytically using the RTS smoother algorithm (Murphy, 2012b). It follows that we only need to approximate $q_\phi(\mathbf{a}_{1:T} \mid \mathbf{x}_{1:T})$ with the VAE inference network. By replacing the factorized variational distribution shown in Eq. (3.16) into Eq. (3.15) and rearranging the terms we arrive at

$$\begin{aligned}\log p(\mathbf{x}_{1:T} \mid \mathbf{u}_{1:T}) \geq \mathbb{E}_{q_\phi(\mathbf{a}_{1:T} \mid \mathbf{x}_{1:T})} \Bigg[&\log \frac{p_\theta(\mathbf{x}_{1:T} \mid \mathbf{a}_{1:T})}{q_\phi(\mathbf{a}_{1:T} \mid \mathbf{x}_{1:T})} \\ &+ \mathbb{E}_{p_\psi(\mathbf{z}_{1:T} \mid \mathbf{a}_{1:T}, \mathbf{u}_{1:T})} \left[\log \frac{p_\psi(\mathbf{a}_{1:T} \mid \mathbf{z}_{1:T}) p_\psi(\mathbf{z}_{1:T} \mid \mathbf{u}_{1:T})}{p_\psi(\mathbf{z}_{1:T} \mid \mathbf{a}_{1:T}, \mathbf{u}_{1:T})} \right] \Bigg].\end{aligned} \tag{3.17}$$

Our cost function is the negative of the lower bound of the marginal log-likelihood of the data, as shown by Eq. (3.17). We estimate the intractable expectations in the lower bound by sampling. We then use stochastic gradient descent to update the parameters $\{\phi, \theta, \psi\}$ of the respective encoder, decoder, and GRU networks. Given a sequence of input images of length I from our training set, $\mathbf{x}_{1:I} = \{\mathbf{x}_i\}_{i=1}^I$, we first sample $\tilde{\mathbf{a}}_{1:I} \sim q_\phi(\mathbf{a}_{1:I} \mid \mathbf{x}_{1:I})$ with the distribution parameters of $q_\phi(\mathbf{a}_{1:I} \mid \mathbf{x}_{1:I})$ coming from the recognition network. Based on $\tilde{\mathbf{a}}$, we can analytically obtain the parameters of $p_\psi(\mathbf{z}_{1:I} \mid \mathbf{a}_{1:I}, \mathbf{u}_{1:I})$ from the the RTS smoothing equations. To do so, we require a known initial distribution $p(\mathbf{z}_1)$, the locally linear matrices $\mathbf{A}_{1:I}, \mathbf{B}_{1:I}, \mathbf{C}_{1:I}$ from the GRU network, and known control inputs $\mathbf{u}_{1:I} = \{\mathbf{u}_i\}_{i=1}^I$. Next, we sample $\tilde{\mathbf{z}}_{1:I} \sim p_\psi(\mathbf{z}_{1:I} \mid \mathbf{a}_{1:I}, \mathbf{u}_{1:I})$ and obtain the parameters of $p_\psi(\mathbf{z}_{1:T} \mid \mathbf{u}_{1:T})$ and $p_\psi(\mathbf{a}_{1:T} \mid \mathbf{z}_{1:T})$, using Eq. (3.7) and Eq. (3.8), respectfully. Lastly, $p_\theta(\mathbf{x}_{1:I} \mid \mathbf{a}_{1:I})$ is assumed to be a Gaussian or Bernoulli distribution with parameters coming from the decoder network, as is commonly done with VAEs.

3.5 Robustness via Novelty Detection

We can set a per time step uncertainty $\mathbf{R}_t$, instead of a constant uncertainty $\mathbf{R}$ as shown in Eq. (3.8). Crucially, in Eq. (3.12), we explicitly learn the parameters $\mathbf{A}_t, \mathbf{B}_t, \mathbf{C}_t$ based on past latent states $\{\mathbf{z}_i\}_{t=1}^t$,

as opposed to past measurements (features) $\{\mathbf{a}_i\}_{t=1}^{t}$ as done in (Fraccaro et al., 2017b) and (Chiappa and Piquet, 2019). We do this to decouple the dynamics from the measurements. If required, this allows our model to prioritize the dynamics and to reduce the influence of the measurement $\mathbf{a}_t$ on the state $\mathbf{z}_t$ during inference by inflating the respective uncertainty $\mathbf{R}_t$. We note that $\mathbf{R}_t$ can be constrained to be diagonal without loss of generality when our latent states have no predefined meaning (Murphy, 2012b), as in this case. By doing so, we reduce the number of free parameters and improve numerical stability.

To account for OOD data, we raise our density to a power $\alpha_t(\mathbf{a}_t, \mathbf{x}_t) \in \mathbb{R}^+$,

$$p^{\alpha_t}(\mathbf{a}_t|\,\mathbf{z}_t). \tag{3.18}$$

Raising a density to a power is proposed as a technique in (Cao and Fleet., 2014) to modify a product of experts such that each expert's reliability can be scaled based on the input datapoint. Raising a density to a power has also been used in Markov chain Monte Carlo (MCMC) algorithms to anneal distributions and to balance probabilistic models with varying degrees of freedom (Urtasun et al., 2006). In our case, where the original distribution is a Gaussian, the resulting distribution remains Gaussian and the operation can be shown to be equivalent to scaling the precision matrix (or the measurement covariance),

$$p^{\alpha_t}(\mathbf{a}_t \mid \mathbf{z}_t) = \frac{1}{Z}\exp\left(-\frac{1}{2}\mathbf{r}_t^T\alpha_t\mathbf{R}^{-1}\mathbf{r}_t\right), \tag{3.19}$$

where $\mathbf{r}_t = (\mathbf{a}_t - \mathbf{C}_t\mathbf{z}_t)$ is the innovation or residual and $\mathbf{R}_t^{-1} = \alpha_t\mathbf{R}^{-1}$ is the weighted precision matrix. Based on our network's generative capabilities, we use the following reconstruction measure as a proxy for the network's confidence or reliability,

$$\alpha_t(\mathbf{a}_t, \mathbf{x}_t) = \log\left(1 + \frac{\bar{L}_{\text{train}}}{L_t}\right), \tag{3.20}$$

where $\bar{L}_{\text{train}} = \frac{1}{N}\sum_{n=1}^{N}\|\mathbf{x}_n - \hat{\mathbf{x}}_n(\mathbf{a}_n)\|_F^2$ is the average reconstruction loss over the N images in our training set and $L_t = \|\mathbf{x}_t - \hat{\mathbf{x}}_t(\mathbf{a}_t)\|_F^2$ is the reconstruction loss for image $\mathbf{x}_t$. We note that the reconstruction loss is equivalent up to a constant to the reconstruction negative log likelihood. The reconstructed images $\hat{\mathbf{x}}$ are produced by the generative decoder defined by Eq. (3.11); an image with a larger reconstruction error produces a smaller measurement precision (or a larger measurement covariance). Similar approaches using reconstruction or likelihood-based novelty detection have been used in (Pomerleau, 1993), (Richter and Roy, 2017) and (Amini et al., 2018).

3.6 Multimodal Sequential Latent Variable Model

We can also extend latent SSMs to learn directly from multimodal observations. We consider N sequences of separate observations or modalities, $\mathbf{X}_{1:N} = \{\mathbf{x}_{1:T}^n\}_{n=1}^{N}$, where we assume that each sequence is of equivalent length T: $\mathbf{x}_{1:T}^n = \{\mathbf{x}_t^n\}_{t=1}^{T}$. As done in the previous section, we include the respective control inputs $\mathbf{u}_{1:T} = \{\mathbf{u}_t\}_{t=1}^{T}$ and again introduce a set of latent variables $\mathbf{z}_{1:T} = \{\mathbf{z}_t\}_{t=1}^{T}$

as a lower-dimensional latent space containing some underlying dynamics of interest. The final joint distribution factorizes as $p_\theta(\mathbf{X}_{1:N}, \mathbf{z}_{1:T} \mid \mathbf{u}_{1:T}) = p_\theta(\mathbf{X}_{1:N} \mid \mathbf{z}_{1:T}, \mathbf{u}_{1:T})\, p_\theta(\mathbf{z}_{1:T} \mid \mathbf{u}_{1:T})$. We choose to define the generative process as

$$\begin{aligned} p_\theta(\mathbf{X}_{1:N} \mid \mathbf{z}_{1:T}, \mathbf{u}_{1:T}) &= \prod_{n=1}^{N} p_\theta(\mathbf{x}_{1:T}^n \mid \mathbf{z}_{1:T}, \mathbf{u}_{1:T}) \\ &= \prod_{n=1}^{N} \prod_{t=1}^{T} p_\theta(\mathbf{x}_t^n \mid \mathbf{z}_t), \end{aligned} \tag{3.21}$$

and with $p_\theta(\mathbf{z}_{1:T} \mid \mathbf{u}_{1:T})$ remaining the same as shown in Eq. (3.7). The marginal likelihood is then

$$p_\theta(\mathbf{X}_{1:N} \mid \mathbf{u}_{1:T}) = \int p_\theta(\mathbf{X} \mid \mathbf{z}_{1:T}, \mathbf{u}_{1:T})\, p_\theta(\mathbf{z}_{1:T} \mid \mathbf{u}_{1:T}) \mathrm{d}\mathbf{z}_{1:T}, \tag{3.22}$$

and the respective ELBO, for a single sequence, is then:

$$\begin{aligned} \log p_\theta(\mathbf{X}_{1:N} \mid \mathbf{u}_{1:T}) \geq \mathbb{E}_{q_\phi(\mathbf{z}_{1:T} \mid \mathbf{X}_{1:N}, \mathbf{u}_{1:T})} \left[\sum_{\mathbf{x}^n \in \mathbf{X}_{1:N}} \log p_\theta(\mathbf{x}_{1:T}^n \mid \mathbf{z}_{1:T}, \mathbf{u}_{1:T}) \right] \\ - \mathrm{KL}(q_\phi(\mathbf{z}_{1:T} \mid \mathbf{X}_{1:N}, \mathbf{u}_{1:T}) \mid p_\theta(\mathbf{z}_{1:T} \mid \mathbf{u}_{1:T})). \end{aligned} \tag{3.23}$$

In order to decide on the factorization of $q_\phi(\mathbf{z}_{1:T} \mid \mathbf{X}_{1:N}, \mathbf{u}_{1:T})$, we draw inspiration from the MVAE architecture (Wu and Goodman, 2018), and base our inference network on the structure of the true multimodal posterior $p(\mathbf{z}_{1:T} \mid \mathbf{X}_{1:N}, \mathbf{u}_{1:T})$:

$$\begin{aligned} p(\mathbf{z}_{1:T} \mid \mathbf{X}_{1:N}, \mathbf{u}_{1:T}) &= \frac{p(\mathbf{z}_{1:T} \mid \mathbf{u}_{1:T})\, p(\mathbf{X}_{1:N} \mid \mathbf{z}_{1:T}, \mathbf{u}_{1:T})}{p(\mathbf{X}_{1:N} \mid \mathbf{u}_{1:T})} \\ &= \frac{p(\mathbf{z}_{1:T} \mid \mathbf{u}_{1:T})}{p(\mathbf{x}_{1:T}^1, ..., \mathbf{x}_{1:T}^N \mid \mathbf{u}_{1:T})} \prod_{n=1}^{N} p(\mathbf{x}_{1:T}^n \mid \mathbf{z}_{1:T}, \mathbf{u}_{1:T}) \\ &= \frac{p(\mathbf{z}_{1:T} \mid \mathbf{u}_{1:T})}{p(\mathbf{x}_{1:T}^1, ..., \mathbf{x}_{1:T}^N \mid \mathbf{u}_{1:T})} \prod_{n=1}^{N} \frac{p(\mathbf{z}_{1:T} \mid \mathbf{x}_{1:T}^n, \mathbf{u}_{1:T}) p(\mathbf{x}_{1:T}^n \mid \mathbf{u}_{1:T})}{p(\mathbf{z}_{1:T} \mid \mathbf{u}_{1:T})} \\ &= \frac{\prod_{n=1}^{N} p(\mathbf{x}^n \mid \mathbf{u})}{p(\mathbf{x}^1, ..., \mathbf{x}^N \mid \mathbf{u}_{1:T})} \cdot \frac{\prod_{n=1}^{N} p(\mathbf{z}_{1:T} \mid \mathbf{x}_{1:T}^n, \mathbf{u}_{1:T})}{\prod_{n=1}^{N-1} p(\mathbf{z}_{1:T} \mid \mathbf{u}_{1:T})} \\ &\propto \frac{\prod_{n=1}^{N} p(\mathbf{z}_{1:T} \mid \mathbf{x}_{1:T}^n, \mathbf{u}_{1:T})}{\prod_{n=1}^{N-1} p(\mathbf{z}_{1:T} \mid \mathbf{u}_{1:T})}. \end{aligned} \tag{3.24}$$

Based on the last term in Eq. (3.24), the final factorization of the joint posterior is then a quotient between a product of the individual modality-specific posteriors and the prior. Accordingly, we choose

our inference model to be

$$q(\mathbf{z}_{1:T} \mid \mathbf{X}_{1:N}, \mathbf{u}_{1:T}) = \frac{\prod_{n=1}^{N} q(\mathbf{z}_{1:T} \mid \mathbf{x}_{1:T}^{n}, \mathbf{u}_{1:T})}{\prod_{n=1}^{N-1} p(\mathbf{z}_{1:T} \mid \mathbf{u}_{1:T})}. \tag{3.25}$$

We also use the same representation reformulation trick from the MVAE (Wu and Goodman, 2018) and set $q(\mathbf{z}_{1:T} \mid \mathbf{x}_{1:T}^{n}, \mathbf{u}_{1:T}) = \tilde{q}(\mathbf{z}_{1:T} \mid \mathbf{x}_{1:T}^{n}, \mathbf{u}_{1:T})\, p(\mathbf{z}_{1:T} \mid \mathbf{u}_{1:T})$ in order to produce a simpler and numerically more stable product of experts,

$$q_\phi(\mathbf{z}_{1:T} \mid \mathbf{X}_{1:N}, \mathbf{u}_{1:T}) = p_\theta(\mathbf{z}_{1:T} \mid \mathbf{u}_{1:T}) \prod_{n=1}^{N} \tilde{q}_\phi(\mathbf{z}_{1:T} \mid \mathbf{x}_{1:T}^{n}, \mathbf{u}_{1:T}), \tag{3.26}$$

where $\tilde{q}_\phi(\mathbf{z}_{1:T} \mid \mathbf{x}_{1:T}^{n}, \mathbf{u}_{1:T})$ would be equivalent to the standard single-modality posterior shown in Eq. (3.6). Finally, we can further factorize the posterior in Eq. (3.26) into a more intuitive form for our sequential setting:

$$\begin{aligned} q_\phi(\mathbf{z}_{1:T} \mid \mathbf{X}_{1:N}, \mathbf{u}_{1:T}) &= \underbrace{p(\mathbf{z}_1) \prod_{n=1}^{N} \tilde{q}_\phi(\mathbf{z}_1 \mid \mathbf{x}_1^n)}_{q_\phi(\mathbf{z}_1 \mid \mathbf{x}_1^1, \ldots, \mathbf{x}_1^N)} \prod_{t=2}^{T} \underbrace{\left(p_\theta(\mathbf{z}_t \mid \mathbf{z}_{t-1}, \mathbf{u}_{t-1}) \prod_{n=1}^{N} \tilde{q}_\phi(\mathbf{z}_t \mid \mathbf{x}_{\leq t}^n, \mathbf{u}_{<t}) \right)}_{q_\phi(\mathbf{z}_t \mid \mathbf{x}_{\leq t}^1, \ldots, \mathbf{x}_{\leq t}^N, \mathbf{u}_{<t})} \\ &= \prod_{t=1}^{T} q_\phi(\mathbf{z}_t \mid \mathbf{x}_{\leq t}^1, \ldots, \mathbf{x}_{\leq t}^N, \mathbf{u}_{<t}). \end{aligned} \tag{3.27}$$

Our factorization reveals that, at every time step, each data modality is first separately encoded into a Gaussian distribution by its own inference model. A product is then taken of each modality-specific distribution and the prior, which is also the transition distribution of our latent space. It is interesting that we recover a similar form to the commonly-used recurrent SSM (Hafner et al., 2019), where the transition distribution is included in the posterior. The product of distributions at each time step is not generally solvable in closed form. However, by assuming Gaussian distributions for the prior dynamics and each modality-specific inference distribution, we end up with a final product of Gaussians for which a closed-form analytical solution does exist. Conveniently, a product of Gaussians is also itself a Gaussian (Cao and Fleet., 2014).

3.7 Model Predictive Control in Latent Space

Latent SSMs can be used for control with algorithms such as model predictive control (MPC). As an example, we consider the task of producing a specified goal image from a given initial image (i.e., mapping from pixels to pixels); we can use a history of multiple images if needed. We solve for the optimal controls over a prediction horizon of length T by way of MPC with the learned SSM. We note that any reward or cost based on the predicted latent states could be used (e.g., a learned reward or value function). We write the MPC formulation for the structured sequential latent space variant as an example, but a similar formulation can be used for the other models covered in this chapter.

We embed a window of T_i initial images $\{\mathbf{x}_t^i\}_{t=1}^{T_i}$ into measurements $\{\mathbf{a}_t^i\}_{t=1}^{T_i}$ using the encoder network Eq. (3.10). From both $\{\mathbf{x}_t^i\}_{t=1}^{T_i}$ and $\{\mathbf{a}_t^i\}_{t=1}^{T_i}$, we can calculate the heteroscedastic weighing factors $\{\alpha_t^i\}_{t=1}^{T_i}$ using Eq. (3.20) and rescale the measurement precision matrices $\{(\mathbf{R}_t^i)^{-1}\}_{t=1}^{T_i} = \{\alpha_t^i \mathbf{R}^{-1}\}_{t=1}^{T_i}$ accordingly. We can then solve for a window of initial latent states $\{\mathbf{z}_t^i\}_{t=1}^{T_i}$ given the control inputs $\{\mathbf{u}_t^i\}_{t=1}^{T_i}$ by exact inference with the RTS smoothing equations. In a similar manner, we can solve for a window of goal latent states $\{\mathbf{z}_t^g\}_{t=1}^{T_g}$ for a given window of T_g goal images $\{\mathbf{x}_t^g\}_{t=1}^{T_g}$ (potentially, a single image repeated T_g times) and control inputs $\{\mathbf{u}_t^g\}_{t=1}^{T_g} = \mathbf{0}$.

We define the optimization problem for the controls $\mathbf{u}_{1:T}^*$ over the next T time-steps as

$$\begin{aligned} \mathbf{u}_{1:T}^* = \arg\min_{\mathbf{u}_{1:T}} & \sum_{t=1}^{T} (\mathbf{z}_{t+1} - \mathbf{z}_{T_g}^g)^T \mathbf{Q}_{\text{mpc}} (\mathbf{z}_{t+1} - \mathbf{z}_{T_g}^g) + \mathbf{u}_t^T \mathbf{R}_{\text{mpc}} \mathbf{u}_t, \\ \text{s.t.} \quad & \mathbf{z}_t = \mathbf{A}_t \mathbf{z}_{t-1} + \mathbf{B}_t \mathbf{u}_{t-1}, \\ & \mathbf{z}_1 = \mathbf{z}_{T_i}^i, \end{aligned} \tag{3.28}$$

where $\mathbf{u}_{1:T} = \{\mathbf{u}_1, ..., \mathbf{u}_T\}$, $\mathbf{Q}_{\text{mpc}} \in \mathbb{R}^{n \times n}$ is positive semidefinite, $\mathbf{R}_{\text{mpc}} \in \mathbb{R}^{m \times m}$ is positive definite, and $\mathbf{z}_{T_i}^i$ is set as the initial latent state and $\mathbf{z}_{T_g}^g$ as the goal latent state. We generate the transition matrices $\mathbf{A}_{1:T}$ and control matrices $\mathbf{B}_{1:T}$ using the GRU network (Eq. (3.12)). We can then minimize Eq. (3.28) for $\mathbf{u}_{1:T}^*$ with any common convex optimization technique; we use the CVXPY modelling language (Diamond and Boyd, 2016). The updated solution is used to produce a new set of transition and control matrices, and iterated until convergence. Once converged, we send the first control input $\mathbf{u}_1^*$ to the system. We then observe the resulting image and use it to update our initial latent state. This process continues with the solution from the previous horizon as the new initial guess.

3.8 Network Architecture and Training

We parametrize our models with neural networks. For image data, we use a fully convolutional neural network based on the architecture of (Ha and Schmidhuber, 2018). The respective decoder is a matching deconvolutional network. For proprioceptive and force-torque data, we use a simple 1D convolutional architecture for the encoder and a 1D deconvolutional network for the decoder.

Our transition function or motion model, $p_\theta(\mathbf{z}_t \mid \mathbf{z}_{t-1}, \mathbf{u}_{t-1})$ is parameterized as a single-layer GRU network Cho et al. (2014) with 256 units that produces linear transition matrices (i.e., $\mathbf{z}_t = \mathbf{A}_t \mathbf{z}_{t-1} + \mathbf{B}_t \mathbf{u}_{t-1}$, where $\mathbf{A}_t$ and $\mathbf{B}_t$ are outputs of the network). During training, we sample mini-batches of 32 trajectories. We apply weight normalization (Salimans and Kingma, 2016) to all of the network layers except for the GRU. We use ReLU activation functions for all of the networks. Our latent space for all experiments consists of 16-dimensional Gaussians with diagonal covariance matrices. We apply the `Adam` optimizer (Kingma and Ba, 2015) with a learning rate of .0003 and a gradient clipping norm of 0.5 for the GRU network.

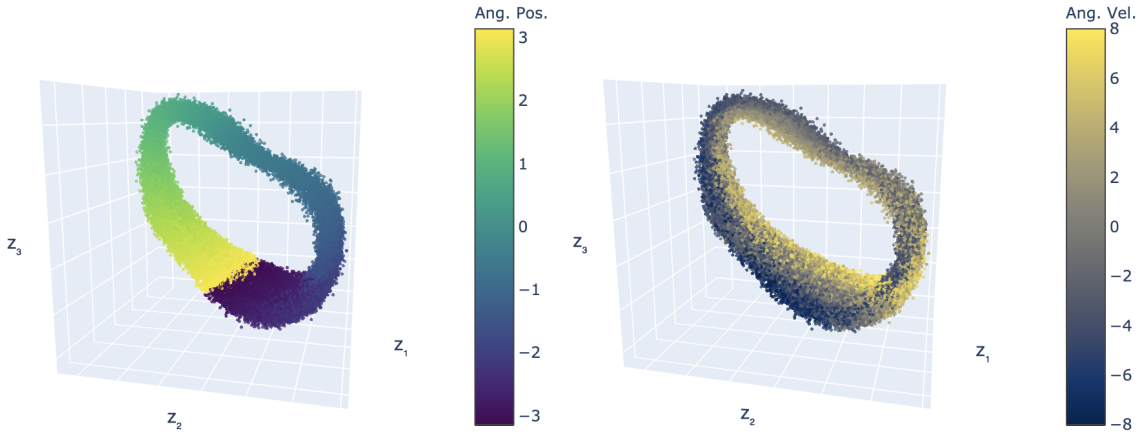


Figure 3.3: Smoothed, inferred latent space, $\mathbf{z}_k \in \mathbb{R}^3$, for the pendulum task. *Left:* Points are colour-coded by ground truth joint angle. *Right:* Points are colour-coded by angular velocity.

3.9 Experiments on Robustness of the Structured SLV Model

For our first set of experiments, we evaluated the robustness of the structured latent SSM (i.e., structured SLV model) variant (introduced in Section 3.4) using a combination of visualization, prediction and control experiments. We conducted experiments involving two different image-based control tasks: a modified simulated pendulum task from OpenAI Gym (Brockman et al., 2016) and a real-world manipulator 2D reaching task. In both cases, we tested the predictive and control capability of our model under challenging conditions using image data alone. The high-dimensional observations (i.e., images with thousands of pixels) and the highly nonlinear mapping from the underlying system dynamics to the image make these tasks non-trivial; naive application of classical optimal control would be infeasible. We chose a random, zero-mean Gaussian (i.e., random control input) exploration policy to collect training trajectories with a control or action repeat of three (Mnih et al., 2013).

3.9.1 Simulated Visual Pendulum Task

We began with a 'toy' environment as an initial test to allow us to directly visualize and analyze what our model learned. As our only modification to the existing OpenAI environment, we used grayscale images as inputs, $\mathbf{x} \in \mathbb{R}^{64\times64}$, instead of the pendulum joint angle and velocity; the control input $u \in \mathbb{R}$ (i.e., the input torque to the pendulum) remained unchanged. We chose a latent space $\mathbf{z} \in \mathbb{R}^3$ and a measurement space $\mathbf{a} \in \mathbb{R}^2$. During training, we collected a total of $N = 2048$ trajectories of length 32 to train our model.

Visualization Experiments

The trained model learned a latent state representation that resembles a ring-like manifold, as shown in Fig. 3.3. Each point in the latent space in Fig. 3.3 is colour-coded based on the ground truth angular position and velocity of the pendulum (on the left and right, respectively). Fig. 3.4 visualizes the measurements, $\mathbf{a}_k$, and their respective covariances, $\mathbf{R}_k$, for three different input image conditions:

Figure 3.4: The extracted latent measurements for the pendulum task, $\mathbf{a}_k \in \mathbb{R}^2$, from clean images (filled circles) and corrupted images (hollow squares and triangles) with their respective covariances (in red). Note that the covariances of the clean measurements are small and more difficult to visually distinguish.

clean, partially obstructed, and noisy (i.e., with added Gaussian pixel noise). The uncertainties of the measurements associated with the degraded images are substantially larger than the those of the clean images, as we would expect.

Prediction Experiment

We tested the capability of the model to predict 20 future latent states, $\hat{\mathbf{z}}_{5:24}$, based on just four initial images from a validation set, $\mathbf{x}_{1:4}$. To compute the future states, we used the same control inputs $\mathbf{u}_{4:23}$ which were sent to the pendulum and produced the ground-truth images $\mathbf{x}_{5:24}$. The initial images were first embedded into measurements $\mathbf{a}_{1:4}$ using the encoder network. We recovered the initial latent states $\mathbf{z}_{1:4}$ with the standard Kalman filter equations. We then 'rolled-out' a trajectory, using the learned transition and control matrices from the GRU network, in order to arrive at the predicted future latent states $\hat{\mathbf{z}}_{5:24}$. The learned measurement matrices, also from the GRU network, were used to generate the predicted measurements $\hat{\mathbf{a}}_{5:24}$ from the predicted latent states $\hat{\mathbf{z}}_{5:24}$. Finally, the decoder network utilized the predicted measurements to generate predicted images $\hat{\mathbf{x}}_{5:24}$. The results are shown in Fig. 3.5. Two of the initial images were corrupted, by noise (Image 2) and by a single-frame obstruction (Image 4). For the baseline model, with no notion of heteroscedastic uncertainty (Row 3), the added noise and obstruction in the initial images caused the future predictions to be highly inaccurate when compared to the ground-truth images (Row 1). On the other hand, when using our heteroscedastic uncertainty weighing (Row 2), the model was capable of effectively ignoring the corrupted images and predicting the future states accurately.

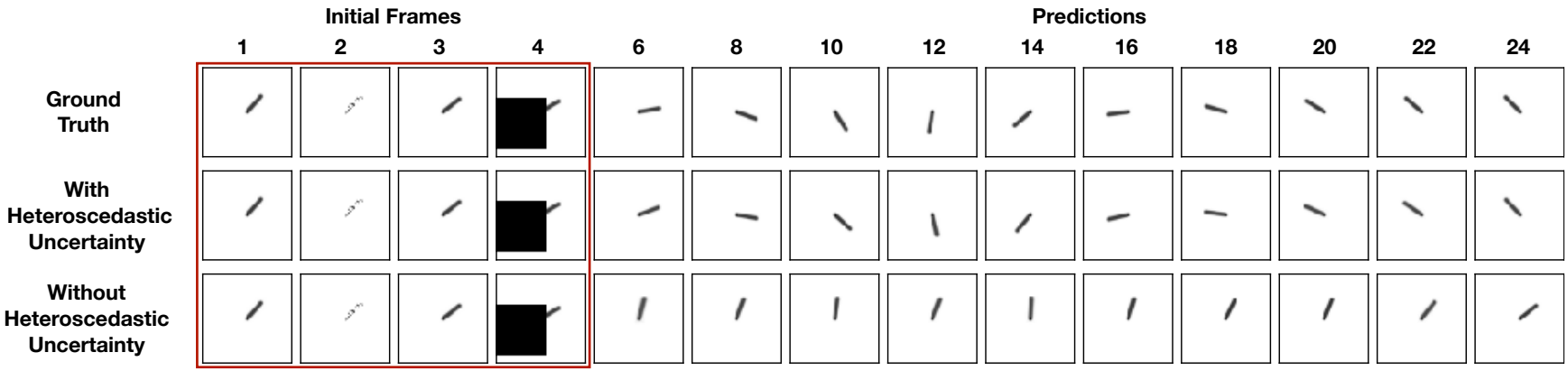
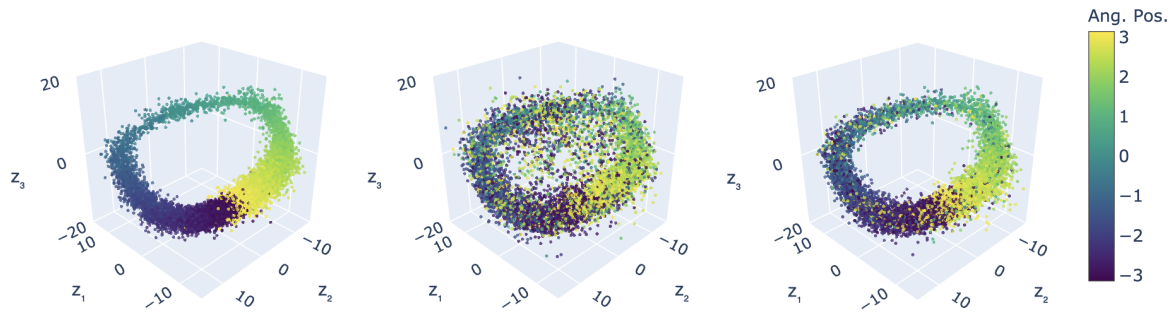


Figure 3.5: Image prediction results for the simulated pendulum task. Images 2 and 4 are corrupted with noise and an obstruction, respectively. The first four columns, marked by a red rectangle, represent the frames used to initialize our state. The remaining images are generated from future latent state predictions by our learned dynamics model and the decoder network. The number at the top identifies the time step for the images in that column. *Top row:* The ground truth images. *Middle row:* Image predictions with use of heteroscedastic uncertainty. *Bottom row:* Image predictions without use of heteroscedastic uncertainty.



(a) (b) (c)

Figure 3.6: Predicted or generated latent states, $\hat{\mathbf{z}}_k \in \mathbb{R}^3$, for the pendulum task. The latent state is initialized as follows. *Left*: 4 clean images, *Middle*: 2 out of 4 initial images obstructed randomly and no heteroscedastic uncertainty, *Right*: 2 out of 4 initial images obstructed randomly and with heteroscedastic uncertainty. Points are colour-coded by the ground truth joint angle.

As a final experimental verification in the pendulum environment, we visualized sets of predicted future latent states (Fig. 3.6). Each subplot in Fig. 3.6 shows the predicted latent states under specific conditions. On the left, we visualized the predicted latent states when all four of the initial images were unobstructed and noise-free. For the middle and right subfigures, we corrupted two of the four images with randomly placed obstructions. The middle subfigure demonstrates that, without heteroscedastic uncertainty modelling, the ring-like structure and the ordering of the points (in terms of their colour-coded joint angles) is compromised. In the right subfigure, we show that the use of heteroscedastic uncertainties better preserves the manifold structure and joint angle ordering despite the presence of the same obstructions.

3.9.2 Real-World Visual Reacher

We tested our model on a more difficult (real-world) 2D visual reaching task. This is a continuous control task based on the real-world robot reinforcement learning benchmark 'UR-Reacher-2' described in (Mahmood et al., 2018). We collected a dataset of images of a robot arm performing the reaching task, along with the corresponding control inputs. Our experimental setup is shown in Fig. 3.7. The control inputs $\mathbf{u} \in \mathbb{R}^2$ were restricted to the velocities of joints 2 and 3 of the UR10 arm. Input images were captured by an external, fixed camera. We emphasize that the joint encoder readings were not used to train our dynamics model; we kept track of the joint angles solely for ground truth evaluation and analysis. The input images were downsampled to $\mathbf{x} \in \mathbb{R}^{64\times64}$ and converted to grayscale. We chose a latent space $\mathbf{z} \in \mathbb{R}^{10}$ and a measurement space $\mathbf{a} \in \mathbb{R}^4$. In order to enforce synchronized communication with reduced control latency, we relied on the SenseAct framework (Mahmood et al., 2018) to gather our data. We collected a total of $N = 1024$ trajectories for training, each with a length of 15 time steps, where each step had a duration of 0.5 seconds.

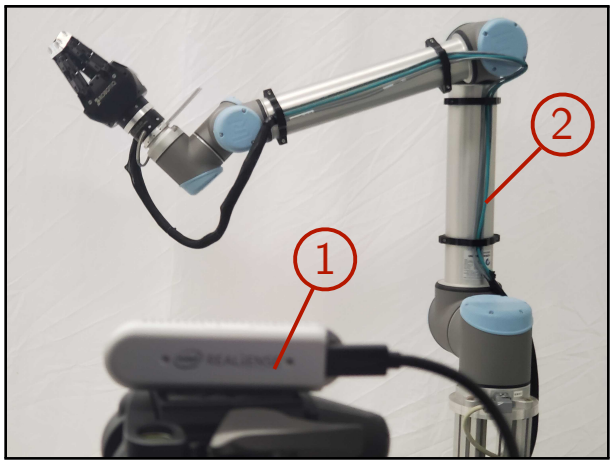


Figure 3.7: Experimental setup for our real-world visual reaching task: (1) UR10 manipulator and (2) Intel RealSense D435 camera.

Prediction Experiment

As shown in Fig. 3.8, we tested our model's predictive capability in the presence of noise (Image 2) and a single-frame obstruction (Image 4). We followed the same procedure detailed in Section 3.9.1. Similar to the results for the pendulum experiment, the model was better able to predict future states in the presence of corrupted measurements when it incorporated heteroscedastic uncertainty. Fig. 3.9 demonstrates the robustness of the model to a real (non-simulated) obstruction (i.e., a person blocking the camera) in Images 3 and 4.

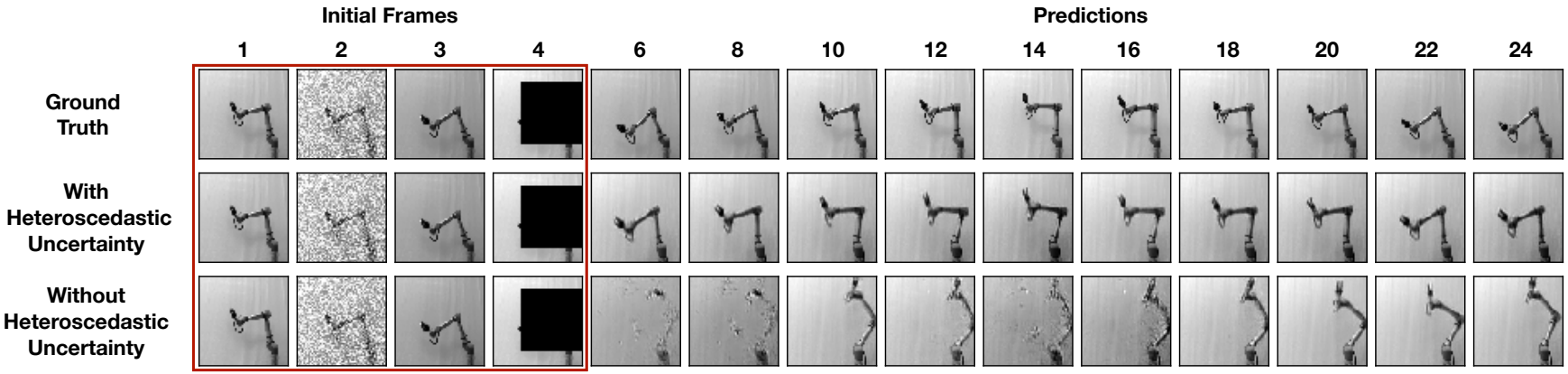


Figure 3.8: Image prediction results for the real-world visual reaching task. Images 2 and 4 are corrupted with noise and an obstruction, respectively. The first four columns, marked by a red rectangle, represent the frames used to initialize our state. The remaining images are generated from future latent state predictions by our learned dynamics model and the decoder network. The number at the top identifies the time step for the images in that column. *Top row:* The ground truth images. *Middle row:* Image predictions with use of heteroscedastic uncertainty. *Bottom row:* Image predictions without use of heteroscedastic uncertainty.

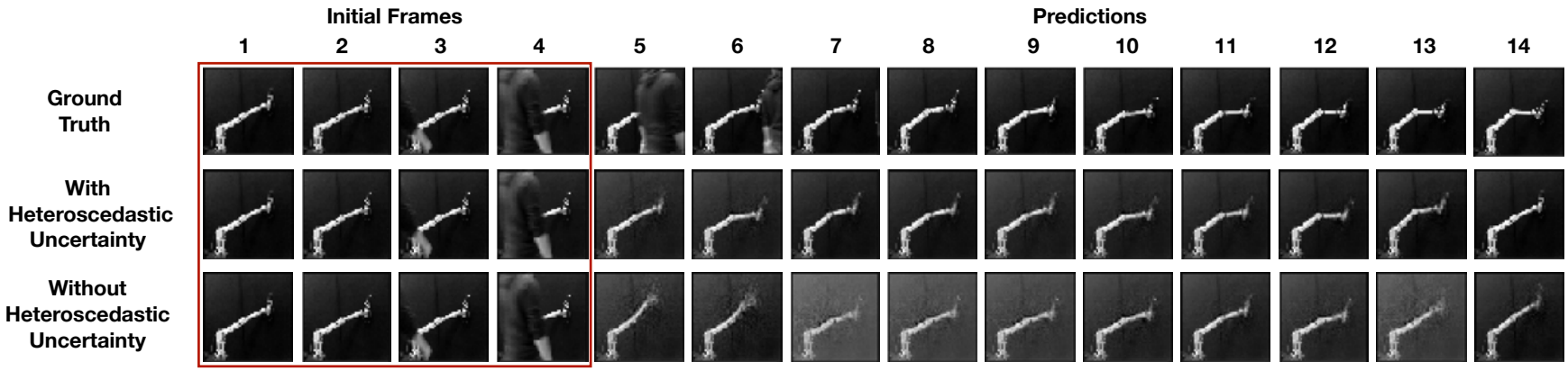
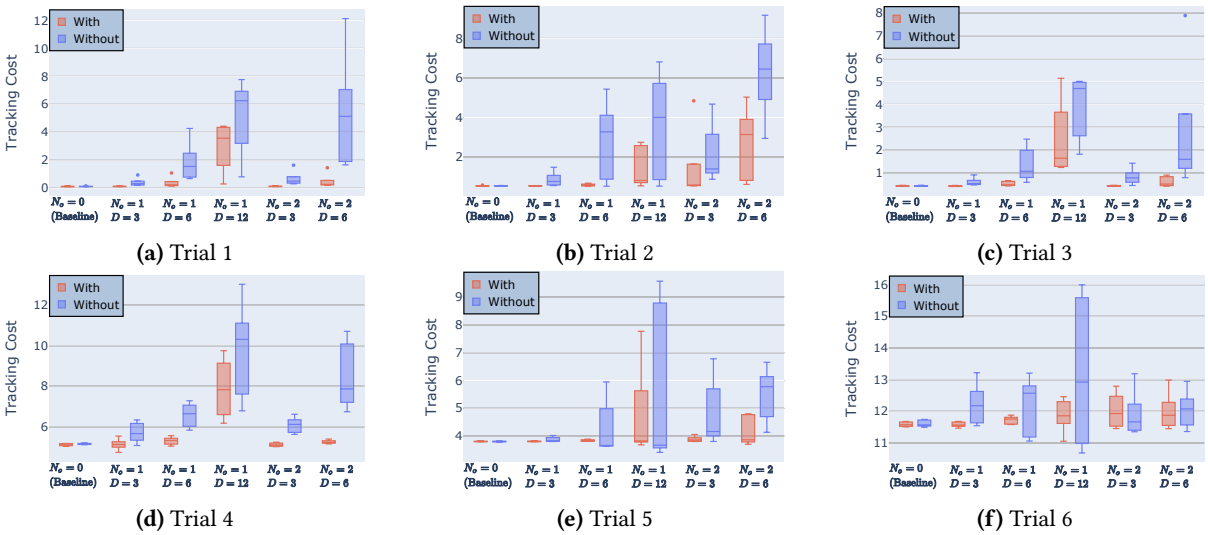


Figure 3.9: Image prediction results for the real-world visual reaching task. Image 4 is corrupted with a person obstructing the camera. The first four columns, marked by a red rectangle, represent the frames used to initialize our state. The remaining images are generated from future latent state predictions by our learned dynamics model and the decoder network. The number at the top identifies the time step for the images in that column. *Top row:* The ground truth images. *Middle row:* Image predictions with use of heteroscedastic uncertainty. *Bottom row:* Image predictions without use of heteroscedastic uncertainty.



(a) Trial 1 **(b)** Trial 2 **(c)** Trial 3

(d) Trial 4 **(e)** Trial 5 **(f)** Trial 6

Figure 3.10: Comparison of tracking costs for six trials with different initial images and the same goal image. For each trial, we tested six different occlusion settings, five times each ((*a*) - (*f*)). We varied the number of times an occlusion was present, N_o, and the 'duration' or number of images sequentially occluded, D. The data coloured in red with the label 'With' includes the use of heteroscedastic uncertainty, while the data coloured in blue with the label 'Without' does not.

Control Experiments

We investigated the control performance of the model by introducing a varying number of synthetic obstructions, each of which appeared for a varying duration. A summary of the results is shown in Fig. 3.10. We used the following cumulative squared tracking error, $\sum_{k=1}^{K} \|\mathbf{q}_{\text{goal}} - \mathbf{q}_k\|^2$, where $\mathbf{q}_{\text{goal}}$ and $\mathbf{q}_k$ are the manipulator's joint positions for the goal image and at time step k, respectively. We ran each model with the controller for $K = 24$ time steps, where each time step lasted 0.5 seconds, with the following settings for the MPC optimization: $\mathbf{Q}_{\text{mpc}} = \mathbf{I}$, $\mathbf{R}_{\text{mpc}} = \mathbf{I}$, and a prediction horizon of $T = 9$. The use of synthetic degradations allowed us to fairly perform an ablation study by comparing the performance of both models under identical conditions. We randomly and uniformly sampled a time step or multiple time steps within the trajectory at which an obstruction was visible. We ran six different trials with different initial images and the same goal image. For each of the trials, we applied five different 'classes' of occlusions, by varying the number of times an occlusion appeared, N_o, and the 'duration' or quantity of images sequentially occluded, D. We repeated each test five times. The same random seed was used for trials with and without our heteroscedastic uncertainty weighing. In all cases, our model achieved a significantly lower tracking cost by being more robust to occlusions. We visualize four example joint space trajectories in Fig. 3.11, where the manipulator joint encoders provided ground truth. For these experiments, we alternated between two unobstructed images and two obstructed images. The trajectory in orange was produced by the model with heteroscedastic weighing, while the trajectory in blue was produced without this weighting. Heteroscedastic uncertainty weighing confers the ability to more consistently and accurately track the target arm pose; predictions made with homoscedastic uncertainties led to erratic control behaviour and poor performance.

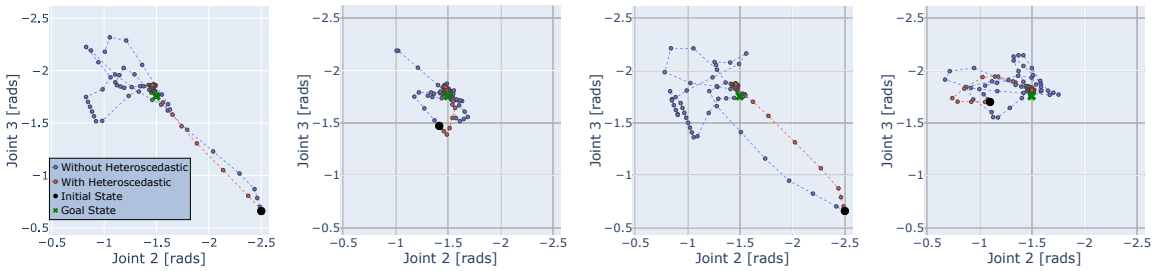


Figure 3.11: Visualization of four different trajectories from our control experiments, with and without heteroscedastic uncertainty weighing. The model without weighing was unable to reach and remain in the configuration shown by the goal image. The obstructions caused the manipulator to move erratically. The model with heteroscedastic uncertainty weighing was able to reach and remain at the goal.

3.10 Experiments on Learning with the Multimodal SLV Model

For our second set of experiments, we investigated the behaviour of the multimodal (SLV) model (introduced in Section 3.6) when learning a planar pushing task. The planar pushing task involved a robotic manipulator pushing an object on a plane. The manipulator was equipped with sensors that provided vision, haptic, and proprioceptive data from a camera, a force-torque sensor, and joint encoders, respectively. Planar pushing incorporates complex contact dynamics that are difficult to model with vision alone, while the multimodal sensor data produced are highly heterogeneous in both dimension and quality.

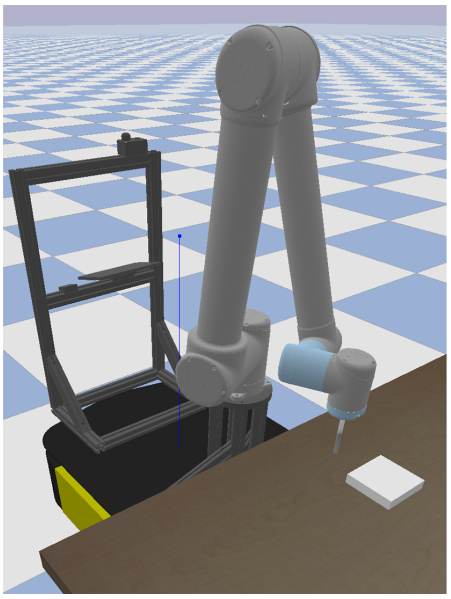
We compared five different models to study the effects of multimodality on this task. The models were, in order: 1) vision-only (denoted by V), 2) visual-proprioceptive (denoted by VP), 3) visual-haptic (denoted by VH), 4) visual-haptic-proprioceptive (denoted by VHP), and 5) a commonly-used baseline where the latent embedding of each data modality is simply concatenated (denoted by VHP-C).

3.10.1 Datasets

We use both a synthetic and a real-world dataset as shown in Fig. 3.12. We provide more details on the data and the collection procedure below.

Simulated Manipulator Pushing

We used PyBullet (Coumans and Bai, 2019) to generate data from a simulated manipulator pushing task. We collected grayscale images of size 64×64 pixels, $\mathbf{x}_t^1 \in \mathbb{R}^{64\times64}$, with pixel intensities rescaled to be in the range of zero to one. The proprioceptive data consisted of the Cartesian position and velocity of the end-effector, while the haptic data included three-axis force and torque measurements. We combined the haptic and proprioceptive data into a single second modality when both were used, $\mathbf{x}_t^2 \in \mathbb{R}^{12}$, and otherwise used one of the two, $\mathbf{x}_t^2 \in \mathbb{R}^6$. We argue that this is reasonable since both proprioceptive and haptic data have similar characteristics, as opposed to, for example, image data. The control inputs were the end-effector velocity commands along the planar x and y directions, $\mathbf{u}_t \in \mathbb{R}^2$.



(a) PyBullet pushing simulation.

(b) Real-world pushing task.

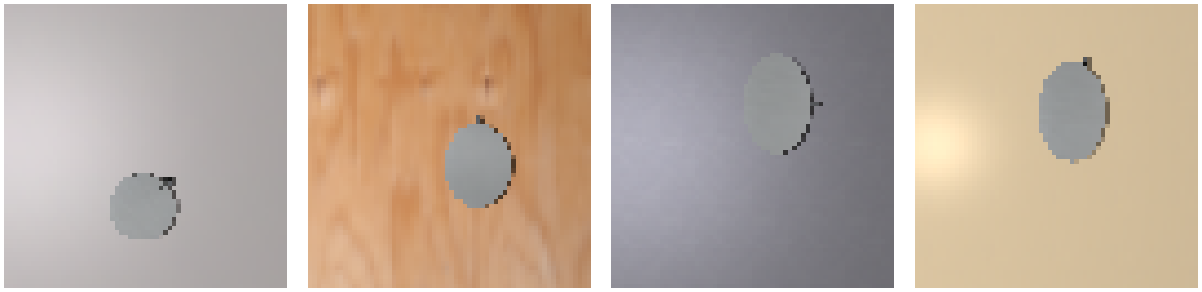
Figure 3.12: Environments from which the datasets were collected. Real-world setup image from existing MIT pushing dataset (Yu et al., 2016).

A total of 4,800 trajectories were collected. The image data was recorded at a frequency of approximately 4 Hz and force-torque and proprioceptive data at a frequency of 120 Hz. We concatenated sequences of measurements in order to keep a consistent number of time steps between modalities. We used 4,320 trajectories for training and held out 480, or 10%, for evaluation. Each trajectory was of length $T = 16$ (i.e., 16 images and 16×32 (concatenated) force-torque and proprioception measurements in total). The number of measurements per discrete time step was based on the respective frequency of each data source. The object being pushed was a single square plate. In order to collect training data, we used a policy with actions drawn from a fixed Gaussian distribution. We used the same initial object position for each trajectory.

MIT Pushing

The MIT pushing dataset (Yu et al., 2016) consists of high-fidelity real robot pushes carried out on various material surfaces and with various object shapes. Our dataset was a subset of trajectories with three different ellipse-shaped plates and four different surface materials (Delrin, plywood, ABS, and polyurethane). We followed the experimental protocol of the authors in (Lee et al., 2020).

We used the code provided by previous work (Kloss et al., 2020) to preprocess the data and to artificially render the respective images of the trajectories (since no image data was collected as part of the original dataset). In Fig. 3.13, we show four examples of rendered images based on the real-world pushing data (i.e., object pose, end-effector pose, and force-torque data). We note that the rendered images are not completely realistic (e.g., the images are not occluded and only the manipulator's tip is rendered). However, for our purposes, the dataset provided an adequate starting point for preliminary



(a) Ellipse-1 on Delrin. **(b)** Ellipse-2 on plywood. **(c)** Ellipse-3 on polyurethane. **(d)** Ellipse-3 on polyurethane.

Figure 3.13: Downsampled images generated from the real-world MIT pushing dataset using existing tools (Kloss et al., 2020). Realistic lighting and material textures are also rendered.

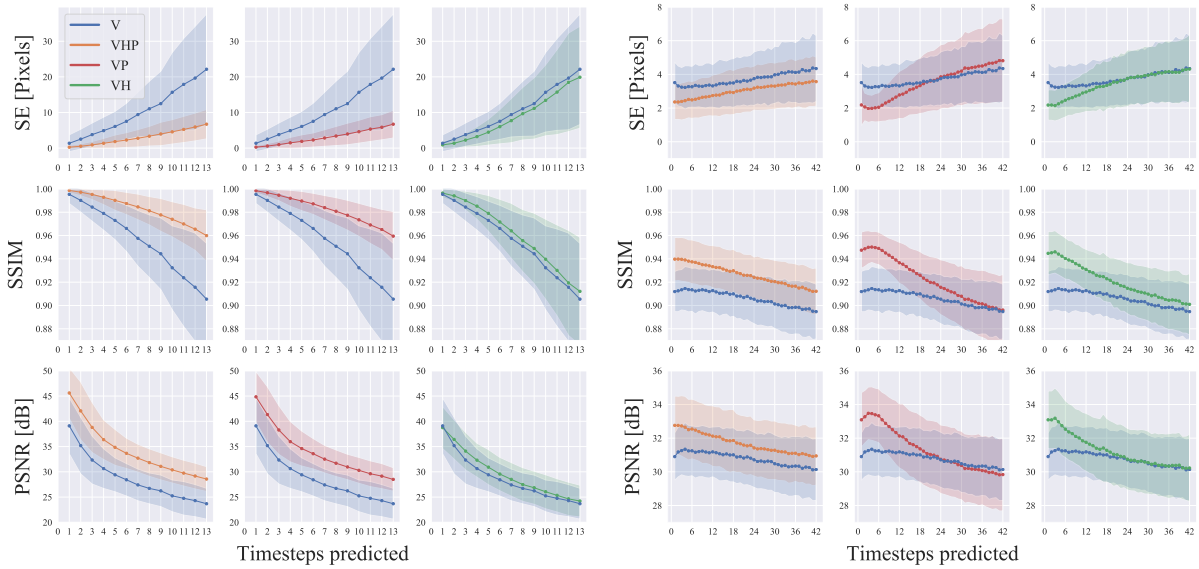
experiments and comparisons.

We downsampled the images to 64×64 pixels and transformed them to grayscale, $\mathbf{x}_t^1 \in \mathbb{R}^{64\times64}$, with pixel intensities rescaled to be in the range of zero to one. The proprioceptive data consisted of the x and y Cartesian coordinates of the end-effector. The haptic data consisted of force measurements along the x and y (in-plane) axes and the torque measurements about the z axis. We combined the haptic and proprioceptive data to form a single second modality, $\mathbf{x}_t^2 \in \mathbb{R}^5$, if both were used, and otherwise used one or the other, $\mathbf{x}_t^2 \in \mathbb{R}^3$ or $\mathbf{x}_t^2 \in \mathbb{R}^2$. The control inputs were position commands along the x and y axes, $\mathbf{u}_t \in \mathbb{R}^2$. The images were recorded at a frequency of 18 Hz and the force and proprioceptive data were recorded at a frequency of 180 Hz. We used a subset of 2,332 trajectories out of the total dataset (2,099 trajectories for training and 233, or approximately 10%, held out for evaluation). Each trajectory was of length $T = 48$ (i.e., 48 images and 48×10 (concatenated) force-torque and proprioception measurements in total). The data were collected using several pre-planned pushes with varying velocities, accelerations, and contact points and angles.

3.10.2 Image Prediction Experiments

In order to compare the models, we first evaluated the quality of the generated image sequences relative to the known ground truth images as a proxy for prediction quality. We began by encoding an initial amount of data into the latent space and then predicted future latent states with our learned dynamics models and known control inputs. Given these predicted states, we then decoded and generated future predicted images. We computed the squared error (SE) in terms of pixels, structural similarity index measure (SSIM), and peak signal-to-noise ratio (PSNR) between the ground truth images and predicted images. Accurate predictions translate into a lower SE, higher SSIM, and higher PSNR.

For the synthetic data, we first conditioned on four frames and predicted the next 11 frames in the sequence. Fig. 3.14a is a visualization of the mean score, with one standard deviation shaded, based on all of the held out test data. For clear comparisons, we overlay each multimodal model (i.e., VHP, VP, and VH) on top of the baseline vision-only model (i.e., V). Additionally, we compile the average score over the entire predicted horizon and summarize the results in Table 3.1. We also include the baseline results of simply concatenating all modalities, labelled as VHP-C.



(a) Results from the synthetic dataset.

(b) Results from the MIT dataset.

Figure 3.14: Graph of prediction quality over multiple prediction horizons from both datasets. We compare each model to the baseline vision-only (V) model. We plot the mean values of the SE, SSIM and PSNR with one standard deviation shaded.

Table 3.1: Quantitative prediction quality for the synthetic dataset. We compared the predicted images' quality with their respective ground truth for four models trained with various subsets of modalities. We show one standard deviation for the SSIM and PSNR values and calculate the average score across the entire predicted horizon.

Model	**RMSE** (↓)	**SSIM** (↑)	**PSNR** (↑)
V	3.243	0.955 ± 0.04	28.639 ± 5.84
VP	1.758	0.982 ± 0.02	33.935 ± 5.93
VH	2.956	0.961 ± 0.04	29.684 ± 5.70
VHP-C	2.911	0.962 ± 0.04	29.700 ± 5.39
VHP	**1.749**	**0.983 ± 0.02**	**34.202 ± 6.19**

Table 3.2: Quantitative prediction quality with the MIT dataset as described in Table 3.1.

Model	**RMSE** (↓)	**SSIM** (↑)	**PSNR** (↑)
V	1.917	0.907 ± 0.02	30.806 ± 1.66
VP	1.850	0.923 ± 0.03	31.395 ± 2.28
VH	1.839	0.922 ± 0.03	31.298 ± 2.03
VHP-C	1.846	0.923 ± 0.03	31.333 ± 2.12
VHP	**1.734**	**0.927 ± 0.02**	**31.741 ± 1.84**

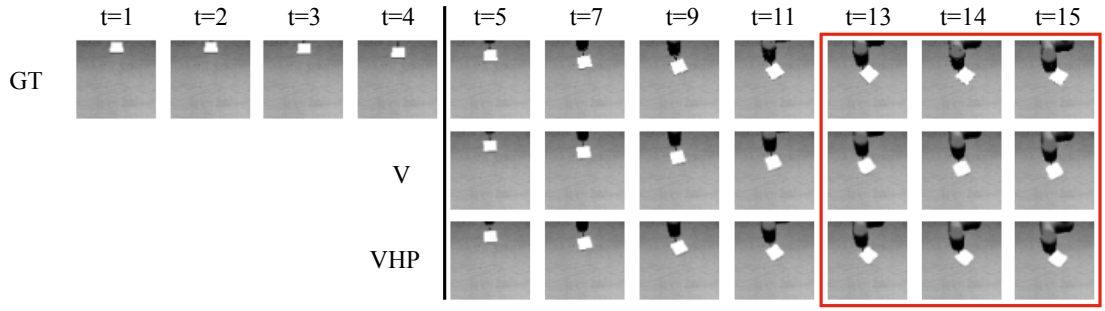


Figure 3.15: A prediction example that demonstrates a failure mode of the vision-only (V) model trained on the synthetic dataset; the model fails to fully predict the object's true motion (GT) and instead predicts less counterclockwise rotation in the later frames (denoted by red box). In contrast, the multimodal model (VHP) correctly predicts the object's motion.

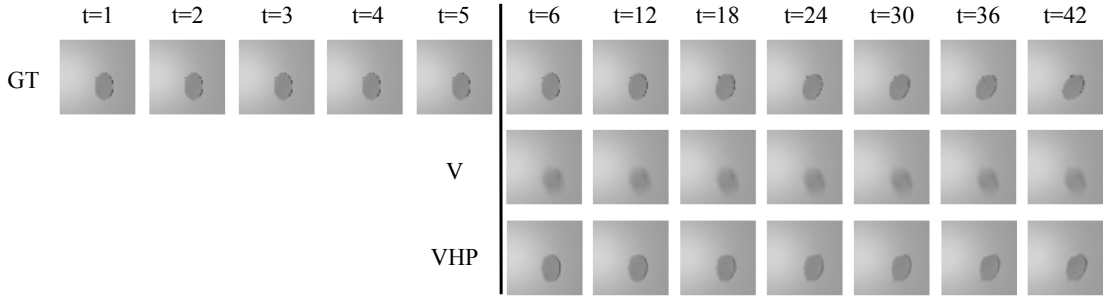


Figure 3.16: A prediction example that demonstrates a different failure mode of the vision-only (V) model trained on the MIT dataset; the model is unable to produce 'crisp' predictions of the object form and instead outputs a blurry average at all time steps. The multimodal model (VHP) is able to predict the slight angle of the ellipse (clockwise), matching the ground truth (GT).

Overall, the VHP model performed the best for all three metrics when averaged over all prediction lengths, but it was closely followed by the VP model. All multimodal models outperformed the baseline, vision-only model, although the improvement in the case of the VH model was marginal. We hypothesize that this result may be due to the quality of the simulated force torque sensor data and the simulated contact dynamics. In this case, proprioception was the most effective second modality, as demonstrated by the performance of the VP model. However, using vision, haptic and proprioceptive data together with the VHP model led to the best results.

Notably, simply concatenating each modality (i.e., VHP-C) yielded poorer performance when compared to our product of experts approach (i.e., VHP). The product of experts appears to have an appealing 'filtering' inductive bias for prediction problems. The product of experts formulation enables decisions to be made about when to use each modality at each time step based on the respective uncertainties. Further, each expert can selectively focus on a few dimensions without having to cover the full dimensionality of the state. Finally, a product of experts produces a sharper final distribution than the individual expert models (Hinton, 2002). This idea is well known in the context of digit image generation: one low-resolution model can generate the approximate overall shape of the digit while other local models can refine segments of the stroke with the correct fine structure (Hinton, 2002). Fig. 3.15 is a visualization of a selected roll-out or prediction that demonstrates a typical failure mode

of the vision-only model. As shown by the later frames generated by the vision-only model in the red box outline, the overall motion of the object is almost correct, but the finer-scale changes (e.g., the exact amount of rotation) are not well captured. Our VHP model, however, was able to capture these smaller changes.

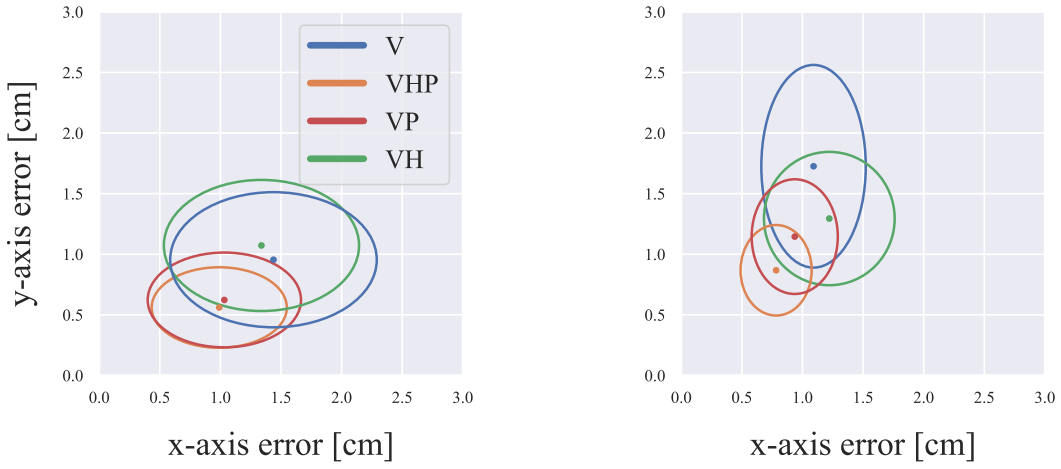
We ran similar image prediction experiments for the MIT pushing dataset. In this case, we first conditioned on five frames and predicted the next 37 frames in the sequence. As shown by Fig. 3.14b, both the VP and VH models performed slightly better than the full VHP during the early time steps according to all three metrics (SE, SSIM and PSNR). However, over a longer prediction horizon the VHP model was clearly superior. For the real-world data, unlike in the synthetic case, the VH model significantly outperformed the baseline V model. Table 3.2 lists the average scores over the entire prediction horizon. Consistent with the previous synthetic experiment, simply concatenating the modalities (VHP-C) led to poorer performance when compared to the product of experts model (VHP). Each multimodal model performed better than the baseline vision-only model. We visualize another failure mode of the vision-only model in Fig. 3.16. In this case, we observe that the vision-only model failed to capture the more subtle dynamics of the object, which ended up angled and tilted clockwise; the model defaulted to outputting a blurry average at the approximate location of the object. On the other hand, our VHP model was able to produce a relatively crisp and accurate prediction of the object pose.

3.10.3 Regression Experiments

We further evaluated the predictive capability of the models with a downstream regression experiment that measured how well the model was able to capture and predict the underlying position of the object being pushed. As is commonly done in the self-supervised representation learning literature (Grill et al., 2020; Kornblith et al., 2019), we trained separate regressors on top of the frozen representations and dynamics models to do this.

We first encoded an initial latent state $\mathbf{z}_i$ (i.e., a filtered state) from a set of observations $\mathbf{x}_{1:i}$. Then, using our learned dynamics and h known future controls $\mathbf{u}_{i:i+h}$, we predicted h future latent states $\hat{\mathbf{z}}_{i+1:i+1+h}$ (i.e., predicted states). Using the predicted states as inputs, we regressed the ground truth positions of the object, while keeping the weights of all of our previously-learned networks frozen. Poor regression results would indicate lower correlation between the predicted latent states and the object's ground truth position. This in turn would imply that the learned latent representation did not encode all of the necessary information to represent the state of the object, or that the learned dynamics of the object were inaccurate, or both. We kept the same prediction horizons from the image prediction experiments in Section 3.10.2.

For our results with synthetic data, we found ordinary least squares to be adequate for regression. Fig. 3.17a shows that the mean and variance of the absolute translation errors from the regression are lowest for the full VHP model. We (again) observe that the inclusion of haptic data alone (VH) as an extra modality in the model did not lead to huge improvements. On the other hand, using proprioception (VP) did lead to significant improvements. The inclusion of both proprioceptive and haptic data (VHP) outperformed using proprioceptive data alone (VP).



(a) Results from the synthetic dataset.

(b) Results from the MIT dataset.

Figure 3.17: Mean translation errors for regressing the object's x and y coordinates from the predicted latent states. The ellipses represent one standard deviation.

For the MIT pushing dataset, we trained a separate simple neural network with a single hidden layer of size 50 to regress the position of the object. The real-world results generally match our synthetic results, as shown by Fig. 3.17b. Including any sort of multimodal data reduced the mean and variance of the absolute translation errors when compared to the results from the vision-only model.

As an additional benchmark with the MIT dataset, we compared the accuracy of our regressed values of the position of the object to those computed by a related approach on differentiable filtering (Lee et al., 2020). Using the same dataset, the authors trained a variety of differentiable filters to directly regress the positions of the objects from the multimodal sensor data in a supervised manner using ground truth labels. Because the multimodal differentiable filters were trained with ground truth annotations of the objects' locations, while our model is completely self-supervised, the results provide a reasonable estimate of an upper bound on the expected performance. Table 3.3 demonstrates that our regression results, from both the predicted and filtered states, are comparable.

Table 3.3: Root-mean-square error (RMSE) of the objects' regressed positions with the MIT pushing dataset. We compared both the filtering (filt.) and prediction (pred.) results from our self-supervised multimodal model (VHP) to various types of differentiable filters from (Lee et al., 2020).

Model	**RMSE [cm]** (↓)
EKF (Sup. w/ GT, filt.)	1.33
PF (Sup. w/ GT, filt.)	1.14
LSTM (Sup. w/ GT, filt.)	2.32
VHP (Self-sup., filt.) (Ours)	1.80
VHP (Self-sup., pred.) (Ours)	1.96

3.11 Summary and Future Work

In this chapter, we leveraged the generative latent variable modelling approach to learn latent SSMs directly from sequential observations. Our contributions extend the existing literature on latent SSMs, a key backbone of many methods in vision-based control, model-based reinforcement learning, and language modelling, in two novel ways. First, we introduced a robust generative latent model capable of accurately predicting future states under difficult input degradations that (ideally) should not affect the underlying dynamical system. Second, we provided a principled probabilistic formulation for self-supervised, SLV modelling of multimodal time series data. We demonstrated the benefits of these contributions through a range of experiments in simulation and in the real world.

There are several avenues for future work. The combination novelty detection with self-supervised uncertainty learning would be an interesting area to investigate. Uncertainties play a crucial role in both filtering OOD data and the fusion of multimodal data. We chose to weight our measurement uncertainties using a reconstruction-based loss as a proxy for novel or OOD detection. This only captures the epistemic uncertainty, or what the model does not know, and not the inherent uncertainty in the data itself. It would be valuable to study whether uncertainties can be captured in the self-supervised sequential setting.

3.12 Associated Publications

1. Limoyo, O., Chan, B., Maric, F., Wagstaff, B., Mahmood, R., and Kelly, J. (2020b). Heteroscedastic uncertainty for robust generative latent dynamics. *IEEE Robotics and Automation Letters*, 5(4):6654–6661

2. Limoyo, O., Ablett, T., and Kelly, J. (2023a). Learning sequential latent variable models from multimodal time series data. In *Intelligent Autonomous Systems 17*, volume 577 of *Lecture Notes in Networks and Systems*, pages 511–528. Best Paper Finalist

Chapter 4

Generative Graphical Inverse Kinematics

In this chapter, we apply the generative SSL paradigm to the problem of robot inverse kinematics. In the previous chapter, we showed how the temporal structure of image sequences could be used as a learning signal. Here, we demonstrate that a robot's kinematic structure can also be used as a learning signal. Perhaps obvious, the forward mapping from joint angles to end-effector pose provides valuable labels for the inverse mapping from end-effector pose to joint angles. As previously done with SSMs, we formulate our inverse kinematics solver as a generative model that generates joint angle solutions conditioned on an end-effector pose. We introduce a novel framework called *generative graphical inverse kinematics* (GGIK). By using a probabilistic generative formulation, we are able to quickly find multiple feasible inverse kinematics solutions with a single forward pass through a neural network. Critically, our solver, which uses a distance geometric representation of manipulators and employs graph neural networks for learning, is also able to generalize across multiple manipulators.

4.1 Motivation

Robotic manipulation tasks are naturally defined in terms of end-effector poses. However, the configuration of a manipulator is typically specified in terms of joint angles, and determining the joint configuration(s) that correspond to a given end-effector pose requires solving the *inverse kinematics* (IK) problem. For redundant manipulators (i.e., those with more than six degrees of freedom or DOF), target poses may be reachable by an infinite set of feasible configurations. While redundancy allows high-level algorithms (e.g., motion planners) to choose configurations that best fit the overall task, it makes solving IK substantially more involved.

Since the full set of IK solutions cannot, in general, be derived analytically for redundant manipulators, individual configurations reaching a target pose are found by locally searching the configuration space using numerical optimization methods and geometric heuristics. The search space is sometimes reduced by constraining solutions to have particular properties (e.g., collision avoidance, manipula-

bility). While existing numerical solvers are broadly applicable, they usually only produce one single solution at a time and rely on incremental search techniques to minimize highly nonconvex objective functions.

These limitations have motivated the use of learned IK models that approximate the entire feasible set of solutions. In terms of success rate, learned models that output individual solutions are able to compete with the best numerical IK solvers when high accuracy is not required (von Oehsen et al., 2020). Data-driven methods are also useful for integrating abstract criteria such as "human-like" poses or motions (Aristidou et al., 2018). Generative approaches (Ren and Ben-Tzvi, 2020; Ho and King, 2022) have demonstrated the ability to rapidly produce a large number of approximate IK solutions and to even model the entire feasible set for specific robots (Ames et al., 2022). Access to a large number of configurations fitting desired constraints has proven beneficial in motion planning applications (Lembono et al., 2021). Unfortunately, these learned models, parameterized by deep neural networks (DNNs), require specific configuration and end-effector input-output vector pairs for training (by design). In turn, it is not possible to generalize learned solutions to robots that vary in link geometry and DOF. Ultimately, this drawback limits the utility of learning for IK over well-established numerical methods that are easier to implement and to generalize (Beeson and Ames, 2015).

In this chapter, we describe a generative graphical inverse kinematics (GGIK) model and explain its capacity to simultaneously represent general (i.e., not tied to a single robot manipulator model or geometry) IK mappings and to produce approximations of entire feasible sets of solutions. In contrast to existing DNN-based approaches (Ren and Ben-Tzvi, 2020; Lembono et al., 2021; von Oehsen et al., 2020; Ho and King, 2022; Ames et al., 2022), we explore a new path towards learning generalized IK by adopting a *graphical* description of robot kinematics. This graph-based description, first introduced in (Porta et al., 2005) and extended and generalized in (Maric et al., 2021), is formulated using distances between points rigidly attached to the robot structure. IK can then be cast as the problem of finding unknown distances (i.e., weights in a distance graph), allowing us to make use of graph neural networks (GNNs) to capture varying robot geometries and DOF within a single learned model. Furthermore, the graphical formulation exposes the symmetry and Euclidean equivariance of the IK problem that stems from the spatial nature of robot manipulators. We exploit this symmetry by encoding it into the structure of our model architecture to efficiently learn accurate IK solutions. When compared to several other learned IK methods, GGIK provides more accurate results. GGIK is also able to generalize reasonably well to *unseen* robot manipulators. We show that GGIK can complement local IK solvers by providing reliable initializations. Finally, we demonstrate that GGIK is capable of learning a constrained distribution of solutions that obey joint limits.

4.2 Related Work

Our work leverages prior research in several diverse areas, including classical IK methods, learning for IK and motion planning, and generative modelling on graphs. We briefly summarize relevant literature in each of these areas below.

4.2.1 Inverse Kinematics

Robotic manipulators with six or fewer DOF can reach any feasible end-effector pose (Lee and Liang, 1988) with up to sixteen different configurations that can be determined analytically using various representations (Manocha and Canny, 1994; Husty et al., 2007). For example, the IKFast algorithm (Diankov, 2010) uses symbolic computation to recover the full set of configurations as solutions to polynomial equations that are robot-specific. However, analytically solving IK for redundant manipulators is generally infeasible and numerical optimization methods or geometric heuristics must be used instead. Simple instances of the IK problem involving small changes in the end-effector pose (e.g., for kinematic control or reactive planning) are also known as *differential* IK. These instances can be reliably solved by computing required configuration changes using a first-order Taylor series expansion of the manipulator's forward kinematics (Whitney, 1969; Sciavicco and Siciliano, 1986). As an extension of this approach, the infinite solution space afforded by redundant DOF can be used to optimize secondary objectives, such as avoiding joint limits, for example, through redundancy resolution techniques (Nakamura et al., 1987).

By incrementally "guiding" the end-effector along a path in the task space, the differential IK approach can also be used to reach end-effector poses far from the initial configuration (e.g., a pose enabling a desired grasp) (Lynch and Park, 2017). This typically amounts to solving a sequence of convex optimization problems (Boyd and Vandenberghe, 2004) such as quadratic programs (QPs) using general-purpose solvers (Goldfarb and Idnani, 1983; Wächter and Biegler, 2006; Bambade et al., 2022). However, these *closed-loop IK* techniques are particularly vulnerable to singularities and may require a user-specified end-effector path that reaches the goal (Siciliano et al., 2010).

The "guidance-free" or full IK problem (Tedrake, 2022) that appears in applications such as motion planning is usually solved with first-order (Engell-Nørregård and Erleben, 2009; Beeson and Ames, 2015) or second-order (Deo and Walker, 1993; Erleben and Andrews, 2019) nonlinear programming methods formulated over joint angles. These methods have robust theoretical underpinnings (Nocedal and Wright, 1999) and are featured in popular libraries such as KDL (Bruyninckx, 2002) and TRAC-IK (Beeson and Ames, 2015), which (approximately) support a range of constraints through the addition of penalties to the cost function. However, the highly nonconvex nature of IK makes them susceptible to local minima, often requiring multiple initial guesses before returning a feasible configuration, if at all. Such issues can sometimes be circumvented through the use of heuristics such as the cyclic coordinate descent (CCD) algorithm (Kenwright, 2012), which iteratively adjusts joint angles using simple geometric expressions. Similarly, the FABRIK (Aristidou and Lasenby, 2011) algorithm solves the IK problem using iterative forward and backward passes over joint positions to quickly find solutions.

Some alternative IK solvers forgo the joint angle parametrization in favour of Cartesian coordinates and geometric representations (de Jalón, 2007). Dai et al. (Dai et al., 2019) use a Cartesian parameterization together with a piecewise-convex relaxation of $\mathrm{SO}(3)$ to formulate IK as a mixed-integer linear program, while Yenamandra et al. (Yenamandra et al., 2019) apply a similar relaxation to formulate IK as a semidefinite program. Le Naour et al. (Le Naour et al., 2019) express IK as a nonlinear program over interpoint distances, showing that solutions can be recovered for unconstrained articulated bod-

ies. The distance-geometric robot model introduced in (Porta et al., 2005) was used in (Maric et al., 2021) and (Giamou et al., 2022) to produce Riemannian and convex optimization-based IK formulations, respectively. Our learning architecture adopts this distance-geometric paradigm to construct a graphical representation of the IK problem that can be leveraged by a GNN, maintaining the generalization capability inherent to conventional approaches. In contrast to conventional methods, however, our model outputs a distribution representing the IK solution set rather than individual solutions only.

4.2.2 Learning Inverse Kinematics

Jordan and Rumelhart show in (Jordan and Rumelhart, 1992) that the non-uniqueness of IK solutions presents a major difficulty for learning algorithms, which often yield erroneous models that "average" the nonconvex feasible set. However, Bullock et al. (Bullock et al., 1993) observe that relating end-effector and configuration *directions* instead of positions (i.e., working in the velocity space) ensures that the space of feasible inputs can be linearly approximated at each point along a trajectory, ensuring that their superposition is also feasible. Bócsi et al. (Bócsi et al., 2011) use an SVM to parameterize a quadratic program whose solutions match those of position-only IK for particular workspace regions. In computer graphics, Villegas et al. (Villegas et al., 2018) apply a recursive neural network model to solve a highly constrained IK instance for motion transfer between skeletons with different bone lengths. In this work, we show that a GNN-based IK solver allows a higher degree of generalization by capturing not only different link lengths, but also different numbers of DOF.

Recently, generative models have demonstrated the potential to represent the full set of IK solutions. A number of invertible architectures (Ardizzone et al., 2018; Kruse et al., 2021) have been able to successfully learn the feasible set for 2D kinematic chains. Generative adversarial networks (GANs) have been applied to learn the inverse kinematics and dynamics of an 8-DOF robot (Ren and Ben-Tzvi, 2020). The learned model is used to improve motion planning performance by sampling configurations constrained by link positions and (partial) orientations (Lembono et al., 2021). Ho et al. (Ho and King, 2022) have proposed a model that retrieves configurations reaching a target position by decoding *posture indices* for the closest position from a database of positions. A series of individual networks have been leveraged to sequentially generate distributions of joint values of a manipulator in an auto-regressive manner (Bensadoun et al., 2022). Finally, Ames et al. (Ames et al., 2022) describe IKFlow, a model that relies on normalizing flows to generate a distribution of IK solutions for a desired end-effector pose. Our architecture differs from previous work by allowing the learned distribution to generalize to a larger class of robots, conveniently removing the requirement of training an entirely new model from scratch for each specific robot.

4.2.3 Learning for Motion Planning

Prior applications of learning in motion planning include warm-starting optimization-based methods (Ichnowski et al., 2020), learning distributions for sampling-based algorithms (Khan et al., 2020; Ichter et al., 2018), directly learning a motion planner (Qureshi et al., 2020), and learning cost functions for joint grasp and motion optimization with diffusion models (Urain et al., 2023). In (Ichter et al., 2018),

Ichter et al. introduce a learning-based generative model, based on the conditional variational autoencoder (CVAE), to help solve various 2D and 3D navigation planning problems. The learned model is applied to sample the state space in a manner that is biased towards favourable regions (i.e., where a solution is more likely to be found). GGIK shares the same premise, to learn an initialization or sampler that improves upon the uniform sampling that is traditionally performed for both sampling-based motion planning and "assumption-free" IK. However, although our methodology is similar in some respects, IK introduces additional challenges from the perspective of learning. Specifically, for the IK problem, we aim to capture multiple solutions and to handle multiple manipulator structures.

4.2.4 Generative Models on Graphs

GGIK is a deep generative model of graphs. We provide a short review of generative models for graph representations, emphasizing recent deep learning-based methods, and point readers to (Liao, 2022) for a more extensive survey. Existing learning-based approaches typically make use of VAEs (Kipf and Welling, 2016), generative adversarial networks (GANs) (De Cao and Kipf, 2018), or deep autoregressive methods (Li et al., 2018). The authors of (Kipf and Welling, 2016) demonstrate how to extend the VAE framework to graph data. Since GGIK uses a graph-based CVAE, it builds upon the theoretical groundwork of the VGAE (variational graph autoencoder) family of models from (Kipf and Welling, 2016). We use a conditional variant of the VGAE and a more expressive node decoder that is parametrized with a graph neural network as opposed to a simple inner product edge decoder. In work proximal to our own, the authors of (Simm and Hernández-Lobato, 2020) present a CVAE model based on a distance-geometric representation for the molecular conformation problem. The generative model for molecular conformation maps atom and bond types into distances. We map partial sets of distances and positions to full sets of distances and positions, which is equivalent to solving a distance-based formulation of the IK problem. Although our overall learning architecture shares similarities, we are solving a different problem.

4.3 Preliminaries

Our approach casts IK as the problem of completing a partial graph of distances. We are interested in learning how to generate full graphs conditioned on partial graphs. In this section, we begin by introducing the general forward and inverse kinematics problems, and their solution sets, as they occur in robotic manipulation. This is followed by a description of the distance-geometric graph representation of manipulators used in this work, where nodes correspond to points on the robot and edges correspond to known inter-point distances. We then briefly introduce variational inference, which is the learning framework used in this work. Finally, we describe the feature representation that we employ for learning.

4.3.1 Forward and Inverse Kinematics

Most robotic manipulators are modelled as kinematic chains composed of revolute joints connected by rigid links. The joint angles can be arranged in a vector $\boldsymbol{\theta} \in \mathcal{C}$, where $\mathcal{C} \subseteq \mathbb{R}^n$ is known as the manipulator *configuration space*. Analogously, coordinates $\boldsymbol{\tau}$ that parameterize the task being performed constitute the *task space* $\mathcal{T}$. The *forward kinematics* function $\mathrm{FK} : \mathcal{C} \to \mathcal{T}$ maps joint angles $\boldsymbol{\theta}$ to task space coordinates

$$\mathrm{FK}(\boldsymbol{\theta}) = \boldsymbol{\tau} \in \mathcal{T}. \tag{4.1}$$

This relationship can be found in closed form using known structural information (e.g., joint screws (Murray et al., 2017) or Denavit-Hartenberg (DH) parameters (Hartenberg and Denavit, 1955)). We focus on the task space $\mathcal{T} := \mathrm{SE}(3)$ of 6-DOF end-effector poses. Instead of $\boldsymbol{\tau}$, we use the standard notation $\mathbf{T}$ for elements of $\mathrm{SE}(3)$.

The mapping $\mathrm{IK} : \mathcal{T} \to 2^{\mathcal{C}}$ defines the *inverse kinematics* of the robot, where $2^{\mathcal{C}}$ is the powerset of $\mathcal{C}$. In other words, the mapping IK is the inverse of the forward kinematic mapping in Eq. (4.1), connecting a target pose $\mathbf{T} \in \mathrm{SE}(3)$ to one or more feasible configurations $\boldsymbol{\theta} \in \mathcal{C}$. The solution to an IK problem is generally not unique (i.e., FK is not injective and therefore multiple feasible configurations exist for a single target pose $\mathbf{T}$). In this thesis, we consider the associated problem of determining this mapping for manipulators with $n > 6$-DOF (also known as *redundant* manipulators), where each end-effector pose corresponds to a set of configurations

$$\mathrm{IK}(\mathbf{T}) = \{\boldsymbol{\theta} \in \mathcal{C} \,|\, \mathrm{FK}(\boldsymbol{\theta}) = \mathbf{T}\} \tag{4.2}$$

that we refer to as the full set of IK solutions. The solution set in Eq. (4.2) cannot be expressed analytically for most redundant manipulators. Further, the infinite number of configurations cannot be recovered using local search methods. Therefore, most of the literature refers to finding even a single feasible configuration as "solving" inverse kinematics. Instead of finding individual configurations that satisfy the forward kinematics equations, we approximate the full solution set $\mathrm{IK}(\mathbf{T})$ itself as a learned conditional distribution.

4.3.2 Distance-Geometric Graph Representation of Robots

We eschew the common angle-based representation of the configuration space in favour of a *distance-geometric* model of robotic manipulators comprised of revolute joints (Porta et al., 2005). The distance-geometric approach allows us to represent a robot in configuration $\boldsymbol{\theta}$ with a complete weighted graph $G = (V, E, d)$. The weight function $d : E \to \mathbb{R}^+$ gives the distances between a collection of points p indexed by the vertices V:

$$d_{u,v} \triangleq d(\{u, v\}) = \|p(u) - p(v)\|. \tag{4.3}$$

Given the complete set of distances (i.e., the range of d), we can use multidimensional scaling to recover unique[1] coordinates of the points p, which can then be used to recover the joint angles $\boldsymbol{\theta}$. In (Maric

[1] Up to any Euclidean transformation of p, since distances are invariant to such a transformation.

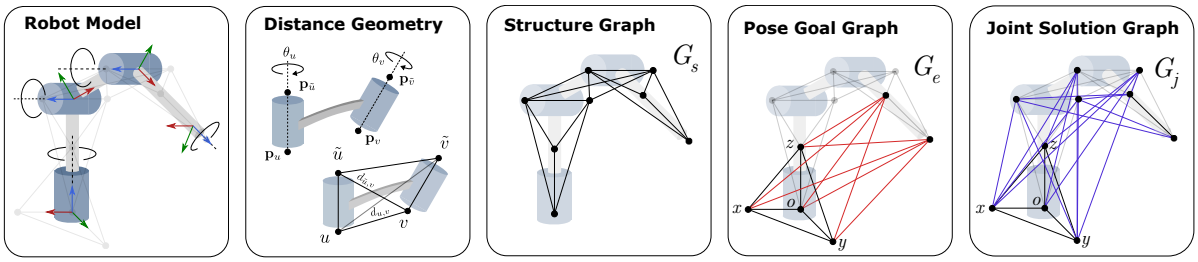


Figure 4.1: The process of defining an IK problem as an incomplete or *partial* graph $\widetilde{G}$ of inter-point distances and the associated IK solution as a complete graph G. (a) Conventional forward kinematics model parameterized by joint angles and joint rotation axes. (b) The point placement procedure for the distance-based description, first introduced in (Maric et al., 2021). (c) A structure graph of the robot based on inter-point distances. (d) Addition of distances (in red) specifying the end-effector pose, with auxiliary points to define the base coordinate system, that completes the IK problem description. (e) The distances in blue that define a specific joint configuration.

et al., 2021), it was shown that the restriction $\tilde{d} \triangleq d|_{\widetilde{E}}$ of the distance function d to $\widetilde{E} \subset E$, represented by the incomplete or *partial* graph $\widetilde{G} \triangleq (V, \widetilde{E}, \tilde{d})$, may be used to recover the points p corresponding to configurations that adhere to some set of geometric constraints. This is achieved by solving the distance geometry problem (DGP):

Distance Geometry Problem (Liberti et al. (2014)). *Given an integer $D > 0$, a set of vertices V, and a simple graph $\widetilde{G} = (V, \widetilde{E}, \tilde{d})$ whose edges $\{u, v\} \in \widetilde{E}$ are assigned non-negative weights $\tilde{d} : \{u, v\} \mapsto d_{u,v} \in \mathbb{R}_+$, find a function $p : V \to \mathbb{R}^D$ such that the Euclidean distances between neighbouring vertices match their edges' weights (i.e., $\forall \{u, v\} \in \widetilde{E},\ \|p(u) - p(v)\| = d_{u,v}$).*

Crucially, the partial graph $\widetilde{G} = (V, \widetilde{E}, \tilde{d})$ can be constructed with $\widetilde{E} \subset E$ corresponding to distances $\tilde{d}$ determined by an end-effector pose $\mathbf{T}$ and the robot's structure. In this case, any set of points recovered by solving the distance geometry problem for a partial graph $\widetilde{G}$ corresponds to a particular IK solution $\boldsymbol{\theta} \in \mathrm{IK}(\mathbf{T})$.

The generic procedure for constructing $\widetilde{G}$ is demonstrated for a simple manipulator in Fig. 4.1. We visualize the point placement step for a pair of consecutive joints in Fig. 4.1b. Two pairs of points labeled by vertices $u, \tilde{u}$ and $v, \tilde{v}$ are attached to the rotation axes of neighbouring joints at a unit distance. The edges associated with every combination of points are then weighted by their respective distances, which are defined solely by the link geometry. We build the *structure graph* $G_s = (V_s, E_s, d|_{E_s})$ shown in Fig. 4.1c by repeating this process for every pair of neighbouring joints. The resulting set of vertices V_s and distance-weighted edges E_s, shown in Fig. 4.1c, describe the rigid geometry of the robot because the edge weights are invariant to feasible motions of the robot (i.e., they remain constant in spite of changes to the configuration $\boldsymbol{\theta}$). In order to uniquely specify points with known positions (i.e., end-effectors) in terms of distances, we define the *base vertices* $V_b = \{o, x, y, z\}$, where o and z are the vertices in V_s associated with the base joint. Setting the distances weighting the edges E_b such that the points in V_b form a coordinate frame with o as the origin, we specify the edges E_p weighted by distances between vertices in $V_p \subset V_s$ associated with the end-effector and the base vertices V_b. The resulting subgraph $G_e = G_b \cup G_p$, where we define the union of two graphs as

$G_A \cup G_B \equiv (V_A \cup V_B, E_A \cup E_B, d|_{E_A \cup E_B})$, is shown in Fig. 4.1d. G_e uniquely specifies an end-effector pose under the assumption of unconstrained rotation of the final joint, while $\widetilde{G} = G_s \cup G_e$ is the partial graph that uniquely specifies the associated IK problem. Finally, the *solution graph* $G_j = (V_j, E_j)$ shown in Fig. 4.1e contains the remaining distances that define a joint configuration $\boldsymbol{\theta}$. The full graph is defined as $G = G_s \cup G_e \cup G_j$ and uniquely specifies a joint configuration solution to the associated IK problem (Maric et al., 2021). Note that the partial graphs $\widetilde{G}$ of any two revolute manipulators with the same number of joints and goal pose differ only in the edge weights (i.e. distances) $\tilde{d}$ associated with $\widetilde{E}$, which are a function of the kinematic parameters of the two robots.

4.3.3 Feature Representation for Learning

To train our GNN-based model, we need to choose a representation for the partial graphs $\widetilde{G}$, representing the IK problem, and complete graphs G, representing associated solution configurations. For the complete graph G, we define the GNN node (vertex) features as a combination of point positions $\mathbf{P} = [\mathbf{p}_0, \ldots, \mathbf{p}_N]^\intercal \in \mathbb{R}^{N \times D}$, where $D \in \{2, 3\}$ is the dimension of the workspace, and general features $\mathbf{H} = [\mathbf{h}_0, \ldots, \mathbf{h}_N]^\intercal \in \mathbb{R}^{N \times L}$, where each $\mathbf{h}_i$ is a feature vector containing extra information about the node. For GGIK, we use a three-dimensional ($L = 3$) one-hot-encoded vector for $\mathbf{h}_i$ that indicates whether the node is part of the base coordinate system, a general joint or link, or the end-effector. The edge features are simply the inter-point distances. We define $e_{ij} \in \mathbb{R}$ as the edge feature between node i and j. For the partial graph $\widetilde{G}$, only the nodes defining the base vertices and the end-effector have known positions (i.e., the respective positions associated with nodes V_b and V_p). The remaining unknown node positions are set to zero. We denote $\tilde{\mathbf{P}} \in \mathbb{R}^{N \times D}$ as the zero-padded node features of the partial graph. The partial graph shares the same general features $\mathbf{H}$ as the complete graph, given that we know which part of the robot each node belongs to in advance. The edge features of the partial graph are the known inter-point distances, with the unknown distances initialized to zero. The distances that define the structure graph and the end-effector pose (i.e., the respective distances associated with edges E_s and E_e) are known. The possible choices of node and edge features offers significant flexibility. For example, we could add an extra class or dimension to the general feature vector to indicate nodes that represent obstacles in the task space, or even more abstract features such as 3D shape information.

4.4 Learning to Generate Inverse Kinematics Solutions

GGIK is a learning-based IK solver that is, crucially, capable of producing multiple diverse solutions while also generalizing across a family of kinematic structures. We consider the problem of modelling complete graphs corresponding to IK solutions given partial graphs that define the problem instance (i.e., the robot's geometric information and the task space goal pose). Intuitively, we would like our network to map or "complete" partial graphs into full graphs. Since multiple or infinite solutions may exist for a single end-effector pose and robot structure, a single partial graph may be associated with multiple or even infinite valid complete graphs. We interpret the learning problem through the lens

of generative modelling and treat the solution space as a multimodal distribution conditioned on a single problem instance. By sampling from this distribution, we can generate diverse solutions that approximately cover the space of feasible configurations.

A visual overview of the training procedure is shown in Fig. 4.2. At its core, GGIK is a CVAE model (Sohn et al., 2015b) that parameterizes the conditional distribution $p(G \mid \widetilde{G})$ using GNNs. By introducing an unobserved stochastic latent variable $\mathbf{Z}$, our generative model is defined as

$$p_\gamma(G \mid \widetilde{G}) = \int p_\gamma(G \mid \widetilde{G}, \mathbf{Z})\, p_\gamma(\mathbf{Z} \mid \widetilde{G})\, d\mathbf{Z}, \tag{4.4}$$

where $p_\gamma(G \mid \widetilde{G}, \mathbf{Z})$ is the conditional likelihood of the full graph, $p_\gamma(\mathbf{Z} \mid \widetilde{G})$ is the prior, and γ are the learnable generative parameters. The conditional likelihood is given by

$$\begin{aligned} p_\gamma(G \mid \widetilde{G}, \mathbf{Z}) &= \prod_{i=1}^{N} p_\gamma(\mathbf{p}_i \mid \widetilde{G}, \mathbf{z}_i), \quad \text{with} \\ p_\gamma(\mathbf{p}_i \mid \widetilde{G}, \mathbf{z}_i) &= \mathcal{N}(\mathbf{p}_i \mid \boldsymbol{\mu}_i, \boldsymbol{\Sigma}_i), \end{aligned} \tag{4.5}$$

where $\mathbf{Z} = [\mathbf{z}_0, \ldots, \mathbf{z}_N]^\intercal \in \mathbb{R}^{N \times F}$ are the latent embeddings of each node, and $\{\boldsymbol{\mu}_i\}_{i=i}^N$ and $\{\boldsymbol{\Sigma}_i\}_{i=i}^N$ are the respective predicted means and covariances of the distribution of node positions. We parametrize the likelihood distribution with a GNN decoder, in other words, $\{\boldsymbol{\mu}_i\}_{i=i}^N$ and $\{\boldsymbol{\Sigma}_i\}_{i=i}^N$ are the outputs of $\text{GNN}_{dec}(\widetilde{G}, \mathbf{Z})$. The GNN decoder propagates messages and updates the nodes at each intermediate layer and, at the final layer, outputs the predicted distribution parameters of all node positions. In practice, for the input node features of $\text{GNN}_{dec}(\cdot)$, we concatenate the latent embeddings $\mathbf{Z}$ with the respective position features $\tilde{\mathbf{P}}$ of the original partial graph $\widetilde{G}$ and the general features $\mathbf{H}$. The latent edge features are set as the known distances or the zero-initialized unknown distances, e_{ij}. We follow the common practice of only learning the mean of the decoded Gaussian likelihood distribution and use a fixed diagonal covariance matrix $\boldsymbol{\Sigma}_i = \sigma_\epsilon \mathbf{I}$, where σ_ϵ is set to an arbitrarily small value for all i (Doersch, 2016). The prior distribution is given by

$$\begin{aligned} p_\gamma(\mathbf{Z} \mid \widetilde{G}) &= \prod_{i=1}^{N} p_\gamma(\mathbf{z}_i \mid \widetilde{G}), \quad \text{with} \\ p_\gamma(\mathbf{z}_i \mid \widetilde{G}) &= \sum_{k=1}^{K} \pi_{k,i}\, \mathcal{N}\left(\mathbf{z}_i \mid \boldsymbol{\mu}_{k,i}, \text{diag}(\boldsymbol{\sigma}_{k,i}^2)\right). \end{aligned} \tag{4.6}$$

Here, we parameterize the prior as a Gaussian mixture model with K components. Each Gaussian is parameterized by a mean $\boldsymbol{\mu}_k = \{\boldsymbol{\mu}_{k,i}\}_{i=1}^N$ and diagonal covariance $\boldsymbol{\sigma}_k = \{\boldsymbol{\sigma}_{k,i}\}_{i=1}^N$. The mixture model is parametrized by the mixing coefficients $\boldsymbol{\pi}_k = \{\pi_{k,i}\}_{i=1}^N$, where $\sum_{k=1}^K \pi_{k,i} = 1,\ \forall\, i = 1, ..., N$. We chose a mixture model to have an expressive prior capable of capturing the latent distribution of multiple solutions. We parameterize the prior distribution with a multi-headed GNN encoder $\text{GNN}_{prior}(\widetilde{G})$ that outputs parameters $\{\boldsymbol{\mu}_k, \boldsymbol{\sigma}_k, \boldsymbol{\pi}_k\}_{k=1}^K$.

For GGIK, the goal of learning is to maximize the marginal likelihood or evidence of the full graphs

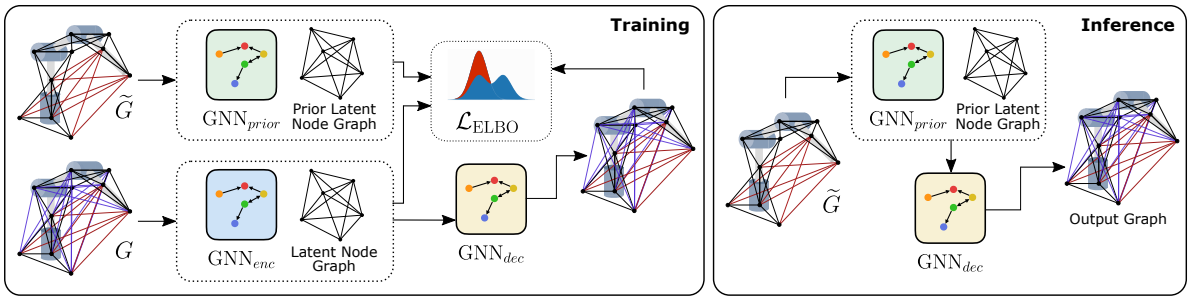


Figure 4.2: Our GGIK solver is based on the CVAE framework. GNN_{enc} encodes a complete graph representation of a manipulator into a latent graph representation and GNN_{dec} "reconstructs" it. The prior network, GNN_{prior}, encodes the partial graph into a latent embedding that is near the embedding of the full graph. At test time, we decode the latent embedding of a partial graph into a complete graph to generate a solution.

as shown in Eq. (4.4). We do this by maximizing the ELBO

$$\mathcal{L} = \mathbb{E}_{q_\phi(\mathbf{z}|G)}[\log p_\gamma(G \mid \widetilde{G}, \mathbf{Z})] - \beta \, \text{KL}(q_\phi(\mathbf{Z} \mid G) \,||\, p_\gamma(\mathbf{Z} \mid \widetilde{G})), \tag{4.7}$$

where $KL(\cdot||\cdot)$ is the Kullback-Leibler (KL) divergence, β is a weighting hyperparameter, and $q_\phi(\mathbf{Z} \mid G)$ is the inference model with learnable parameters ϕ, defined as

$$\begin{aligned} q_\phi(\mathbf{Z} \mid G) &= \prod_{i=1}^{N} q_\phi(\mathbf{z}_i \mid G), \quad \text{with} \\ q_\phi(\mathbf{z}_i \mid G) &= \mathcal{N}(\mathbf{z}_i \mid \boldsymbol{\mu}_i, \text{diag}(\boldsymbol{\sigma}_i^2)). \end{aligned} \tag{4.8}$$

As with the prior distribution, we parameterize the inference distribution with a multi-headed GNN encoder, $\text{GNN}_{enc}(G)$, that outputs parameters $\{\boldsymbol{\mu}_i\}_{i=1}^N$ and $\{\boldsymbol{\sigma}_i\}_{i=1}^N$. The inference model is an approximation of the intractable true posterior $p(\mathbf{Z} \mid G)$. We note that the resulting ELBO objective in Eq. (4.7) is based on an expectation with respect to the inference distribution $q_\phi(\mathbf{Z} \mid G)$, which itself is based on the parameters ϕ. Since we restrict $q_\phi(\mathbf{Z} \mid G)$ to be a Gaussian variational approximation, we can use stochastic gradient descent (i.e., Monte Carlo gradient estimates) via the reparameterization trick (Kingma and Welling, 2014) to optimize the lower bound with respect to parameters γ and ϕ.

At test time, given a goal pose and the manipulator's geometric information encapsulated in a partial graph $\widetilde{G}$, the prior network, GNN_{prior}, encodes the partial graph as a latent distribution $p_\gamma(\mathbf{Z} \mid \widetilde{G})$. During training, the distribution $p_\gamma(\mathbf{Z} \mid \widetilde{G})$ is optimized to be simultaneously near multiple encodings of valid solutions by the KL divergence term in Eq. (4.7). We can sample this multimodal distribution $\mathbf{Z} \sim p_\gamma(\mathbf{Z} \mid \widetilde{G})$ as many times as needed, and subsequently decode all of the samples with the decoder network GNN_{dec} to generate IK solutions represented as complete graphs. This procedure can be done quickly and in parallel on the GPU. We provide a more detailed explanation of the sampling procedure in Section 4.6.

4.5 Equivariant Network Architecture

In this section, we discuss the choice of architecture for networks GNN_{dec}, GNN_{enc}, and GNN_{prior}. During inference, our model maps zero-padded partial point sets $\tilde{\mathbf{P}}$ to full point sets $\mathbf{P}$, $f : \mathbb{R}^{N\times D} \to \mathbb{R}^{N\times D}$, where f is a combination of networks GNN_{prior} and GNN_{dec} applied sequentially. The point positions of the nodes in the distance geometry problem contain underlying geometric relationships and symmetries that we would like to preserve with our choice of architecture. Most important, the point sets are *equivariant* to the Euclidean group $\text{E}(n)$ of rotations, translations, and reflections. Let $S : \mathbb{R}^{N\times D} \to \mathbb{R}^{N\times D}$ be a transformation consisting of some combination of rotations, translations and reflections on the initial zero-padded partial point set $\tilde{\mathbf{P}}$. Then, there exists an equivalent transformation $\text{T} : \mathbb{R}^{N\times D} \to \mathbb{R}^{N\times D}$ on the complete point set $\mathbf{P}$ such that

$$f(S(\tilde{\mathbf{P}})) = \text{T}(f(\tilde{\mathbf{P}})). \tag{4.9}$$

As a specific example, if we translate and rotate our initial partial graph, which defines the IK problem instance, then the complete graph, which defines the solution, will be equivalently translated and rotated. To leverage this geometric prior or structure in the data, we use $\text{E}(n)$-equivariant graph neural networks (EGNNs) (Satorras et al., 2021) for GNN_{dec}, GNN_{enc}, and GNN_{prior}. The EGNN layer splits up the node features into an equivariant coordinate or position-based part and a non-equivariant part. We treat the positions $\mathbf{P}$ and $\tilde{\mathbf{P}}$ as the equivariant portion and the general features $\mathbf{H}$ as non-equivariant. As an example, for a single EGNN layer l and a single node i, the update equations for the equivariant node features are

$$\begin{aligned} \mathbf{p}_i^{l+1} &= \mathbf{p}_i^l + C\sum_{j\neq i}(\mathbf{p}_i^l - \mathbf{p}_j^l)\phi_x(\mathbf{m}_{ij}) \\ \mathbf{m}_{ij} &= \phi_e(\mathbf{h}_i^l, \mathbf{h}_j^l, \|\mathbf{p}_i^l - \mathbf{p}_j^l\|^2, e_{ij}), \end{aligned} \tag{4.10}$$

where we define $\mathbf{m}_{ij} \in \mathbb{R}^d$ as a message of dimension d from node j to i. The position of each node is updated by the weighed sum of relative differences between the node itself and all other nodes. The function $\phi_x : \mathbb{R}^d \to \mathbb{R}$ provides the weights for each term in the sum. The function ϕ_e represents a general edge operation. C is a normalizing factor based on the number of elements. Similarly, the update equations for the non-equivariant node features are

$$\begin{aligned} \mathbf{h}_i^{l+1} &= \phi_h(\mathbf{h}_i^l, \mathbf{m}_i) \\ \mathbf{m}_i &= \sum_{j\neq i}\mathbf{m}_{ij}, \end{aligned} \tag{4.11}$$

where ϕ_h represents a node update operation and $\mathbf{m}_i \in \mathbb{R}^d$ is the sum of all messages received by node i. We model ϕ_x, ϕ_h, and ϕ_e with multilayer perceptrons (MLPs). For all learned distributions, we map the concatenated equivariant and non-equivariant features of each node from the last layer to the respective distribution parameters. For more details about the model and a proof of the equivariance

property, we refer readers to (Satorras et al., 2021). We present ablation studies on the use of the EGNN network architecture in Section 6.4, demonstrating its importance to our approach.

4.6 Sampling Inverse Kinematics Solution

We summarize the full sampling procedure in Algorithm 1. We first sample N sets of latent graphs $\mathbf{Z}_N$ from a Gaussian distribution $p_\gamma(\mathbf{Z} \mid \widetilde{G})$ parameterized by outputs from the prior encoder GNN_{prior}. The samples $\mathbf{Z}_N$ are then passed to the decoder network GNN_{dec}, producing N point sets $\mathbf{P}_N$ sampled from the distribution $p_\gamma(\mathbf{P} \mid \widetilde{G}, \mathbf{Z})$. Next, N joint configuration (i.e., joint angle) estimates $\boldsymbol{\theta}_N$ are recovered from the point sets $\mathbf{P}_N$ using a simple geometric procedure described in (Maric et al., 2021). Finally, we select the M best configurations $\boldsymbol{\theta}^*$ according to some criteria. We filter to find the M configurations with the lowest error with respect to the goal pose $\mathbf{T}_{goal}$, where we define the error as

$$e = \left\| \text{Log}\left(\mathbf{T}^{-1}\mathbf{T}_{goal}\right) \right\|_2 .$$

Note that the configuration recovery and filtering steps may be performed in parallel for all N configurations, in which case they require approximately 5 ms of computation time on a laptop CPU for $N = 128$ and $M = 32$. For more details on the total computation time required, see Section 4.8.4.

4.7 Learning Implementation Details

In this section, we first cover how we generate our training dataset, followed by additional training and network architecture details.

4.7.1 Training Data

Acquiring training data is convenient—any valid manipulator configuration can be used for training. In our experiments, we use training data from two different datasets. The first dataset comprises of IK problems for existing commercial manipulators with typical kinematic structures. The second dataset contains IK problems for manipulators with randomly generated kinematic structures. In both datasets, IK "instances" were generated by uniformly sampling a set of joint angles and using forward

Algorithm 1: GGIK

Parameters : $\widetilde{G}, \mathbf{T}_{goal}, N, M$
Result: Solution configurations with the lowest pose error $\boldsymbol{\theta}^* \in \mathbb{R}^{M \times n_{joints}}$.
$\mathbf{Z}_N \sim p_\gamma(\mathbf{Z} \mid \widetilde{G})$ ▷ Sample N latents $\mathbf{Z}$ from GNN_{prior}.
$\mathbf{P}_N \sim p_\gamma(\mathbf{P} \mid \widetilde{G}, \mathbf{Z}_N)$ ▷ Get N solutions via GNN_{dec}.
$\boldsymbol{\theta}_N \leftarrow \text{fromPoints}(\mathbf{P}_N)$ ▷ Recover N configurations.
$\boldsymbol{\theta}^* \leftarrow \text{selectSol}(\mathbf{T}_{goal}, \boldsymbol{\theta}_N, M)$ ▷ Choose best M.

kinematics to compute the associated end-effector pose. The end-effector pose is then used as the goal for IK, which ensures that all problems are feasible (i.e., have at least one solution).

Commercial Manipulator Dataset This dataset contains IK problems that are uniformly distributed over five different robots with varying kinematic properties. Specifically, we chose the Universal Robots UR10, Schunk LWA4P, Schunk LWA4D, KUKA IIWA and the Franka Emika Panda robots. The UR10 and LWA4P are 6-DOF robots with up to 8 discrete IK solutions for a given goal pose. On the other hand, the IIWA, LWA4D and the Panda robots all share a redundant kinematic structure with 7-DOF and, by extension, an infinite solution set to any feasible IK problem.

Randomized Manipulator Dataset This dataset contains a large number of randomly generated pairs of robots and associated IK problems. A unique robot was created for each problem. A key consideration for any randomization scheme used to produce such a dataset is that most real manipulators exist within a narrow class of kinematic structures. For example, the rotation axes of consecutive joints for many manipulators are either parallel or perpendicular. Furthermore, manipulators generally do not feature more than two sequential joints with parallel rotation axes, and these joints are generally displaced along a direction orthogonal to their axis of rotation. Therefore, we generate samples by randomizing DH parameters following the original convention of (Hartenberg and Denavit, 1955) using the procedure in Algorithm 2. In the absence of an informative prior, we sample from a uniform distribution that can be discrete (e.g., $\mathcal{U}\{a, b, c, \cdots\}$) or continuous within a range (e.g., $\mathcal{U}(a, b)$). We select the parameters of this distribution such that the resulting structures include the UR10, KUKA, LWA4D and LWA4P robot, although it is improbable that the exact parameters of these robots will be sampled. The resulting problems are constrained to robots with structures similar, but never identical to, the commercial robots used in the first dataset.

4.7.2 Additional Training and Network Architecture Details

We provide some brief and practical advice for training GGIK based on our experimental observations. Picking a network architecture that captures the underlying geometry and equivariance of the problem improved the performance of the models by a large margin. We discuss this in in further details in Section 4.8.6. We used 16 components for the Gaussian mixture model parameters from GNN_{prior}. The MLPs used for message passing in the EGNN: ϕ_e, ϕ_h, and ϕ_x all had two hidden layers of dimension 128. In order to increase the accuracy of our models, we found it particularly effective to lower the learning rate when the training loss stagnated. We empirically observed that increasing the number of layers used in the GNNs up to five improved the overall performance. There was very little performance improvement beyond five layers. We found that using a mixture model for the distribution of the prior encoder lead to more accurate results. We hypothesize that a more expressive prior distribution can better fit multiple encoded IK solutions when conditioned on a single goal pose. Finally, we found that lowering the weight of the reconstruction loss or likelihood contribution from nodes that belonged to the *structure graph* (see Section 4.3.2) led to overall lower losses. We hypothesize that by

Algorithm 2: Randomized DH parameters

Parameters : $d_{min}, d_{max}, a_{min}, a_{max}$
Result: Randomized robot
$\alpha_0 \sim \mathcal{U}\{-\pi/2, \pi/2\}$ ▷ First set of DH parameters.
$a_0, \theta_0 \leftarrow 0$
$d_0 \sim \mathcal{U}\,(0, d_{max})$
for $k = 1, \ldots n-2$ **do**
  $\theta_k, a_k, d_k \leftarrow 0$
  ▷ Limit sequential parallel rotation axes.
  if $\alpha_{k-1} = 0$ *and* $\alpha_{k-2} = 0$ **then**
    $\alpha_k \sim \mathcal{U}\{-\pi/2, \pi/2\}$
  else
    $\alpha_k \sim \mathcal{U}\{0, -\pi/2, \pi/2\}$
  end
  ▷ Ensure rotation axes are coplanar.
  if $\alpha_k \neq 0$ **then**
    $d_k \sim \mathcal{U}\,(0, d_{max})$
  else
    $a_k \sim \mathcal{U}\,(a_{min}, a_{max})$
  end
end
$a_{n-1}, \theta_{n-1}, \alpha_{n-1} \leftarrow 0$ ▷ End-effector frame.
$d_{n-1} \sim \mathcal{U}\,(0, d_{max})$

lowering the weight of the reconstruction loss for the structure graph, we encourage the optimization procedure to focus on reconstructing nontrivial parts of the solution, which then avoids local minima that overemphasize learning the structure graph G_s. We used SiLU nonlinearities (Elfwing et al., 2018) with layer normalization (Ba et al., 2016) for all networks. We used the Adam optimizer with weight decay (Loshchilov and Hutter, 2019) and a learning rate of $3e-4$. The smaller models were each trained for a total of approximately 16 hours on 512,000 IK problems and the larger models were trained for approximately 90 hours on 2,560,000 IK problems. All models were trained for 360 epochs. For more implementation details, we invite interested readers to refer to our open-source code repository.

4.8 Experiments

In this section, we present results from a series of seven experiments, where we studied the behaviour and properties of GGIK. We evaluated (i) GGIK's capability to learn accurate solutions and generalize within a class of manipulator structures (Section 4.8.1), (ii) GGIK's ability to generalize to unseen manipulators (Section 4.8.2), (iii) GGIK's ability to produce multiple solutions (Section 4.8.3), (iv) the scaling of the computation times of GGIK (Section 4.8.4), (v) whether GGIK can effectively initialize local numerical IK solvers (Section 4.8.5), (vi) the importance of our choice of learning architecture (Section 4.8.6), and (vii) whether or not GGIK can learn constrained distributions that account for joint limits (Section 4.8.7). All experiments were performed on a computer with an AMD Ryzen Threadrip-

per 2920X 12-Core Processor CPU and an NVIDIA Quadro RTX 8000 GPU.

4.8.1 Accuracy

We evaluated the accuracy of GGIK for a variety of existing commercial manipulators featuring different structures and numbers of joints: the Kuka IIWA, Schunk LWA4D, Schunk LWA4P, Universal Robots UR10, and Franka Emika Panda. We also assessed the performance of GGIK for a hypothetical revolute robot with 12-DOF, to demonstrate the feasibility of scaling to more complex manipulators.

Individual Robots

We trained a bespoke model for each of the robots on 512,000 data points (IK problems), which we generated using the procedure described in Section 4.7.1. We evaluated these models on 2,000 randomly-generated IK problem instances per robot and report the error between the computed and goal end-effector poses. The orientation error is the magnitude of the angular difference in orientation, measured using the axis-angle representation. The results in Table 4.1 show that GGIK generates solutions with a mean position error of under 6 mm and a mean orientation error of under 0.4 degrees. Moreover, the first (Q_1) and third (Q_3) quartiles of the error distributions, as well as the maximum error values, show that the majority of sampled configurations produced end-effector poses with errors that do not substantially deviate from the mean. Overall, these results indicate that, when trained on a particular manipulator, GGIK has the ability to produce solutions with an accuracy sufficient for common manipulation tasks such as grasping, pushing, and obstacle avoidance.

As a benchmark, we trained a comparably sized network on 512,000 UR10 poses using the distal teaching (DT) method of (von Oehsen et al., 2020). The DT network was only able to achieve a mean position error of 43.5 mm and a mean orientation error of 15.3 degrees. Most important, this network maps an end-effector pose to a single fixed solution and does not model the entire solution set—multiple calls to the network will always produce the same solution. We also compare the performance of GGIK with results reported from the IKFlow (Ames et al., 2022) and IKNet (Bensadoun et al., 2022) models on the Franka Emika Panda manipulator. GGIK achieved better overall accuracy than IKFlow with approximately 2 mm lower translational error and 2 degrees lower rotational error on average. It is also important to note that the IKFlow model was trained on 2.5 million problems per robot, roughly five times more than GGIK (see Table 4.1). GGIK achieves significantly better accuracy than IKNet, with approximately 25 mm less translational error on average. The authors of IKNet did not report on rotational errors or the size of their training set.

Robot	Err. Pos. [mm]					Err. Rot. [deg]				
	mean	min	max	Q_1	Q_3	mean	min	max	Q_1	Q_3
KUKA	4.6	2.4	7.2	3.7	5.5	0.4	0.2	0.5	0.3	0.4
LWA4D	4.3	1.3	8.1	3.0	5.4	0.4	0.1	0.6	0.3	0.5
LWA4P	3.8	1.6	6.5	2.9	4.7	0.3	0.1	0.4	0.2	0.4
Panda	5.8	1.6	11.1	4.0	7.4	0.4	0.1	0.7	0.3	0.5
Panda with IKFlow (Ames et al., 2022)	7.7	-	-	-	-	2.8	-	-	-	-
Panda with IKNet (Bensadoun et al., 2022)	31.0	-	-	13.5	48.6	-	-	-	-	-
UR10	5.5	2.7	8.6	4.4	6.6	0.3	0.1	0.5	0.2	0.4
UR10 with DT (von Oehsen et al., 2020)	43.5	0.7	1124.7	16.2	45.9	15.3	0.2	177.7	5.7	18.3
12-DOF	15.3	4.6	25.4	11.5	19.2	0.7	0.1	1.3	0.5	1.0

Table 4.1: Performance of GGIK instances on 2,000 randomly generated IK problems per individual robotic manipulator types. A separate GGIK model was trained for each of the five robots. For GGIK, taking 32 samples from the learned distribution, the error statistics are presented as the mean and mean minimum and maximum error per problem, as well as the two quartiles of the distribution. We include baseline results from distal teaching (DT) (von Oehsen et al., 2020), IKFlow (Ames et al., 2022) and IKNet (Bensadoun et al., 2022). Dashed results were not provided by previous work. For the DT baseline, we calculate the statistics directly over all 2,000 problems.

	Err. Pos. [mm]					Err. Rot. [deg]				
Robot	mean	min	max	Q_1	Q_3	mean	min	max	Q_1	Q_3
KUKA	4.9	2.4	7.8	3.9	5.9	0.4	0.2	0.6	0.3	0.5
LWA4D	4.8	2.1	7.9	3.7	5.8	0.4	0.2	0.6	0.4	0.5
LWA4P	6.2	3.5	9.0	5.2	7.1	0.5	0.3	0.6	0.4	0.5
Panda	6.3	2.8	10.7	4.9	7.7	0.5	0.2	0.7	0.4	0.6
UR10	9.3	6.9	11.5	8.6	10.0	0.5	0.4	0.7	0.5	0.6

Table 4.2: Performance of GGIK on 10,000 randomly generated IK problems (2,000 per robot) for a single model trained on five different robotic manipulators. Taking 32 samples from the learned distribution, the error statistics are presented as the mean and mean minimum and maximum error per problem, as well as the two quartiles of the distribution. Note that all solutions were produced by a *single* model.

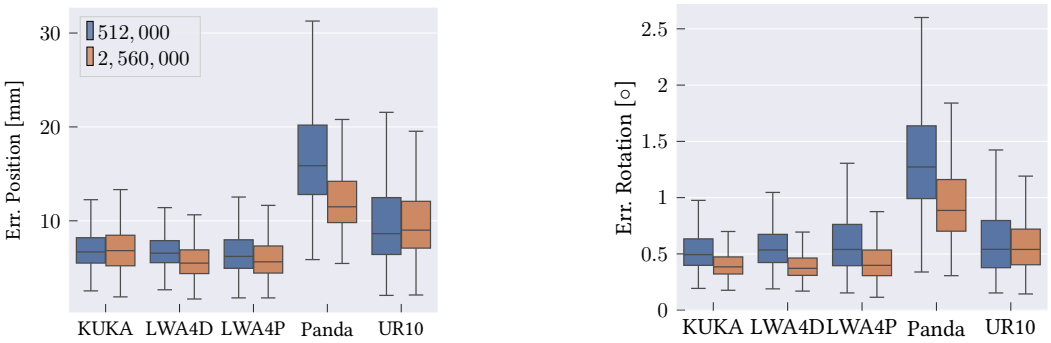
Multiple Robots

Next, we trained a single instance of GGIK on a total of 2,560,000 IK problems uniformly distributed over all five manipulators, that is, a single model for multiple robots. The model was evaluated on 10,000 randomly generated feasible IK problems (2,000 per robot). The error rates in Table 4.2 suggest that the proposed approach can be used to generate solutions in a variety of practical applications. The results shown in Fig. 4.3 indicate that by simply increasing the number of samples, the gap in solution accuracy between the various manipulators is reduced.

4.8.2 Generalization

In order to determine the capacity of GGIK to generalize to unseen manipulator structures, we evaluated the accuracy of a trained model on robots outside of the training set. To this end, we generated a dataset of inverse kinematics problems for manipulators with randomized kinematic structures, following the procedure outlined in Section 4.7.1. The training set consists of 4,096,000 inverse kinematics problems, each for a unique 6- or 7-DOF robot generated using randomized DH parameters (i.e., no two robots in the dataset are identical). We randomly selected batches of 128 robots during training, independent of their DOF and structure. The robots and procedure used in evaluation are identical to the experiment presented in Section 4.8.1. Each line in Table 4.3 displays the results from 2,000 IK problems solved by the trained model for a particular robot.

Our learned model was able to generalize reasonably well across all robots, reaching a mean position error of approximately 2 to 4 cm and a mean orientation error of 2 degrees. In contrast to the model trained on specific robots, discussed in Section 4.8.1, this model exchanges accuracy for the capability to generate approximate solutions for a much larger variety of robots. While the accuracy of this model is not sufficient alone for some practical use cases, the results in Section 4.8.5 suggest that the approximate solutions it provides can be refined by a few iterations of a local optimization procedure.



(a) Position error

(b) Rotation error

Figure 4.3: Box-and-whiskers plots comparing the accuracy for identical models trained on datasets containing multiple robots distributed over 512,000 and 2,560,000 datapoints, respectively.

Robot	Err. Pos. [mm]					Err. Rot. [deg]				
	mean	min	max	Q_1	Q_3	mean	min	max	Q_1	Q_3
KUKA	39.7	11.4	75.5	27.8	50.4	2.5	0.5	4.6	1.7	3.3
Lwa4d	35.7	10.2	68.3	25.0	45.5	2.3	0.5	4.3	1.6	3.1
Lwa4p	22.5	7.3	40.4	16.3	28.2	1.5	0.4	2.6	1.0	1.9
UR10	46.5	22.1	69.7	37.9	55.3	2.0	0.6	3.5	1.4	2.5

Table 4.3: Performance of GGIK on manipulators not seen during training. We train on randomly generated manipulators of DOF 6 and 7, and test on actual commercial robots, which were not seen in the dataset. Taking 50 samples from the learned distribution, the error statistics are presented as the mean and mean minimum and maximum error per problem, as well as the two quartiles of the distribution.

4.8.3 Multimodality

In addition to providing accurate solutions, GGIK is able to uniformly sample an approximation of the set of feasible configurations for a given goal pose. We qualitatively demonstrate this capability by plotting pairs of joint angle variables for multiple solutions sampled from GGIK for a single query. An example of this visualization procedure for 1,000 samples for the Kuka arm is displayed in Fig. 4.4. Each of the $\binom{7}{2} = 21$ plots in the symmetric upper and lower triangles of the array in Fig. 4.4 contains the sampled joint angles for a pair θ_i and θ_j. Since each pair contains samples from the torus $S^1 \times S^1$, the $x-$ and $y-$ axes "wrap around" at $\pi \equiv -\pi$. The rendered hue of each sample is based on its pose error, which is defined as the sum of the Euclidean distance in metres and the angular distance in radians. Darker samples indicate less accurate solutions. The continuous and orderly curves produced by accurate solutions demonstrate that GGIK is able to learn a relatively complex distribution over a large, varied solution set.

Since the UR10 is a (non-redundant) 6-DOF manipulator, we expect the same visualization for this arm to consist of discrete clusters. Fig. 4.5 demonstrates that this is indeed the case, with an interesting phenomenon occurring for this particular goal pose: the less accurate samples from GGIK appear to be attempts at *interpolating* between high accuracy clusters (i.e., those with lighter colouring). This does not occur for all poses sampled, and a quantitative clustering experiment confirms that GGIK is in fact learning a few accurate clusters for most poses.

Using 32 GGIK samples for each of 2,000 random goal poses, we computed a density-based spatial clustering of applications with noise (DBSCAN) (Schubert et al., 2017) using the product metric of the chordal distance for S^1. DBSCAN was selected because it supports data on topological manifolds and has very few parameters to tune. Fig. 4.6 contains histograms displaying the distribution of the number of clusters recovered by DBSCAN with a variety of values for the radius parameter ϵ. In all three cases, the minimum cluster size was set to one in order to allow for clusters containing a single sample. Ideally, we would like to see the number of clusters concentrated around eight, which is the maximum number of solutions for the UR10. In Fig. 4.6a, ϵ is small and noisy clusters are erroneously separated. However, the similarity between the distributions in Fig. 4.6b and Fig. 4.6c indicates that this is a fairly stable clustering procedure. Indeed, in both histograms there are very few poses for which GGIK produced greater than eight clusters, while the median number of clusters is four. We note that in the example shown in Fig. 4.5, the erroneous interpolation between accurate clusters reduces the amount of clusters from eight to four. This interpolation behaviour partially explains the reduced number of clusters found by DBSCAN in the samples produced by GGIK for the UR10. Whether our approach can be modified to consistently produce the maximum number of solutions possible in the case of 6-DOF manipulators remains to be seen, but the results in Fig. 4.6 show that our current model does not learn distributions concentrated around a single solution.

4.8.4 Time Taken to Generate Multiple Samples

We studied the scaling of the computation time of GGIK as a function of the number of solution generated and the number of DOF in the robot (i.e., the size of the robot). GGIK is able to take advantage

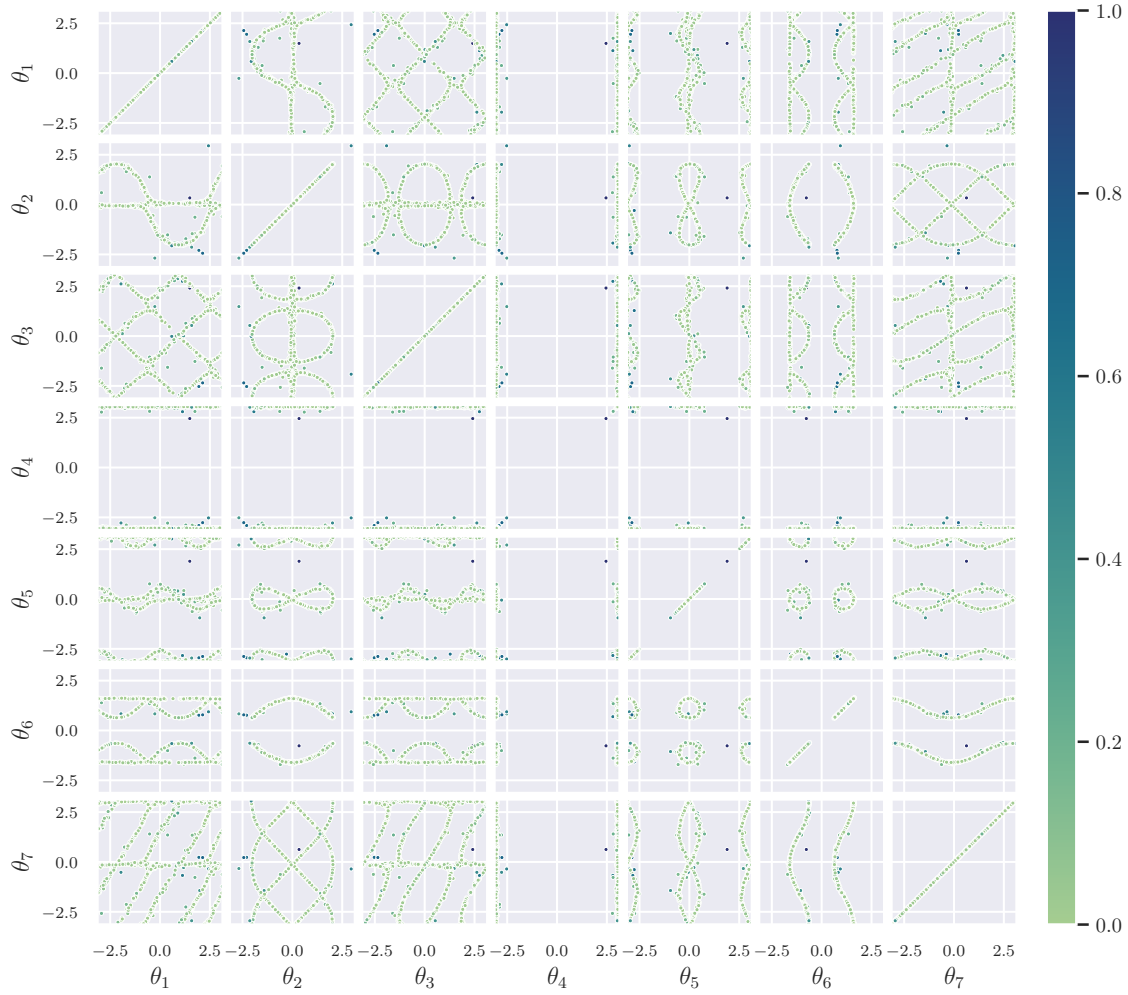


Figure 4.4: Pairwise plotting of 1,000 samples from GGIK (using a single model trained on all of the test manipulators) for a single Kuka goal pose. Each sub-plot contains samples on the 2-dimensional torus for joint angles θ_i and θ_j, leading to a symmetrical pattern (i.e., the upper triangle is a transposition of the lower triangle). The hue of each sample is proportionate to the end-effector's pose error (the sum of the Euclidean distance in metres and the angular distance in radians). For this particular pose, the continuous and orderly curves produced by accurate solutions indicate that GGIK is able to learn a relatively complex distribution over a large, varied solution set. Samples with higher errors appear to happen sporadically.

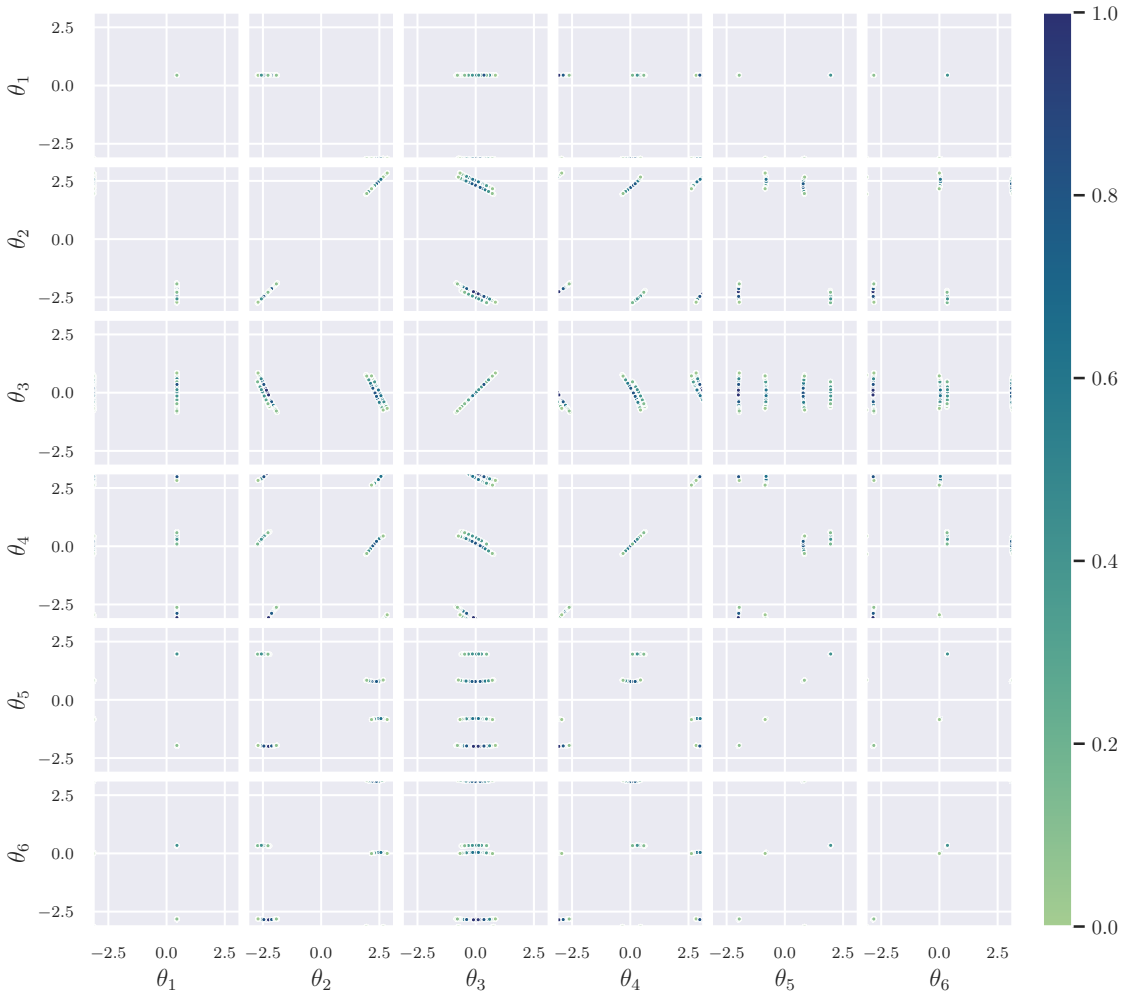
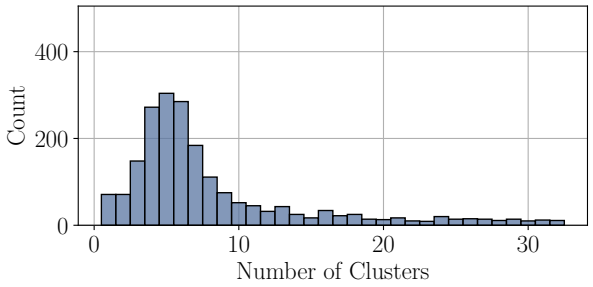
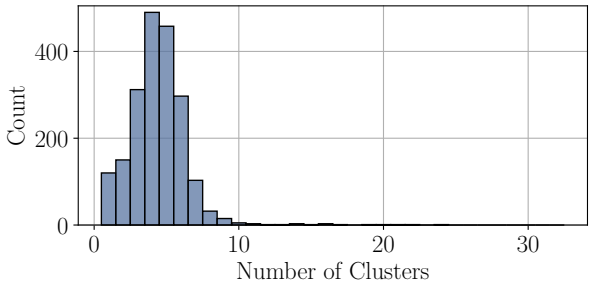


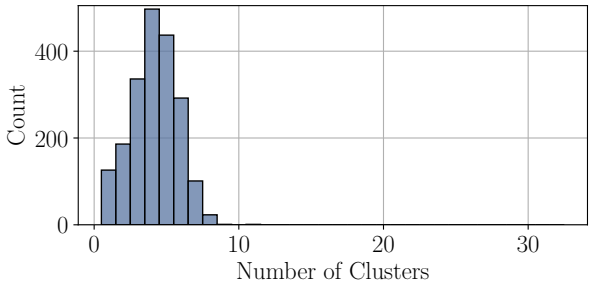
Figure 4.5: Pairwise plotting of 1,000 samples from GGIK (using a single model trained on all of the test manipulators) for a single UR10 goal pose. Each sub-plot contains samples on the 2-dimensional torus for joint angles θ_i and θ_j, leading to a symmetrical pattern (i.e., the upper triangle is a transposition of the lower triangle). The hue of each sample is proportionate to the end-effector's pose error. For this particular goal pose, the less accurate samples from GGIK appear to be attempts at *interpolating* between high accuracy discrete clusters (i.e., those with lighter colouring).



(a) $\epsilon = 0.01$



(b) $\epsilon = 0.1$



(c) $\epsilon = 1$

Figure 4.6: Distribution of the number of clusters produced by GGIK for 2,000 goal poses. Each clustering was performed on 32 GGIK samples with the DBSCAN algorithm and the specified radius ϵ. The ϵ value determines the sensitivity of the clustering procedure. The similarity between all three distributions indicate that the clustering procedure is fairly stable.

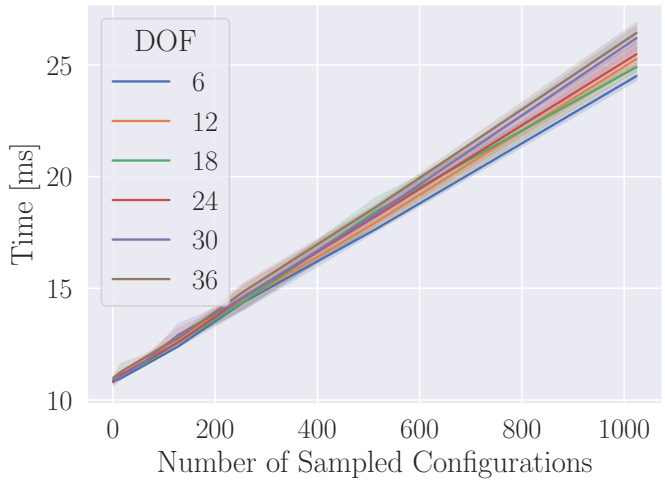


Figure 4.7: Plot of the time taken in milliseconds as a function of the number of sampled solutions generated by GGIK for robots of varying degrees of freedoms (DOF). For a range of robots with 6- to 36-DOF, it takes approximately 10 milliseconds to generate about 10 samples, and approximately 25 milliseconds to generate 1,000 samples. Shaded region shows 95% confidence interval. The size of the robot (i.e., size of the graph) and the number of samples do not significantly change the inference time due to efficient parallelization on the GPU.

of the GPU architecture and yield very low per-solution solve times when amortized across larger batches. This could be used, for example, in sampling-based motion planning and workspace analysis. The results displayed in Fig. 4.7 demonstrate that GGIK was able to generate 10 samples in 10 milliseconds and 1,000 samples in 25 milliseconds for a range of robots with 6- to 36-DOF.

4.8.5 GGIK and Numerical Solvers

The distribution that GGIK learns can be efficiently sampled to produce multiple approximate solutions in parallel. These samples can be further refined with relatively little additional computational cost by using them to initialize optimization-based methods. Table 4.4 shows the results of repeating the experiment in Section 4.8.1 using the TRAC-IK (Beeson and Ames, 2015) inverse kinematics solver, where the results are averaged over all problems and all robots.

	Err. Pos. [mm]					Err. Rot. [deg]					Soln. Time [ms]	
	mean	min	max	Q_1	Q_3	mean	min	max	Q_1	Q_3	mean	std
Method												
GGIK	7.09	2.52	12.63	5.16	8.84	0.52	0.15	0.84	0.38	0.66	0.34	0.03
TracIK + GGIK	0.15	0.01	0.86	0.04	0.18	0.01	0.00	0.08	0.00	0.02	0.50	0.09
TracIK + rand	0.17	0.00	0.97	0.02	0.22	0.02	0.00	0.07	0.00	0.02	0.58	0.39

Table 4.4: Comparison of GGIK (using a single model trained on all of the test manipulators) with TRAC-IK (Beeson and Ames, 2015) on the task of producing 32 solutions for 10000 randomly-generated IK problems. TRAC-IK is initialized using configurations sampled randomly (TRAC-IK + rand) or from GGIK (TRAC-IK + GGIK). The error statistics are presented as the mean and mean minimum and maximum error per problem, as well as the two quartiles of the distribution. The computation time per solution is computed as the average time required for a single solution to be produced per problem.

In our study, we measured the total per-problem computation time for GGIK, including both the time taken to sample point sets and to retrieve associated joint angles on the GPU, operations which can be executed concurrently. We used a single GGIK model trained on all of the test manipulators. We then calculated the per-sample computation time by dividing the total computation time by the number of samples (32). Specifically, GGIK alone achieved an average per-sample computation time of 0.34 ms, totalling approximately 11 ms per problem. For increased accuracy, these GGIK-generated samples were used to initialize the TRAC-IK solver. These initializations helped TRAC-IK converge to a solution in 0.16 ms on average, totalling 0.5 ms per-sample or 16 ms per problem when GGIK computation time is included. Conversely, randomly initialized TRAC-IK required on average 0.58 ms per sample, totalling 19.2 ms per problem. The notably higher convergence time for random initializations can be ascribed to the solver requiring a much larger number of steps to reach a feasible solution from a random point in the configuration space.

The accuracy of both TRAC-IK instances, in terms of position and rotation errors, is superior to that of GGIK alone, reflecting the known advantage of local solvers over learning-based methods. However, as Section 4.8.3 illustrates, GGIK offers control over the diversity of solutions, sampling extensively across the solution space. In contrast, we cannot predict which solutions local solvers will converge to with random initializations. Additionally, using GGIK to initialize TRAC-IK produced solutions that were more accurate than a randomly initialized TRAC-IK.

Our experiments, as detailed in Section 4.8.4, demonstrate that leveraging GPU parallelization allows the generation of a significantly larger number of samples with minimal additional computation time. This finding underscores the efficiency of combining GGIK with local solvers, particularly in complex inverse kinematics scenarios featuring multiple spatial constraints and joint limits. Overall, these results indicate that GGIK can be efficiently used in tandem with local solvers in order to produce a set of highly accurate and diverse solutions with lower or equal computation times.

4.8.6 Ablation Study on the Equivariant Network Architecture

We conducted an ablation experiment to evaluate the importance of capturing the underlying $\mathrm{E}(n)$ equivariance of graphical IK in our learning architecture. We compared the accuracy of GGIK when using the EGNN network (Satorras et al., 2021) as opposed to four commonly used GNN layers that are not $\mathrm{E}(n)$ equivariant: GRAPHsage (Hamilton et al., 2017), GAT (Velickovic et al., 2018), GCN (Kipf and Welling, 2017) and MPNN (Gilmer et al., 2017). We matched the number of parameters of each GNN architecture as closely as possible and kept all other experimental parameters fixed. The dataset for this experiment was the same one used in Section 4.8.1. For each GNN architecture, the results were averaged over all manipulators as shown in Table 4.5. Out of the five different architectures that we compared, only the EGNN and MPNN outputted point sets that could be successfully mapped to valid joint configurations. Point sets that were too far from those representing a valid joint configuration resulted in the configuration reconstruction procedure diverging. The equivariant EGNN model outperformed all other models in terms of the overall solution accuracy and ELBO value attained on a held-out test set.

Model Name	Err. Pos. [mm]					Err. Rot. [deg]					Test ELBO
	mean	min	max	Q_1	Q_3	mean	min	max	Q_1	Q_3	
EGNN (Satorras et al., 2021)	4.6	1.5	8.5	3.3	5.8	0.4	0.1	0.6	0.3	0.4	-0.05
MPNN (Gilmer et al., 2017)	143.2	62.9	273.7	113.1	169.1	17.7	5.3	13.6	21.6	34.1	-8.3
GAT (Velickovic et al., 2018)	-	-	-	-	-	-	-	-	-	-	-12.41
GCN (Kipf and Welling, 2017)	-	-	-	-	-	-	-	-	-	-	-12.42
GRAPHsage (Hamilton et al., 2017)	-	-	-	-	-	-	-	-	-	-	-10.5

Table 4.5: Comparison of different network architectures. EGNN outperforms existing architectures that are not equivariant in terms of overall accuracy and test ELBO. Dashed results are models with output point sets that were too far from a valid joint configuration and diverged during the configuration reconstruction procedure. Using 32 samples from the learned distribution per goal pose, the error statistics are presented as the mean and mean minimum and maximum error per problem, as well as the two quartiles of the error distribution.

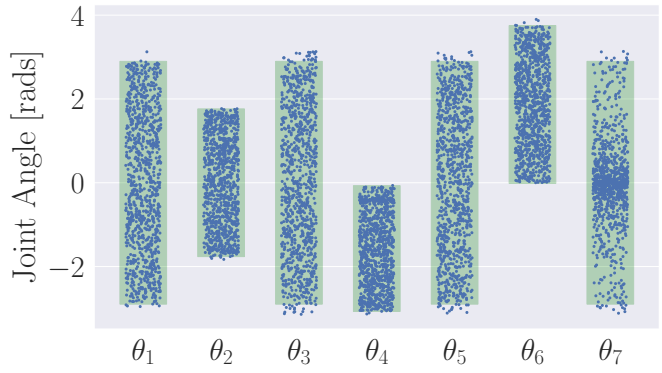


Figure 4.8: Joint angles of solutions generated by GGIK for 1000 Panda IK problems. The green shaded bars denote joint limits of the samples used during training. The solutions are denoted in blue, where each column represents a specific joint. GGIK is able to learn a constrained distribution of solutions with samples that are almost always within the joint limits.

4.8.7 Joint Limits

GGIK is able to learn a constrained distribution of solutions that obey joint limits. We demonstrated this by training a variant of GGIK using 512,000 samples from a Panda manipulator that are within specific joint limits. In our experiment with the Panda we used the joint limit values provided on the website of the manufacturer, Franka Emika.

At test time, we created 1,000 goal poses and used GGIK to solve for the associated joint angles. In Fig. 4.8, we visualized the joint angles of the solutions sampled by GGIK. The green shaded bars denote the range of valid joint angles. The joint angle solutions are denoted in blue, where each column represents a specific joint of the Panda. Notably, we observe that the samples almost always obeyed joint limits. A handful of joint solutions were slightly outside of the limits. This is to be expected since solutions are sampled from a learned distribution and joint limits are not explicitly enforced. In practice, we could run a few iterations of a numerical solver to guarantee that the solutions are within the joint limits. More generally, the findings illustrated in Fig. 4.8 show that the learned distribution of solutions from GGIK can be influenced by curating the training data.

4.9 Limitations

Our approach is not without its limitations. GGIK outputs may require post-processing by local optimization methods in applications with extremely low pose error tolerances. Moreover, the sampled solutions may require filtering in order to adhere to specific task or configuration space constraints. While filtering for common constraints (e.g., joint limits, pose error, proximity, etc.) may be carried out in parallel over all samples, conditioning the learned distribution on such constraints would likely be preferable.

Another potential limitation of our approach is inherited from our chosen representation (Maric

et al., 2021), where the partially connected distance graphs representing the IK problem only uniquely represent robots whose neighbouring joints have coplanar rotation axes. Unlike the matrix completion method used in (Maric et al., 2021), neural networks are capable of representing arbitrary functions based on data, and are therefore able to circumvent this ambiguity when learning the solution distribution for an individual robot with joint rotation axes that are not coplanar. However, the generalization capability a single model jointly learning solution distributions for multiple robots with non coplanar rotation axes may be more challenging due to the aforementioned ambiguity.

Finally, the uniform sampling strategy used in this thesis to generate training data may not work well for more complicated classes of robots such as humanoids or quadrupeds. In these cases, not all joint solutions are equally desireable due to additional potential constraints such as maintaining upright posture and static equilibrium. Furthermore, uniform sampling suffers from the curse of dimensionality as the number of robot DOF increases. We believe that a single potential solution to both of these issues would be targeted and biased sampling (e.g., only sampling solutions that maximize manipulability or where the humanoid remains upright).

4.10 Summary and Future Work

We have presented GGIK, a generative graphical IK solver that is able to produce multiple diverse and accurate solutions in parallel across many different manipulator types. This capability is achieved through a distance-geometric representation of the IK problem in concert with GNNs and generative modelling. The accuracy of the generated solutions points to the potential of GGIK as both a standalone solver and as an initialization method for local optimization methods. To the best of our knowledge, this is the first learned IK model that can generate multiple solutions for robots not present in the training set. Most important, because GGIK is fully differentiable, it can be incorporated as a flexible IK component that is part of an end-to-end learning-based robotic manipulation framework. GGIK provides a framework for learned general IK—a universal solver (or initializer) that can provide multiple diverse solutions for any manipulator structure in a way that complements or replaces numerical optimization.

As future work, it would be interesting to learn constrained distributions of robot configurations that account for obstacles in the task space; obstacles can be incorporated in the distance-geometric representation of IK (Maric et al., 2021; Giamou et al., 2022) as nodes. Learning an obstacle- and collision-aware distribution would yield a solver that implements collision avoidance by way of message passing between manipulator and obstacle nodes. However, the exact representations of the manipulator and obstacle shapes would require further investigation. For example, a sphere-based representation, as used in (Lembono et al., 2021), would be relatively simple to implement but may be too conservative when compared to some other 3D neural representation that might contain more information and higher resolution. It would also be interesting to investigate the idea of fine-tuning a generic model trained on multiple robots to perform better on a specific robot with only a few iterations of learning. We observed that GGIK seems to sometimes erroneously interpolate between the

discrete solution sets of the UR10, and investigating ways to mitigate this behaviour is a promising research direction. Finally, it would be interesting to extend GGIK for use with other classes of robots such as kinematic trees (e.g., quadrupeds and humanoids). In particular, we would like to consider the case where multiple goal poses are specified and to investigate the sampling strategies needed to efficiently generate proper training datasets.

4.11 Associated Publications

This project was a joint collaboration with Filip Marić. Filip lead the design of the distance geometry formulation, while I lead the implementation of the generative learning architecture.

1. Limoyo, O., Maric, F., Giamou, M., Alexson, P., Petrovic, I., and Kelly, J. (2023d). Generative graphical inverse kinematics. *IEEE Transactions on Robotics*. To Appear.

2. Limoyo, O., Maric, F., Giamou, M., Alexson, P., Petrovic, I., and Kelly, J. (2023c). Euclidean equivariant models for generative graphical inverse kinematics. In *Proceedings of the Robotics: Science and Systems (RSS) Workshop on Symmetries in Robot Learning*, Daegu, Republic of Korea

Chapter 5

Learning to Place by Picking

In this chapter, we consider generative self-supervision more broadly as a way to automatically generate training data in the context of robotic pick-and-place. We introduce a method that we call *placing via picking* (PvP) that exploits the symmetric structure of the pick and place problems to generate expert demonstrations of robotic placement in a self-supervised manner and without any human labour. We further show how tactile sensing and compliant control are crucial for both robust and uninterrupted data collection and leveraging environment contact-constraints as guides to inform object placement.

5.1 Motivation

Imitation learning (IL) provides a scaleable, simple, and practical option to learn control policies from expert demonstrations (Ablett et al., 2021; Florence et al., 2021). The general approach of training expressive models on large and diverse datasets has proven to be quite effective outside of robotics (Blattmann et al., 2023; OpenAI, 2023). While there have been many attempts to transfer this paradigm to robotic systems (Jang et al., 2022; Lynch et al., 2023), robots have yet to achieve similar successes. A major bottleneck for IL-based approaches in these data regimes is the human time and effort required to collect a large number of expert demonstrations in the physical world.

Having said this, many recent methods attempt to streamline (Chi et al., 2024; Zhao et al., 2023) or automate (Kalashnikov et al., 2018; Kalashnkov et al., 2021; Ahn et al., 2024; Bousmalis et al., 2023) data collection. As highlighted by the authors of (Ahn et al., 2024), the lack of robust and diverse autonomous data collection policies is a major limitation to scaling up IL. In this work, we are interested in investigating autonomous data collection for robotic object placement. As opposed to much prior research, we tackle the *full* placing problem, without restricting our task to simple, small objects placed on flat, horizontal surfaces with no environment contact constraints. We assert that object and environment contact constraints can be a valuable, natural guide for learning how to place, if handled properly.

Our novel approach, named *placing via picking* (PvP), automates the collection of expert demonstrations for a large subset of contact-constrained placing problems. We do so by taking advantage

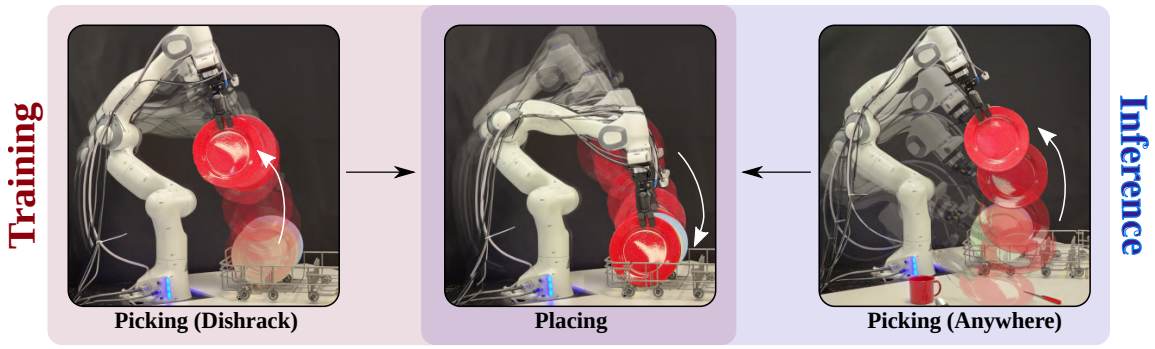


Figure 5.1: An overview of *placing via picking* (PvP). *Left:* During training, we collect placing demonstrations by reversing the grasping trajectory with objects that are initially at their target locations (in the dish rack). *Right:* During inference, our learned vision-based policy can generalize to object placement scenarios where the objects are not at their target locations initially.

of a powerful grasp planner (Sundermeyer et al., 2021), tactile sensing, and compliant control. PvP is a self-supervised pipeline to autonomously collect expert placing demonstrations by leveraging the cyclical nature and inherent symmetry of the pick and place tasks. Given an environment with objects initially at their target locations, we alternate between picking (i.e., grasping and retrieving) and placing, by time-reversing the retrieval trajectory. We generate demonstration data for object placement in a self-supervised manner—the picking phase provides the training supervision for the placing task. While appearing deceptively easy at first, we highlight the crucial importance of two modules, compliant control for grasping (CCG) and tactile regrasping (TR). CCG and TR enable (1) robust and uninterrupted pick-and-place and (2) the use of environment contact-constraints as guides to inform object placement. Fig. 5.1 provides a visual summary of our approach.

5.2 Related Work

In this section, we review prior work on automated robot data collection, discuss the general idea of 'working backwards,' and investigate existing robotic placement strategies.

5.2.1 Automatic Data Collection

A large number of existing policy learning approaches use prior policies to automate the data collection process. Examples include using trajectory optimization (Dalal et al., 2023; Levine et al., 2016) and, closer to our work, simple scripted pick-and-place policies (Kalashnikov et al., 2018; Kalashnkov et al., 2021; Ahn et al., 2024) as forms of supervision. The authors of (Kalashnikov et al., 2018) use a scripted pick-and-place policy to bootstrap the collection of the initial dataset used for off-policy deep reinforcement learning with a real-world manipulator. In (Ahn et al., 2024), the authors present AutoRT, a method that uses a combination of scripted, learned, and teleoperated data collection policies to efficiently collect a large amount of language-specified manipulation demonstration data. Using a combination of a large language model (LLM) and visual language models (VLMs), AutoRT observes

scenes in the wild and comes up with plausible tasks. Once a task is defined and chosen, AutoRT samples a data collection policy to attempt to gather real world data for the task. In our work, we focus on developing a method for collecting demonstrations of placement with larger and more complex objects (i.e., objects that require a planner to grasp) in environments with contact constraints. Unlike prior work, we do not restrict the placing task to simple small objects on flat horizontal surfaces. The authors of (Ahn et al., 2024) mention the lack of robust and diverse real-world autonomous collection policies as a limitation of AutoRT; our work is complementary to many of the previous frameworks, where PvP could be used as a prior policy.

Other works have explored automatic data generation by bootstrapping from a single human demonstration (Johns, 2021; Li et al., 2023b). In (Johns, 2021), a robot manipulation task is modelled as having two phases: a coarse approach trajectory phase towards a bottleneck pose, followed by a fine interaction trajectory phase. The single human demonstration provides the relative pose of the manipulator with respect to the task-relevant object, from which a self-supervised data collection procedure is formulated to train a bottleneck pose predictor. In our work, we also automatically collect and augment approach trajectory data. However, we do not require any human demonstrations since we focus on placing tasks only. By doing so, we can leverage the inherent symmetry of picking and placing for self-supervised data collection through the use of a combination of tactile sensing, compliant control, and an off-the-shelf grasp planner.

5.2.2 Working Backwards

The general idea of working backwards or time-reversal has been exploited in multiple contexts including generative modelling (Sohl-Dickstein et al., 2015), learning visual predictive models for control (Nair et al., 2020; Limoyo et al., 2020a, 2023b), and reinforcement learning (Andrychowicz et al., 2017). Our self-supervised data collection method, which extends the concept of working backwards to the problem of robotic placement, is most similar in spirit to (Fu et al., 2023; Spector and Di Castro, 2021; Spector et al., 2022). These methods consider objects that are assumed to be in their desired place configuration during data collection. The manipulator can then pick an object and move back to the initial goal location to generate training data. However, we focus on placement as opposed to insertion. Insertion policies generally only reason about local interactions. On the other hand, placement involves reasoning over a larger scene with the potential for multiple place poses and more complex objects that require planners to grasp successfully. Furthermore, unlike prior work (Fu et al., 2023; Spector and Di Castro, 2021; Spector et al., 2022), our system does not require a human to guide the manipulator to the object and instead leverages tools from grasp planning. Our approach also resembles (Zakka et al., 2020), where time-reversed trajectories are used to generate assembly data from disassembly data in a self-supervised manner. However, we focus on generating demonstrations for a closed-loop reactive policy that uses a 6-DOF pose-based action representation for placement. In contrast, the self-supervised data collection procedure introduced in (Zakka et al., 2020) uses a 2D open loop pick-and-place action primitive.

5.2.3 Robotic Manipulator Placing

Pick and place is a fundamental problem in robotic manipulation. A significant amount of research has focused on the picking problem (Bicchi and Kumar, 2000; Bohg et al., 2013; Zeng et al., 2022), while placing has been relatively less studied.

Previous approaches have explored the use of motion planning combined with place-specific objectives (Haustein et al., 2019b,a). For example, in (Haustein et al., 2019b), the authors demonstrate a hierarchical sampling-based motion planner that first finds poses which satisfy constraints (e.g., reachability) and then performs a local optimization for user-specified objectives (e.g., clearance). The place problem has also been studied from the perspective of geometric modelling (Harada et al., 2014), where the planar surfaces of a discretized object model are matched to planar patches in the environment. Importantly, 3D models of the objects being placed as well as the environment are assumed to be available ahead of time.

Closer to our approach, other works have trained end-to-end place policies using reinforcement learning (Dong et al., 2021) and imitation learning (Finn et al., 2016). In (Finn et al., 2016), the authors demonstrate an inverse reinforcement learning technique capable of simultaneously learning a cost function and a control policy from human kinesthetic demonstrations of dish placing. The main contribution and novelty of our work is a data collection technique for robotic placing tasks that can be used in tandem with methods that require expert demonstrations.

5.3 Placing via Picking

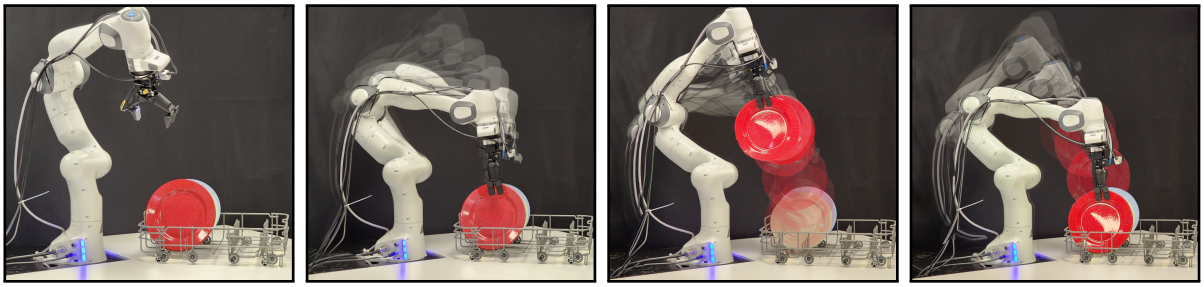
We first present an overview of our self-supervised data collection method, PvP, in Section 5.3.1, and then describe a method for noise augmentation during data collection in Section 5.3.2. Finally, we provide details about the procedure used to train our IL policy in Section 5.3.3.

5.3.1 Self-Supervised Data Collection

PvP consists of a cycle with four main phases: (1) grasp planning, (2) grasping, (3) retrieving, and (4) placing by reversing; these steps are visualized in Fig. 5.2. During training, the objects of interest are initially in their goal positions.

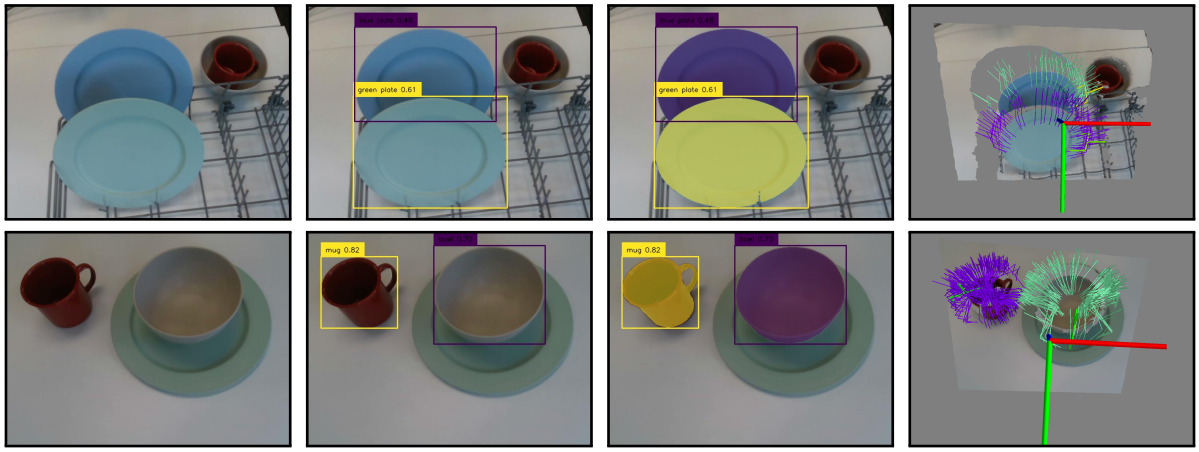
Grasp Planning

At the start of the data collection process, the manipulator moves to a predetermined pose where the camera has a clear view of the objects of interest in the environment, as shown in Fig. 5.2a. We use the grasp planner Contact-GraspNet (Sundermeyer et al., 2021) to generate L candidate grasp poses $\{\mathbf{T}_{g,i}\}_{i=0}^{L}$, where $\mathbf{T}_{g,i} \in \mathrm{SE}(3)$ for all $i \in \{0, ..., L\}$. Given a set of text-based descriptions of the objects of interest (e.g., "plates," "green plate," "cup" or, more generally, "objects"), potentially from a user or high-level LLM planner, we use Grounding-Dino (Liu et al., 2023) to find the object



(a) Grasp planning. **(b)** Grasping. **(c)** Retrieving. **(d)** Placing by reversing.

Figure 5.2: The four steps involved in PvP, our autonomous demonstration data collection process for placing. We (a) generate grasps with an off-the-shelf grasp planner (Sundermeyer et al., 2021); (b) compliantly grasp the object to apply minimal forces to the environment while ensuring a stable grasp via tactile sensing; (c) retrieve the object with rotational compliance while storing the trajectory; and (d) generate placement demonstration data by rolling out the reversed grasp trajectories while storing the observations and actions.



(a) Raw image. **(b)** Text-based object detection. **(c)** Segmentation. **(d)** Pruned grasps.

Figure 5.3: The steps involved in our language-driven grasp planning pipeline. We use (b) Grounding-Dino (Liu et al., 2023) for object bounding box detection based on text descriptions and (c) Segment Anything (Kirillov et al., 2023) on the cropped images to produce object specific masks. We then use (d) Contact-GraspNet (Sundermeyer et al., 2021) for grasp generation on only the segmented objects (i.e., masked areas). *Top*: using "green plate" and "blue plate" as the targets for data collection. *Bottom*: using "bowl" and "mug" as the targets for data collection.

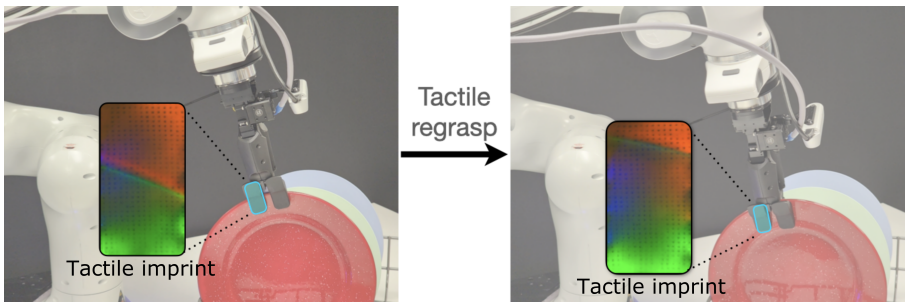


Figure 5.4: Tactile images from before and after a tactile regrasp (TR). Left: the plate's contact surface area fills half of the tactile image indicating a shallow grasp. Right: after a regrasp, the contact surface area has increased indicating a stable grasp.

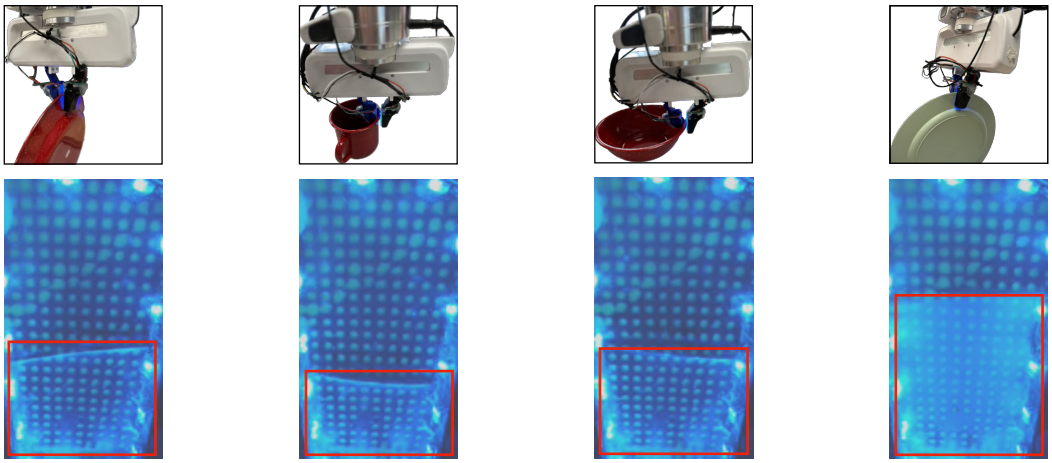
bounding boxes. Finally, we run Segment Anything (Kirillov et al., 2023) on each of the previously-generated image crops from the bounding boxes to generate per-object masks. Contact-GraspNet then uses these masks to filter grasps that are not on objects of interest and keeps only a pruned set of grasps $\{\mathbf{T}_{g,i}\}_{i=0}^{K}$, where $K \leq L$. We visualize an example of this process in Fig. 5.3 for two different scenes. After pruning, we randomly select a single grasp pose, $\mathbf{T}_{\text{grasp}}$, from $\{\mathbf{T}_{g,i}\}_{i=0}^{K}$. To approach without collisions, we include a pregrasp pose $\mathbf{T}_{\text{pregrasp}}$ that is defined with a translational offset above the grasp pose.

Grasping

Given the computed pregrasp and grasp poses, we move to each pose sequentially by linearly and spherically linearly interpolating to produce a smooth trajectory. An example of a grasp sequence is shown in Fig. 5.2b. During grasping, two modules play a critical role in the robustness of PvP: compliant control for grasping (CCG) and tactile regrasping (TR).

We use a Cartesian impedance controller for manipulation. This choice allows us to set the level of compliance of the manipulator at various stages as needed. Before closing the gripper to grasp, we set the translational and rotational stiffness of the manipulator's controller to minimal values. We call this CCG. The high compliance of the robot arm allows the manipulator and gripper to comply with the contact constraints and minimize the applied forces against the environment. Larger contact-constrained objects generate unnecessary reaction forces when gripped without compliance. Large forces, in turn, trigger emergency stops and can damage the environment and the robot. Specific to PvP, large reaction forces also disrupt and move the object relative to the gripper, ultimately leading to inaccurate placements. On the other hand, when grasping with compliance, the environment contact constraints guide the manipulator towards a natural object placement pose.

Our method assumes that the grasps that we perform are *stable*, that is, the object does not move or slip significantly once grasped. In the real world, the accumulation of errors from various sources (e.g., camera calibration, controller, learned grasp planner, and noise from the RGB-D sensor and manipulator encoders) can result in grasps that are unstable (e.g., shallow grasps) and not ideal for PvP. To mitigate this issue, we use tactile sensing—specifically, a visuotactile Finger-STS (Hogan et al., 2022)



(a) (b) (c) (d)

Figure 5.5: Visualization of contact surfaces in the tactile image from grasps of various objects: (a) a red metal plate, (b) a red metal mug, (c) a red metal bowl and (d) a green wheat straw plate.

sensor—to preemptively detect if a grasp is stable and perform a regrasp if needed. The Finger-STS is capable of providing multimodal feedback (i.e., both tactile and visual observations). To this end, we design our TR module to detect the contact surface area and calculate the relative change in end effector (EE) pose required to recover a larger contact surface and, thereby, a more stable grasp. We detect regions of contact based on marker displacements and the RGB image from the Finger-STS. If partial contact is detected, we calculate the difference between the ideal and detected contact region and use this to calculate the commanded change in EE pose. We note that a simpler and more general random regrasp strategy could also be used. An example of this is shown in Fig. 5.4. We visualize contact regions for various household items in Fig. 5.5.

In Section 5.4.1, we experimentally investigate the effects of including these modules through an ablation study.

Retrieving

We begin the retrieval phase when a valid, stable grasp is detected. An image of the retrieval procedure is shown in Fig. 5.2c. We store the poses of the EE as well as the respective timestamps throughout the retrieval process. For our work, we measure joint encoder readings at a rate of 120 Hz. The manipulator is set to be compliant along the rotational axes of the EE only, when commanded to move to the pregrasp pose. Once at the pregrasp pose, we randomly sample a clearance pose $\mathbf{T}_{\text{clearance}}$ around a set fixed pose above the scene. That is, given a set fixed pose, we add random translations sampled from $\mathcal{N}(0,\, \sigma_{tr}^2)$ to recover a clearance pose. We use σ_{tr} = 2.5 cm during data collection.

At the end of the retrieval phase, we have an expert retrieval trajectory of length M from grasp pose to clearance pose via a pregrasp pose,

$$\tau_r^{\text{expert}} = ((\mathbf{T}_0, t_0), ..., (\mathbf{T}_M, t_M)), \tag{5.1}$$

where $\mathbf{T}_0 = \mathbf{T}_{\text{grasp}}$, $\mathbf{T}_M = \mathbf{T}_{\text{clearance}}$ and $t_0, ..., t_M$ are the respective timestamps of the trajectory starting with t_0 = 0. To match the desired control frequency of our policy, we sample states Δt apart to generate a sparser trajectory $\bar{\tau}_r^{\text{expert}}$. We extract a total of $\bar{M} + 1$ states, where $\bar{M} = \lceil \frac{t_M - t_0}{\Delta t} \rceil$, by finding the nearest pose (in terms of time) from the dense trajectory τ_r^{expert} at every Δt interval:

$$\bar{\tau}_r^{\text{expert}} = ((\bar{\mathbf{T}}_0, 0), (\bar{\mathbf{T}}_1, \Delta t), ..., (\bar{\mathbf{T}}_{\bar{M}}, \bar{M}\Delta t)). \tag{5.2}$$

In this work, our visual policies operate at 5 Hz, so we set $\Delta t = 0.20$ s.

Placing by reversing

We reverse the sparse retrieval trajectory $\bar{\tau}_r^{\text{expert}}$ to extract a desired place trajectory,

$$\bar{\tau}_p^{\text{expert}} = (\bar{\mathbf{T}}_{\bar{M}}, ..., \bar{\mathbf{T}}_0). \tag{5.3}$$

An example of a place trajectory is shown in Fig. 5.2d. Additionally, we convert the global poses to relative poses in order to use state differences as expert actions:

$$\Delta\mathbf{T}_i = \bar{\mathbf{T}}_{\bar{M}-i}^{-1} \bar{\mathbf{T}}_{\bar{M}-i-1}, \tag{5.4}$$

where $i \in \{0, ..., \bar{M}-1\}$ and $\Delta\mathbf{T}_i \in \text{SE}(3)$ is a relative change in EE pose. We define the expert action as $\mathbf{a}_i = (\Delta\mathbf{T}_i, a_{\text{gripper},i})$, where $a_{\text{gripper},i} \in \{0, 1\}$ is a binary variable describing the gripper command (open or close). To collect an episode of expert data, we perform a rollout of the downsampled place commands, $\{\mathbf{a}_i\}_{i=0}^{\bar{M}-1}$, and store the sequences of observations and actions as an expert trajectory. We use RGB images from the wrist camera to form the robot's observation space $\mathbf{o}_i$. We downsample the images to a resolution of 128 $\times$ 128 and also stack the three previous RGB frames, which results in $\mathbf{o}_i \in \mathbb{R}^{128\times128\times12}$. During the place demonstration, the robot keeps its gripper closed (i.e., the robot uses the gripper command $a_{\text{gripper},i} = 1$ for all $i \in \{0, ..., \bar{M}-1\}$). Finally, at the end of the trajectory, we command the robot to keep the EE in its current pose and to open the gripper (i.e. $\Delta\mathbf{T}_i = \mathbf{I}$ and $a_{\text{gripper},i} = 0$) for N time steps. We use $N = 5$, which gives the robot an adequate amount of time to open the gripper and release the object. A single expert demonstration or trajectory is then defined as a set of training tuples

$$\tau^{\text{expert}} = \{(\mathbf{o}_t, \mathbf{a}_t, \mathbf{o}_{t+1})\}_{t=0}^{T}, \tag{5.5}$$

with a total length of $T = \bar{M} + N$.

5.3.2 Noise-Augmented Data Collection

To increase the coverage of our demonstrations and train more robust policies, we perturb the first 75% of the poses with noise. We take inspiration from previous works that have investigated the use of noise injection for both the policy observations (Florence et al., 2019), and actions (Laskey et al., 2017). We also take inspiration from robust control theory, where injecting isotropic Gaussian noise has been

shown to achieve *persistent excitation* (Green and Moore, 1986), a condition where the training data is informative enough to learn a model that is robust to compounding errors during deployment. We call this variant *noise-augmented data collection*. In practice, we represent pose commands $\Delta\mathbf{T}$ as a combination of a translation vector $\mathbf{t} \in \mathbb{R}^3$ and a rotation vector $\boldsymbol{\theta} \in \mathbb{R}^3$. The rotation vector is defined as $\boldsymbol{\theta} = \theta\mathbf{e}$, where $\theta \in \mathbb{R}$ is the rotation angle and $\mathbf{e} \in \mathbb{R}^3$ is the Euler axis. We perturb the translation of the pose with

$$\mathbf{t}_{\text{perturbed}} = \mathbf{t} + \delta\mathbf{t}, \tag{5.6}$$

where $\delta\mathbf{t} \in \mathbb{R}^3$ is an isotropic Gaussian noise vector and each dimension of $\delta\mathbf{t}$ is sampled from $\mathcal{N}(0, \sigma_t)$. Similarly, we perturb the rotation with

$$\boldsymbol{\theta}_{\text{perturbed}} = (\theta + \delta\theta)(\mathbf{e} + \delta\mathbf{e}), \tag{5.7}$$

where $\delta\mathbf{e} \in \mathbb{R}^3$ is an isotropic Gaussian noise vector and each dimension of $\delta\mathbf{e}$ is sampled from $\mathcal{N}(0, \sigma_e)$, and $\delta\theta \in \mathbb{R}$ is sampled from $\mathcal{N}(0, \sigma_\theta)$. We perturb both the rotation axis and the rotation angle. We find $\sigma_t = 0.5$ cm, $\sigma_e = 0.5$ cm and $\sigma_\theta = 0.5^\circ$ to be reasonable values. We evaluate the effects of this augmentation on policy performance in Section 5.4.2

5.3.3 Policy Learning

We learn a policy $\pi_{\boldsymbol{\phi}}(\mathbf{a}_t \mid \mathbf{o}_t)$ parameterized by $\boldsymbol{\phi}$ using data generated by PvP. We use behavioural cloning (BC) (Pomerleau, 1989) to train our policy with a likelihood-based loss or objective function. Given a dataset of N_{train} demonstration trajectories, our loss function is then:

$$\mathcal{L} = \frac{1}{N_{\text{train}}} \sum_{i=1}^{N_{\text{train}}} \sum_{t=0}^{T} -\log \pi_{\boldsymbol{\phi}}(\mathbf{a}_t \mid \mathbf{o}_t). \tag{5.8}$$

The main backbone of our policy network consists of a convolutional neural network (CNN) based on the ResNet18 architecture (He et al., 2016), followed by a multilayer perceptron (MLP) layer that maps the CNN output into the parameters of the distribution of actions. In this work, we consider two representations for the action distribution $\pi_{\boldsymbol{\phi}}(\mathbf{a} \mid \mathbf{o})$: a unimodal Gaussian and a multimodal mixture of Gaussians. In both cases, during inference or control, we use a low-noise evaluation scheme, as done in (Wang et al., 2020) and (Mandlekar et al., 2021), by setting the standard deviation of the Gaussian distributions to be an arbitrarily small value. Note that the unimodal Gaussian is then equivalent to a standard deterministic policy trained with a mean squared error (MSE) loss.

5.4 Experiments

We used PvP to collect data and train robotic placement policies for two separate tasks: dishrask loading and table setting, as shown in Fig. 5.6. In both cases, the place policy had to account for the weight and shape of each item, the initial grasp of the object, and the state of the current scene to

(a) dish rack placement task, which consists of placing multiple plates of varying physical properties in evenly spaced slots.

(b) Table placement task, which consists of placing a bowl on a plate and a cup on a coaster.

Figure 5.6: Sequence of images from roll outs of place policies trained using data collected with PvP. The policies are able to place objects of varying properties in the scene using images from the wrist camera directly.

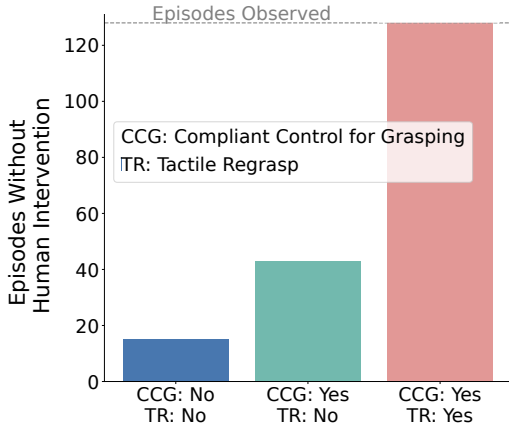


Figure 5.7: Ablation study on compliant control for grasping (CCG) and tactile regrasping (TR). We measured the average number of episodes that could be collected autonomously based on the number of failures. Both CCG and TR play a crucial role in reducing the number of failures to zero, which allowed us to collect the necessary amount of data seamlessly.

decide where and how to place (e.g., place angle). In this section, we only present our quantitative results for the more challenging dish rack loading task as it better represents the type of task (with added contact constraints from the environment) that PvP was designed for. We study the robustness of PvP in Section 5.4.1, the effect of noise augmentation on the performance of the learned placing policy in Section 5.4.2, and the relative quality of policies trained with data from PvP and from kinesthetic teaching in Section 5.4.3.

5.4.1 PvP Data Collection Robustness

We require a robust data collection loop to collect a large number of place demonstrations autonomously with PvP. We studied the effects of CCG and TR, as introduced in Section 6.3, on the robustness of PvP.

We observed two main causes of misplaced objects during data collection, initially discussed in Section 5.3.1, First, the manipulator would stiffly grasp the object causing it to move and generate a large amount of force preload against the environment. During retrieval, significant object motion

Table 5.1: Success rates for a model trained on a dataset collected with and without noise augmentation and using two different policy action representations. The noise-augmented data collection procedure improves the performance of both deterministic and Gaussian mixture policies.

Action Representation	Trained with Noise Aug. Data	Avg. Success Rate (↑)	Total Successes (↑)
Deterministic	No	71.67(5.93)	57/80
	Yes	81.11(5.50)	56/70
Gaussian Mixture	No	72.22(5.67)	50/70
	Yes	**87.78(3.93)**	**62/70**

occurred at the moment when the object's contact force with the environment was released. This unexpected shift in the object pose lead to less accurate placements since the reversal procedure no longer resulted in correct alignment between the object and the environment. With PvP, we mitigate this failure mode with CCG. The second failure mode involved unstable grasps that lead to a large relative motion between the object and the gripper. With PvP, we mitigate unstable grasps using TR to preemptively detect unstable grasps and to regrasp if needed.

We conducted an ablation study on CCG and TR where we ran self-supervised data collection for a total of 128 episodes and recorded the number of failures. We visualize the results in Fig. 5.7. The naive approach without CCG and without TR failed a total of 15 times, which roughly translates to an average of nine episodes collected autonomously before a human intervention is needed. The number of failures dropped to three with the addition of CCG, which raises the average to 43 episodes collected before each intervention. Finally, with both CCG and TR, we were able to successfully collect 128 episodes without any human intervention.

5.4.2 Noise Augmentation Ablation

In Section 5.3.2, we introduced a noise-augmented data collection variant of PvP. We investigated its contribution to the success rate of the policy. We collected two different datasets each containing 128 demonstrations, with and without added noise augmentation. We tested the effect of noise-augmented data on policies trained using two different action representations: a deterministic policy and a Gaussian mixture policy with five modes. We present the results in Table 5.1. For each row, we averaged the success rate over three models trained from scratch with three different seeds. We used 20 or 30 rollouts to calculate the success rate of each model. The numbers in the brackets denote variation of one standard deviation. We also report the total successes summed over the three models. For both action representations, we found that training on noise-augmented data improved the performance of the final policy significantly. The deterministic policy had an improvement of approximately 10% while the mixture policy had a larger improvement of 15%.

We hypothesize that adding noise further accentuates the multimodal nature of the data (i.e., multiple valid actions are available for roughly the same observation) and that a mixture representation for

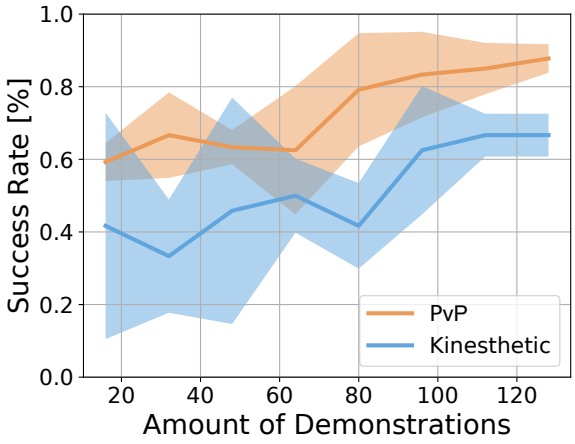


Figure 5.8: Comparison of success rates of policies trained using a dataset collected by PvP and a traditional kinesthetic teaching approach. We tested models trained on an increasing amount of demonstration data. Policies trained on data collected by PvP outperform those trained on data collected from kinesthetic teaching for almost all dataset sizes. The shaded region consists of one standard deviation.

the actions allows the policy to better capture this by not having to average over multiple possible expert actions (Bishop, 1994a). We also qualitatively observed that policies trained with noise-augmented datasets were better at re-adjusting the object with respect to the goal.

5.4.3 Comparison to Kinesthetic Teaching

We compared the performance of policies trained using demonstration data from PvP and from kinesthetic teaching. In particular, we investigated whether the quality of the demonstrations differ. To compare, we trained two separate policies, both sharing the architecture outlined in Section 5.3.3, on two separate datasets: one collected with PvP and another collected with kinesthetic teaching (Billard et al., 2016). We used a Gaussian mixture model as the action representation of both policies. For PvP, we used the noise-augmented variant since it was the best-performing model, as shown in Section 5.4.2.

In Fig. 5.8, we visualize the success rates of both models trained with varying numbers of demonstrations. As done previously, we averaged the performance over three models with different seeds for each datapoint. We tested each model using eight rollouts. The models trained with data from PvP, collected without any human involvement, significantly outperform the models trained using kinesthetic data, by about 20% for almost all numbers of demonstrations. We observed that policies trained with kinesthetic teaching often misplaced the plate (e.g., by not having the right place angle or not resting the plate in a stable manner with respect to the environment contact points) and struggled to open the gripper at the correct times. The latter of the two error modes did not appear in policies trained with PvP.

We hypothesize that the kinesthetic policy performed more poorly than the PvP policy because the overall quality of the human demonstrations was lower than the programmatic or machine-generated demonstrations produced by PvP. Previous work has studied the difficulties and challenges of training

policies using human demonstrations when compared to programmatic or machine-generated demonstrations (Kostrikov et al., 2019; Orsini et al., 2021; Mandlekar et al., 2021). In particular, human demonstrations can be sub-optimal owing to mistakes (e.g., large variations in trajectory length, unwanted movements, and remaining idle for some time to think). The authors of (Mandlekar et al., 2021) note that the relative average trajectory length is a good proxy for the quality of the dataset. The average number of time steps for all demonstrations was 29.41 (2.22) for the PvP dataset and 41.66 (5.69) for the kinesthetic dataset. The numbers in the brackets denote variation of one standard deviation. Kinesthetic demonstrations were on average longer and had greater variance in length. In addition, PvP benefits from systematic noise-augmentation, as covered in Section 5.3.2, which a human demonstrator cannot emulate consistently.

5.5 Limitations and Future Work

PvP relies on effective grasping (i.e., accurate grasp detections and stable grasps during retrieval). This makes this method susceptible to the typical challenges involved in grasping, namely dealing with noisy point cloud measurements, unreliable object segmentation, and unmodelled object dynamics (e.g. object slip). The quantity, quality, and diversity of the grasp poses found by the planner are partly determined by the viewpoint of the camera. In this work, we empirically selected a viewpoint that is adequate for our use case. It would be interesting to allow the manipulator to explore and find better grasp poses. Furthermore, while tactile sensing is used to improve the robustness of data collection, it could also be used as an extra modality for the policy. We have introduced a framework for data collection and tested it with two real-world tasks. We would like to test PvP on a larger set of tasks for a more exhaustive evaluation. Lastly, PvP is a language-driven data collection method, and it would be natural to integrate it into a high-level language based planner (Ahn et al., 2024).

5.6 Associated Publications

1. Limoyo, O., Konar, A., Ablett, T., Kelly, J., Hogan, F. R., and Dudek, G. (2024a). Working backwards: Learning to place by picking. In *IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS)*, Abu Dhabi, United Arab Emirates. Submitted

Chapter 6

Language-Guided Robotic Photography

In Chapters 3, 4, and 5, we re-interpreted classical robotic problems through the lens of generative SSL and explored how various sources of structure in these problems could be used as signals for learning. Much of this work has been inspired by the recent breakthroughs in SSL for computer vision and natural language processing. In this final chapter, we demonstrate how vision (Oquab et al., 2023) and language (OpenAI, 2023) models previously trained solely via generative self-supervision can be effectively and easily integrated downstream into existing robotic algorithms and systems. We present *PhotoBot*, a framework for fully automated photo acquisition based on an interplay between high-level human language guidance and a robot photographer.

6.1 Motivation

Photographing a human subject requires nuanced interaction and clear communication between the photographer and the model. Beyond just capturing well-composed photos, a professional photographer needs to understand what the client wants and to provide suggestions. Much of the prior research in the area of robotic photography Zabarauskas and Cameron (2014); Byers et al. (2003a); Campbell and Pillai (2005); Byers et al. (2003b) has focused on the technical aspects, that is, how to navigate, plan, and control a robot to frame a photo, but not on the interaction between the photographer and human model. Further, approaches relying on heuristic composition rules may not produce captivating photos, in part because quantifying the aesthetic quality of all images in a generalizable way is a difficult, unsolved problem Murray et al. (2012); Datta et al. (2006).

PhotoBot, our automated robot photographer, is capable of interacting with the user by leveraging the reasoning capabilities of large language models (LLMs) Talmor et al. (2021) and the grounding capabilities of visual language models (VLMs) together. Specifically, we first convert a curated gallery of images into text-based descriptions (e.g., including information such as a general description of the image, the mood, and the number of people in the image) using a VLM and an object detector. The VLM and object detector provide an automated approach to describe curated images using language. Given a language query from a user and the detected objects in the scene observed by the camera, we use a LLM to retrieve a relevant *reference image* (i.e., an existing image from the curated gallery of

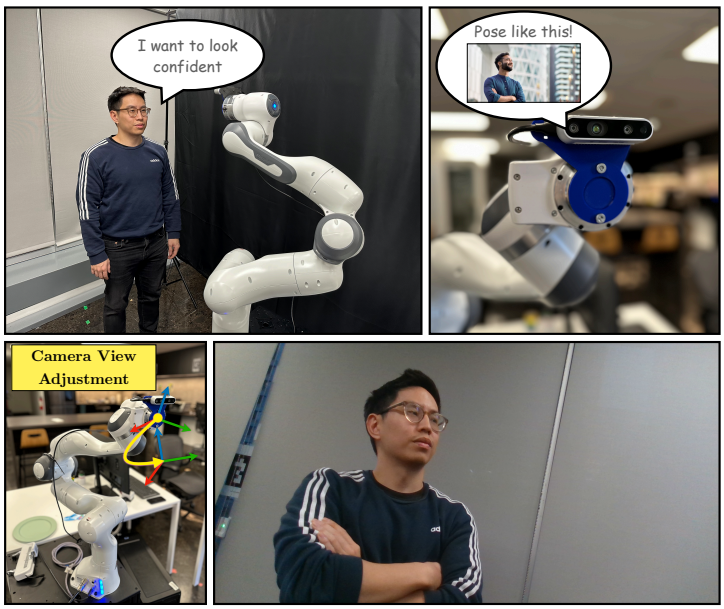


Figure 6.1: PhotoBot provides a reference photograph suggestion based on an observation of the scene and a user's input language query (upper left). The user strikes a pose matching that of the person in the reference photo (upper right) and PhotoBot adjusts its camera accordingly to faithfully capture the layout and composition of the reference image (lower left). The lower-right panel shows an unretouched photograph produced by PhotoBot.

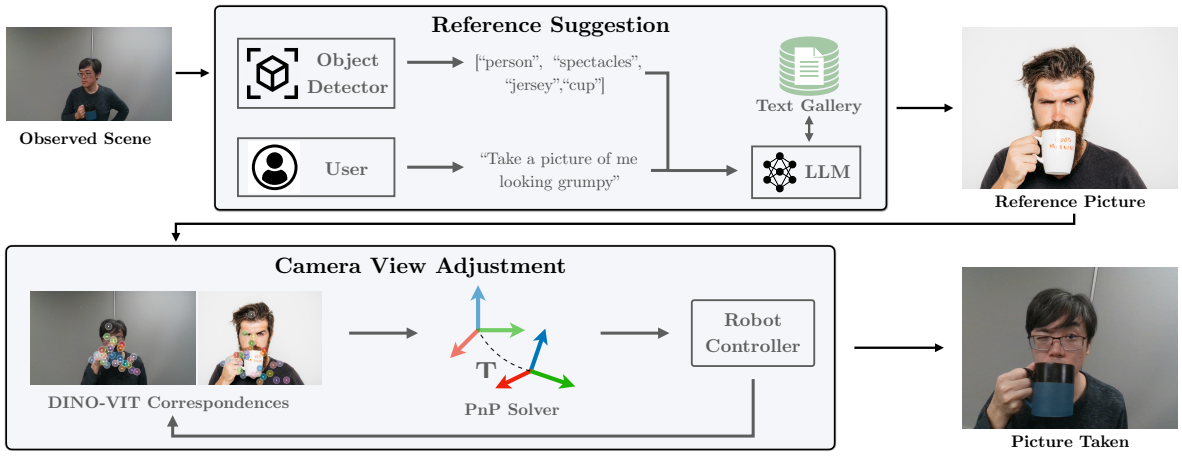


Figure 6.2: PhotoBot system diagram. The two main modules are shown: Reference Suggestion and Camera View Adjustment. Given the observed scene and a user query, PhotoBot suggests a reference image to the user and adjusts the camera to take a photo with a similar layout and composition to the reference image.

high-quality photographs) to suggest to the user through text-based reasoning. The user then imitates what is shown in the image and PhotoBot solves for the respective camera motion and image crop such that the camera view matches the reference image. We formulate camera view adjustment as a perspective-n-point (PnP) problem (Fischler and Bolles, 1981a) with pretrained features from a vision transformer capable of capturing semantic similarity across significantly varying images (Amir et al., 2022; Caron et al., 2021).

The learned components of PhotoBot operate together to yield a scaleable and generalizable system that can capture subjective and difficult-to-quantify aesthetic preferences through natural interaction. Fig. 6.1 shows an example of a problem instance and a photograph produced fully automatically by PhotoBot.

6.2 Related Work

Robot photography has previously been studied in the context of mobile robotics. Early work (Byers et al., 2003a,b) introduced a mobile robotic system that navigates, detects faces, and takes photographs based on hand-engineered composition rules. In (Kim et al., 2010), a combination of sound and skin detection is used to frame subjects. An autonomous robotic photographer based on the low-cost Turtlebot platform is introduced in (Zabarauskas and Cameron, 2014), where head detection and handcrafted photography composition rules are applied with a subsumption control module to capture photos. KL divergence optimization is used in (Hoque et al., 2022) to evaluate composition quality by comparing the distribution of the facial position and direction of photographed human users with a target distribution following composition rules. Within the context of mobile robotics, much of the prior work have focused on the technical navigation and control problems and not on the social aspect (i.e., the interaction between the photographer and human subject) as tackled in our work.

The authors of (Campbell and Pillai, 2005) introduce a method to frame human and nonhuman subjects based on motion parallax and optical flow techniques in posed and cooperative settings. Methods such as (Gadde and Karlapalem, 2011) have also studied generalizing autonomous photography to scenes that do not contain human subjects using varied aesthetic criteria. Similar to (Campbell and Pillai, 2005) and (Gadde and Karlapalem, 2011), our PhotoBot framework generalizes to nonhuman subjects. We also consider the posed and cooperative setting as done in (Campbell and Pillai, 2005). Learning has been used in robot photography to formulate aesthetic models (Newbury et al., 2020) and, more directly, in approaches using reinforcement learning to find policies that optimize a reward function based on aesthetics (AlZayer et al., 2021).

User or subject interactivity in the context of robot photography is addressed in (Ahn et al., 2006) by having the robot move towards users who are waving their hands. In our work, we are also interested in user interactivity. However, unlike in (Ahn et al., 2006), which focuses on getting the attention of the robot photographer, we focus on a different task of suggesting photo ideas to the user. Closer to our approach, (Rivkin et al., 2023) uses a LLM to produce text descriptions of photos that a photographer would typically be expected to take at an event that is described at a high level. A VLM is then used to find the best image matches to these text descriptions from a video stream of a camera on a robot. In our work, we instead focus on producing photo suggestions based on personal user queries and scene observations. We provide reference images as suggestions and not textual descriptions of photos as done in (Rivkin et al., 2023). Our search and match procedure is done purely in text space using a LLM and text descriptions of existing images in a gallery as opposed to using a VLM. Using a LLM allows more sophisticated reasoning and better explainability. In (Bao et al., 2023), the authors use a LLM

to generate trajectories that capture specific videos and photographs based on user language queries. The authors program various camera movement primitives ahead of time that the LLM can then call. Similar to our work, an object detector is used to ground the LLM into the real world. However, we leverage the LLM for photography curation and not control.

The social dimension of photography has also been explored beyond the realm of natural language alone. In (Adamson et al., 2020), humorous content is displayed by a robot photographer to elicit spontaneous smiles. In (Li et al., 2023a), the authors design a system to autonomously capture how-to videos and related photographic content; the robot photographer detects body, hand, and text cues to determine which regions of interests to track. We focus on recreating reference images, which is more challenging from a semantic content perspective but simultaneously allows us to simplify the camera view adjustment problem as a PnP problem.

6.3 PhotoBot

Our proposed pipeline consists of two modules: a reference suggestion component that retrieves reference images for the user to imitate and a camera view adjustment component that alters the camera view to achieve a similar photo composition to the reference. An overview of the system is shown in Fig. 6.2. We describe the overall user interaction workflow in Section 6.3.1, the reference image selection process in Section 6.3.2, the extraction of semantic correspondences in Section 6.3.3, viewpoint alignment in Section 6.3.4 and outlier removal in Section 6.3.5.

6.3.1 User Interaction Workflow

The steps in our proposed workflow for reference-guided robot photography are as follows.

1. The user issues a query (e.g., "Take a picture of me looking happy.").
2. PhotoBot detects objects and the number of people in the environment.
3. Taking both user query, detected objects and number of people into consideration, the system retrieves a shortlist of relevant reference images from a curated gallery.
4. The user selects a preferred reference image from the shortlist.
5. The user imitates what is shown in the selected reference image.
6. PhotoBot moves the camera such that the view matches that of the reference image.
7. PhotoBot captures the photo and crops it to match the reference image aspect ratio.

In the sections below, we describe each component and step in more detail.

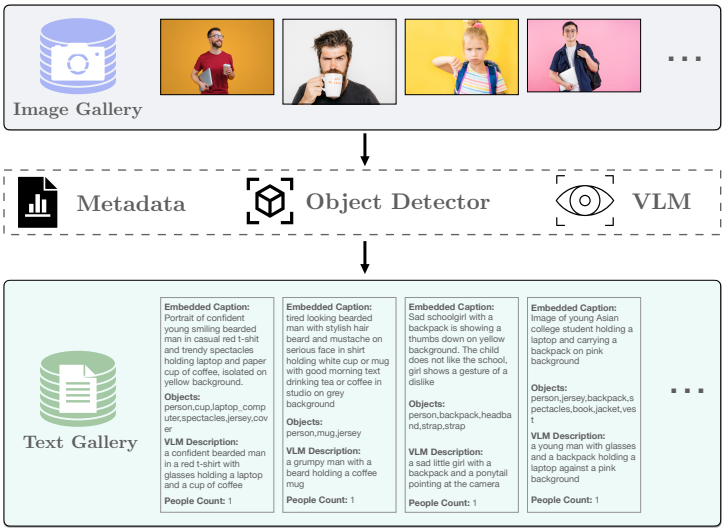


Figure 6.3: We convert a gallery of curated reference images into a text-based representation using a combination of readily-available metadata, an object detector, and a VLM. A text-based gallery enables a LLM to search, match, and suggest reference images based on a language query from a user and a list of detected objects in the current scene.

Objects Detected	User Query	LLM Response
["person", "spectacles", "jersey","cup"]	"Take a picture of me looking grumpy"	 **LLM**: I chose this picture because it shows a grumpy or tired looking bearded man holding a cup, which matches the objects available in your current scene.
["person", "spectacles", "backpack","jersey"]	"Take a picture of me with a laptop"	 **LLM**: I chose this picture because it also features a person with a laptop, matching your query. The person is also carrying a backpack, which matches one of the objects available in your current scene. The spectacles worn by the person in the picture can be used to re-create the image.
["person", "spectacles", "backpack","jersey"]	"Take a picture of me looking sad"	 **LLM**: I chose this picture because it features a sad person with a backpack, which matches the objects available in the current scene.
N/A	"Take a picture of me looking surprised"	 **LLM**: I chose this picture because it features a portrait of a young man with a shocked facial expression, similar to what you described.
["person", "person", "spectacles", "spectacles" "jersey", "jersey"]	"Take a picture of us"	 **LLM**: I chose this picture because it features two people wearing spectacles, which matches the objects available in the current scene. The business partners posing in front of a gray background will create a professional and sophisticated image.

Figure 6.4: Examples of user queries and objects detected in the scene and the resulting reference image suggested by the LLM. We explicitly query the LLM to explain its choice of image suggestion. We can also query for an image suggestion without any information from the observed scene, as shown in the fourth row.

6.3.2 Reference Suggestion

To prepare the curated reference image gallery mentioned in Step 3 of Section 6.3.1, we obtain a set of high-quality, professionally taken photos from the Internet. We preprocess each photo by querying a VLM to describe the image and count the number of people in it. In addition, we use an object detector to identify all objects that are present. If available, we also extract textual data embedded in the metadata of the photo We use Detic (Zhou et al., 2022) as our object detector and InstructBLIP (Dai et al., 2023) as our VLM. Finally, the VLM description, object list, metadata, and people count are concatenated into a single textual caption, which we embed into a vector using a sentence transformer (Reimers and Gurevych, 2019). We visualize this procedure in Fig. 6.3. The embedding vectors of all reference images are entered into a vector database for efficient and scalable retrieval.

At execution time, we similarly create a textual embedding of the user prompt, which consists of the user's query from Step 1, detected objects in the current view, and the people count for the current view. Using the vector database, we retrieve the top m most similar image descriptions. Many of these image descriptions are only coarsely related to the user query and the current image because the sentence transformer typically does not have sufficient representational capacity for detailed reasoning. In turn, we feed the m texts, as well as the user prompt into GPT-4 (OpenAI, 2023), and ask GPT-4 to find the most m^* relevant captions, where $m^* \ll m$. The initial coarse matching based on embedding vectors is necessary since GPT-4 has a prompt character limit and a text description of every gallery image would not fit within the limit. The reference images associated with the m^* captions are then provided to the user in Step 4. Examples of user queries, objects detected in the scene, and the resulting LLM reference image suggestions (with an explanation from the LLM) are shown in Fig. 6.4. We used $m^* = 3$ and $m = 16$ in this work. We set $m^* = 1$ to have a single suggestion visualized in Fig. 6.4 only.

6.3.3 Semantic Keypoint Correspondence

Since the reference image is captured from a different scene, and contains object instances with highly different appearances, traditional local appearance-based features (e.g., SIFT (Lowe, 2004)) are inadequate for establishing correspondences. To address this challenge, we exploit recent advances in self-supervised vision transformers to extract high-level semantic correspondences between the current and reference views. We follow the approach of Amir et al. (Amir et al., 2022) to establish semantic correspondence between a reference image and the current view captured by our RGB-D camera. The reference image is taken from a different scene, but it is assumed that the current scene and the reference image contain semantically-similar elements. To extract features from an image, we feed the image into a pretrained DINO-ViT transformer (Caron et al., 2021) and use the keys from intermediate transformer layers as dense image descriptors. Each key can be interpreted as a descriptor for the image patch associated with the corresponding token. The intermediate layers have been shown to offer a good trade-off between semantic and position information, both of which are important for semantic keypoint matching (Amir et al., 2022). Additional context is added to each descriptor by aggregating descriptors from adjacent patches via log binning. Having extracted descriptors from both the reference image and the RGB channels of the current view, we identify Best-Buddies Pairs (BBPs)

(Oron et al., 2017) between the descriptors from the two views to find correspondences.

Next, we select k salient and well-distributed correspondences for use when solving the PnP problem. To achieve this, we concatenate the descriptors of each correspondence pair and run K-means on the concatenated descriptors using k as the number of clusters. Finally, we select the single most salient correspondence from each cluster to form the final set of k correspondences. Saliency is computed by averaging CLS attention heads in the last layer of the transformer for the image patches associated with the keypoint pair. Since depth is available, we back-project the 2D keypoints to obtain 3D keypoints for the current view.

6.3.4 Camera View Adjustment

We formulate the view adjustment problem as a PnP problem. Given 3D keypoints from the current view and the corresponding 2D keypoints from the reference image, we can solve for the camera transformation that would adjust the current viewpoint to align with that of the reference image. Let $\mathbf{x}_i = [x_i, y_i, 1]$ be the 2D homogeneous coordinates of the i-th keypoint in the reference image, and $\mathbf{X_i} = [X_i, Y_i, Z_i, 1]$ be the 3D homogeneous coordinates of the corresponding keypoint in the current view. The PnP problem is to find a 3D camera transformation, consisting of a rotation $\mathbf{R} \in \mathrm{SO}(3)$ and a 3D translation $\mathbf{t} \in \mathbb{R}^3$, such that the sum of squared reprojection errors ε_i, $i = 1, \ldots, n$,

$$\varepsilon_i = ||\mathbf{x_i} - \mathbf{K}[\mathbf{R} \mid \mathbf{t}]\mathbf{X_i}||_2, \tag{6.1}$$

is minimized. Here, $\mathbf{K}$ is the 3×4 intrinsic matrix of the camera that we control. The intrinsic parameters of the camera used to take the reference image are not required, and we assume that they are not known. Our implementation uses the popular EPnP (Lepetit et al., 2009) method, which provides a closed-form solution given at least four correspondences.

The reference image typically has a different resolution and aspect ratio compared to those captured by our camera. Our proposed approach is to output a photo that has the same aspect ratio as the reference image and at the highest resolution possible. To achieve this, we scale the reference image to be as large as possible without changing its aspect ratio, subject to the constraint that its width and height do not exceed that of our captured image. Then, we pad the reference image as needed such that it has the same dimensions as the captured image. After a photo is taken, we can simply crop out the padded region to obtain a final image with the same aspect ratio as the reference. Fig. 6.5 illustrates the padding added to the reference images and the final, cropped photos. Intuitively, scaling the reference image in this way ensures that the PnP solver outputs a camera pose where the reference visual elements are centered and fill the field of view to the extent possible.

6.3.5 Outlier Removal

Typically, a subset of the feature correspondences are erroneous. Often, this is due to different parts of the scene having similar appearances. For our problem, spurious correspondences may also arise when different parts of the scene having similar semantic interpretations. In turn, PnP solvers are usually

used in conjunction with a robust estimator such as RANSAC (Fischler and Bolles, 1981b) to remove spurious matches (i.e., outliers) and produce a solution based on only a subset of mutually-coherent matches (i.e., inliers). That is, Eq. (6.1) should be solved using the inliers only.

In a typical RANSAC framework for PnP, minimal samples of three to four correspondences are drawn and used to estimate a candidate PnP solution. Then, all correspondences are considered based on the candidate solution: the ones that are consistent with the candidate solution are considered as inliers and the rest are considered as outliers. Finally, the PnP solution with the largest number of inliers is selected and further refined using all of the inliers. Determining the consistency between correspondences and a candidate solution is a key aspect of RANSAC. Usually, a correspondence j is considered an inlier if the reprojection error ε_j falls below some threshold τ, which must be carefully tuned. Most applications of PnP assume that all corresponding points are captured from the same scene, and so it is often sufficient to fix τ to be a small value (e.g., 10 pixels). In our problem setting, however, the 2D points from the reference image are not captured from the same scene as the 3D points (which come from from the camera's current view). Furthermore, a person may pose more similarly to a reference image than another. Since the degree of discrepancy between the current scene and the reference varies between problem instances, common methods for filtering out spurious correspondences are difficult to apply. In Section 6.4, we show that the best RANSAC threshold varies from one problem instance to another.

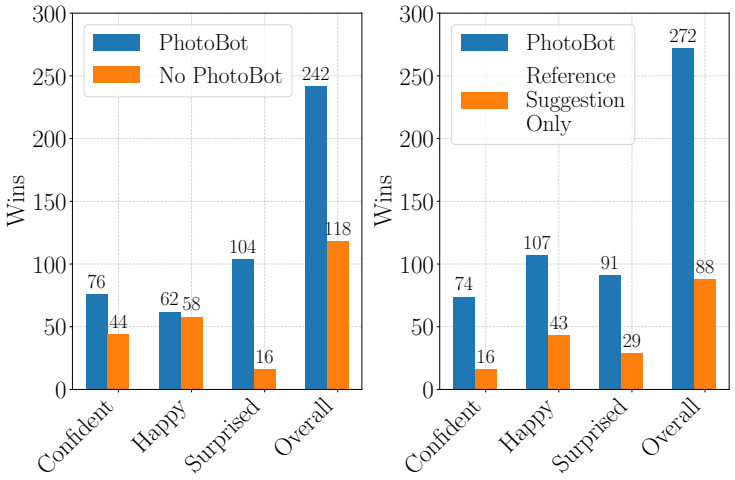
To address the challenges associated with threshold selection, numerous robust estimators have been developed that reduce the need for threshold tuning (Stewart, 1995; Torr and Zisserman, 2000; Moisan and Stival, 2004; Barath et al., 2020). We use MAGSAC++ (Barath et al., 2020), a state-of-the-art robust estimator that does away with the hard inlier threshold. For our PnP problem, MAGSAC++ solves a weighted least squares problem using all correspondences to minimize Eq. (6.1). The weight of each correspondence is based on the expected likelihood of it being an inlier when marginalized over all threshold levels up to a maximum threshold. In this way, MAGSAC++ only requires specifying a maximum threshold, which can be loosely set to a large value.

In practice, we find it useful to have the system make multiple successive camera adjustments before taking the final photo. As the camera approaches the correct viewpoint, the quality of keypoint matches tends to improve, and the number of inliers tends to increase. Thus, the PnP solution will typically improve beyond the initial camera adjustment. In our implementation, we terminate the process when the mean distance in pixels between the k keypoint correspondences does not improve for two iterations.

6.4 Experiments

We evaluated the PhotoBot framework using a real Franka Emika robot manipulator equipped with a RealSense D435 RGB-D camera. We deployed PhotoBot and took photos of various scenes involving both humans and objects. First, we conducted a user study and evaluated the effectiveness of reference suggestions and the view adjustment procedure. Second, we studied the effects of the crucial RANSAC

Figure 6.5: Photos of users evoking various emotions. The user prompts, from top to bottom, are *surprised*, *confident*, *guilty*, *confident*, *happy*, and *confident*. Columns, from left to right, are: user's own creative posing; user mimicking the suggested reference using a static camera; photo taken by our PhotoBot system; and reference image suggested by PhotoBot. The checkered background indicates cropping. The black background indicates padding of the reference image to facilitate the PnP solution. PhotoBot automatically crops the photos it takes to match the image template.



(a) User preference study. **(b)** View adjustment evaluation.

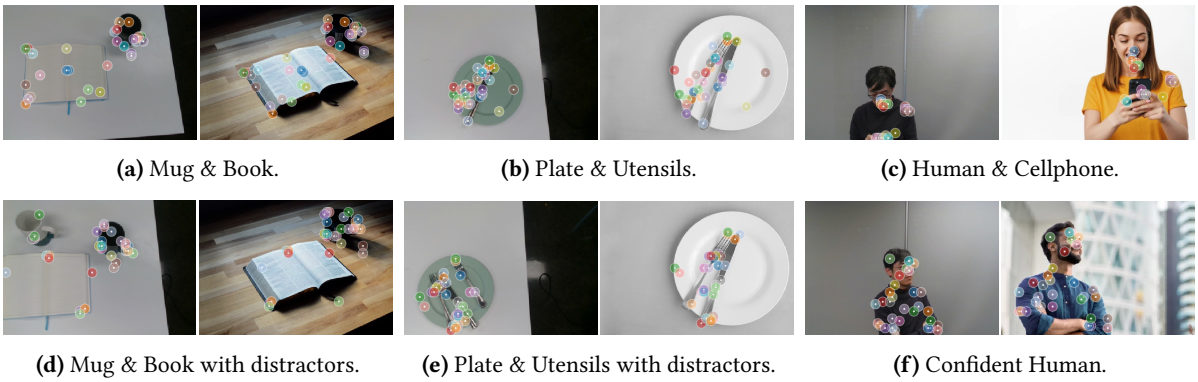
Figure 6.6: Results from user studies. (a) User preference study based on the overall aesthetic quality and how well the user prompt is addressed by photos taken by PhotoBot and the No PhotoBot baseline. (b) View adjustment evaluation based on how close the photos taken by PhotoBot and the Reference Suggestion Only baseline are to the actual reference image in terms of the camera viewing angle and the photo layout. We present results for each category of emotion, and the aggregated value combining all emotions.

inlier reprojection error threshold τ on the quality of the PnP solution when using DINO-ViT features. Third, we then investigated the quality of solutions as a function of the number of keypoints k. Finally, we qualitatively tested whether PhotoBot is able to generalize to reference images with significantly larger distribution shifts (e.g., paintings).

6.4.1 Human Preference Evaluation

To critically assess the reference-based photography approach used in this work, we had a group of users ($N = 8$) interact with and have their photos taken by PhotoBot. We prompted users to query PhotoBot for image suggestions from three general categories of emotions: *confident*, *happy*, and *surprised*. Based on each query, three reference images suggestions were provided, using the procedure detailed in Section 6.3.2, from which the user chose one. For this experiment, we used a gallery of 75 images in total with images ranging from a variety of emotions. Users then posed in a manner similar to the reference image (to the best of their ability) and PhotoBot took their photos while using the camera view adjustment procedure outlined in Section 6.3.4.

We first investigated whether PhotoBot takes aesthetically pleasing photos that satisfy (or match) the query from the user. As a baseline, which we name "No PhotoBot," we asked the same set of users to come up with a gesture and expression on their own that matched the category of emotion and to pose for a photo in front of a static camera. Users took the "No PhotoBot" photos before interacting with PhotoBot so that they were not influenced by any reference images. Similar to taking a 'selfie,' we allowed users to view the image from the fixed camera as they posed. Examples of "No PhotoBot" are shown in the first column of Fig. 6.5. We then surveyed a separate group of individuals ($M =$



(a) Mug & Book. **(b)** Plate & Utensils. **(c)** Human & Cellphone.

(d) Mug & Book with distractors. **(e)** Plate & Utensils with distractors. **(f)** Confident Human.

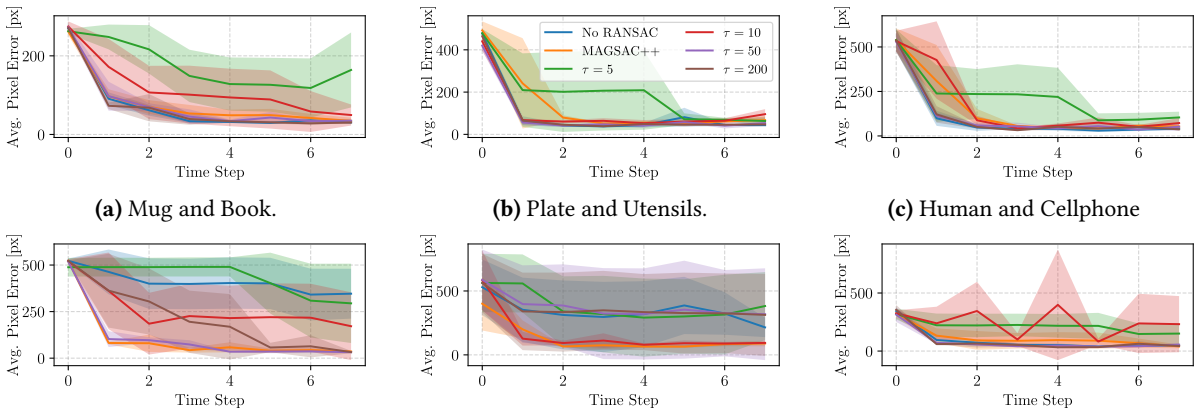
Figure 6.7: Scenes used to analyze and evaluate RANSAC inlier threshold with the learned semantic-level keypoints (i.e., DINO-ViT) used in the PhotoBot view adjustment procedure. For each pair, an initial image captured by PhotoBot is shown on the left and the reference image is shown on the right. We identify the point correspondences between image pairs with matching colours.

20), unrelated to this work, to evaluate the aesthetic quality of the photos from PhotoBot and "No PhotoBot." We simultaneously presented the surveyed users with two photos, one from PhotoBot and one from "No PhotoBot," and asked them to pick the photo that was both (1) more aesthetically pleasing and (2) better addressed the query from the user. We visualize the results from the survey in Fig. 6.6a, separated by the categories of emotions, as well as the aggregated result over all three categories. A vote for a photo consisted of a "win" for that particular method. PhotoBot significantly outperforms the "No PhotoBot" baseline in two out of three categories of emotions, *confident* and *surprised* and performs nearly on par for the *happy* category. Although aesthetics is subjective, our results show that the combination of reference suggestion and view adjustment from PhotoBot generally leads to more aesthetically pleasing photos that better address a user query.

As a second experiment, we explored whether the PhotoBot view adjustment procedure leads to photos that better match the reference images. We use a baseline called "Reference Suggestion Only," where we asked users to try their best to position themselves and pose in front of the static camera in a manner that re-created the reference image as closely as possible. Examples of "Reference Suggestion Only" are shown in the second column of Fig. 6.5. For the PhotoBot picture, we used the "Reference Suggestion Only" pose from the user as the initial image for view adjustment. We surveyed the same 20 individuals as before by simultaneously presenting a photo from PhotoBot, a photo from the "Reference Suggestion Only" baseline, and the reference image itself. We then asked each individual to choose the photo that best matched the reference image in terms of the viewing angle and the layout. We visualize the results from the survey in Fig. 6.6b, separated by the specific categories of emotions, as well as the aggregated result over all three categories. PhotoBot outperforms the "Reference Suggestion Only" baseline in all three categories.

6.4.2 RANSAC Inlier Threshold Evaluation

We experimentally evaluated and analyzed the effect of the RANSAC inlier reprojection error threshold τ on the quality of PnP solutions when using DINO-ViT keypoints. As previously highlighted in



(a) Mug and Book. **(b)** Plate and Utensils. **(c)** Human and Cellphone

(d) Mug and Book with distractions. **(e)** Plate and Utensils with distractors. **(f)** Confident Human

Figure 6.8: Average absolute pixel errors with various RANSAC methods for four different template images and scenes. We measure average pixel error over all keypoints for eight time steps of view adjustment. The shaded region consists of one standard deviation measured over 3 repeated runs. The best-performing inlier reprojection error threshold τ varies between scenes. The adaptive MAGSAC++ algorithm performs close to the best fixed-threshold parameter with RANSAC.

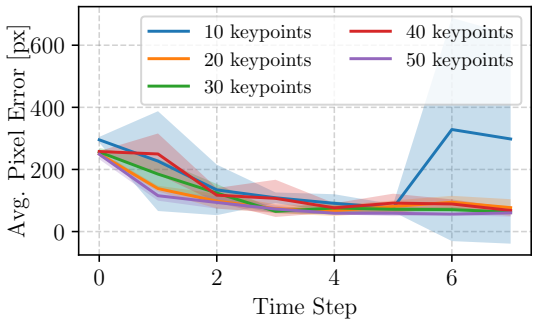
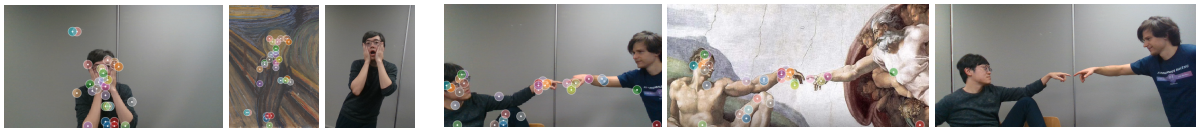


Figure 6.9: Pixel errors at each time step for various numbers of DINO-ViT keypoints used. The shaded region consists of one standard deviation measured over three repeated runs.



(a) Using "The Scream" as an image template. **(b)** Using "The Creation of Adam" as an image template.

Figure 6.10: PhotoBot can generalize to paintings used as reference images. For each image set above, the first and second images (left to right) show the initial set of correspondences found, while the third image is the photo with the highest-scoring alignment found by PhotoBot. The large distribution shift between the images introduces additional outliers.

Section 6.3, finding semantic keypoint correspondences between different scenes poses an interesting challenge when compared to finding correspondences between local appearance-based features of the same scene. In particular, the magnitude of reprojection errors and the number of outliers can vary significantly across different scenes and users may pose at varying levels of similarity to the reference image.

We considered the scenes shown in Fig. 6.7 in our experiments. We chose four settings consisting of static objects ("Mug and Book," "Mug and Book with Distractions," "Plate and Utensils," and "Plate and Utensils with Distractions") to be able to test in a consistent and reproducible manner. For each set of objects, we included a version with and without distractor objects that introduce additional outliers. Distractor objects are repeated objects in the scene not present in the template image (e.g., an extra mug or fork) that induce false correspondences. We also considered two scenes involving humans ("Human and Cellphone" and "Confident Human"). For each reference image, we re-created the scene with similar objects or with a human model and then ran our view adjustment procedure. We tested a total of six methods: RANSAC with four different fixed threshold pixel values of $\tau = \{5, 10, 50, 200\}$, No RANSAC, and MAGSAC++ with a maximum error threshold of 50 pixels. For each method, we ran view adjustment for eight successive steps and repeated the experiment three times for each scene. At each step, if the PnP solver did not find a solution or the suggested pose was outside the workspace of the robot, then the robot did not move for that step.

Fig. 6.8 displays the results for each respective scene and reference image. We measured the error between the reference image and the current image by manually annotating the ground truth correspondences at each step. During annotation, it was up to the human annotator to select keypoints that were salient and meaningful (e.g., the tip of the fork or the handle of the mug). We observed that the best-performing fixed threshold differs for each scene. Furthermore, for DINO-ViT features, larger error thresholds are required for PnP to produce solutions that converge. In all scenes, setting a threshold value of $\tau = 5$ caused the PnP solution to frequently fail to converge, which increased the overall error in the trial. For all of the tested scenes, the adaptive MAGSAC++ algorithm performs similarly to the best variant of RANSAC with a fixed threshold.

6.4.3 Required Number of Keypoints Evaluation

We evaluated the quality (i.e., average pixel error as measured in Section 6.4.2) of the camera view adjustment procedure of PhotoBot as a function of the number of DINO-ViT keypoints. As previously

discussed in Section 6.3.3, the number of keypoints k is a hyperparameter that is set based on the number of chosen K-means clusters. We ran PhotoBot on the Mug and Book scene shown in Fig. 6.8a while varying the number of keypoints used, following the same experimental procedure in Section 6.4.2. The results are visualized in Fig. 6.9. Interestingly, we observed that PhotoBot performs reasonably well when at least 20 keypoints are used. The variant with 10 keypoints often diverged due to insufficient keypoint coverage across the entire scene.

6.4.4 Generalizing to Other Reference Images

We also carried out an evaluation with paintings, rather than photographs, to determine if PhotoBot is capable of generalizing to references from different mediums. Notably, DINO-ViT features have been shown to generalize across large distribution shifts (Caron et al., 2021; Amir et al., 2022). In Fig. 6.10, we show results using two famous paintings as references. Qualitatively, we observe that, despite the changes in medium and format, PhotoBot is still capable of finding snapshots that resemble the paintings. The large distribution shift introduces additional outliers, which MAGSAC++ is able to filter out successfully.

6.5 Summary and Future Work

In this chapter, we presented PhotoBot, a novel interactive robot photography framework. PhotoBot is capable of suggesting reference images based on a natural language query from a user and visual observation of the current scene. In addition, PhotoBot can adjust the camera to match the layout and composition of a chosen reference image. We conducted experiments demonstrating that the photos taken by PhotoBot are aesthetically pleasing, address the users' prompts, and follow the reference images' layouts and compositions. We also studied various factors that affect the quality of the photos taken and showed that PhotoBot is able to generalize to other references sources, including paintings.

In the previous chapters we demonstrated how generative SSL can be used as a framework to solve robotic problems. In this chapter, we showed how generative models can also be used as stand alone modules to enhance robotic systems. Representations learned via generative SSL with images can be used as generalizable semantic features. Models trained via generative SSL with text can be used as a language interface in robotic system for better human-robot interactions.

As future work, we aim to study alternate physical embodiments with a wider range of motion (e.g., a quadcopter or a mobile manipulator), and to develop methods to provide language-based corrective posing feedback to the user.

6.6 Associated Publications

1. Limoyo, O., Li, J., Rivkin, D., Kelly, J., and Dudek, G. (2024c). Photobot: Reference-guided interactive photography via natural language. In *IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS)*, Abu Dhabi, United Arab Emirates. Submitted

2. Limoyo, O., Li, J., Rivkhin, D., Kelly, J., and Dudek, G. (2024b). Reference-guided robotic photography through natural language interaction. In *Proceedings of the ACM/IEEE International Conference on Human Robot Interaction (HRI) Workshop on Human—Large Language Model Interaction: The Dawn of a New Era or the End of it All?*, Boulder, Colorado, USA

Chapter 7

Conclusion

This dissertation explored a generative, self-supervised paradigm for robot learning. We presented two reformulations of robotic problems as generative SSL problems: a reformulation of state space modelling in Chapter 3 and of inverse kinematics in Chapter 4. In Chapter 5, we demonstrated how the symmetry of the pick and place tasks can be exploited to create a self-supervised data collection pipeline for object placement in contact-constrained environments. Finally, in Chapter 6, we showed how vision and language modules trained purely via self-supervision can be used to enhance robotic systems and algorithms. We used representations learned by a network via self-supervision with images as semantic features for camera pose estimation and a language model trained with text as a language interface for better human-robot interaction. Our hope is that these contributions have strengthened the connection between generative SSL and robotics by identifying common threads and shared theoretical roots. We conclude below with a summary of the novel contributions associated with each chapter and with a discussion of potential directions for future work.

7.1 Summary of Contributions

The novel contributions of this dissertation are summarized on a per chapter basis below.

7.1.1 Learning Deep State Space Models from Observations

We introduced an approach to learn robust and multimodal state space models directly from observations. We contributed:

1. a self-supervised method for learning a latent state representation and the associated state space model while capturing a notion of heteroscedastic uncertainty at test time via novelty detection;
2. a demonstration of image-based prediction and closed loop control on a real-world robotic system using the learned latent dynamics that is robust to input degradations;
3. a probabilistic formulation for self-supervised training of latent state space models with multimodal time series data; and

4. open source code for both structured and multimodal sequential latent variable model learning.

7.1.2 Generative Graphical Inverse Kinematics

We presented a generative graphical inverse kinematics model and solver called GGIK. The novel contributions were:

1. a generative, graph-based inverse kinematics formulation that can represent general (i.e., not tied to a single robot manipulator model or geometry) IK mappings and produce approximations of entire feasible sets of IK solutions;

2. an extensive experimental evaluation demonstrating GGIK's ability to: produce accurate solutions, generalize to unseen manipulation, account for joint limits, initialize numerical solvers and produce multiple solutions quickly and in parallel on a GPU;

3. an empirical demonstration that capturing the symmetry of the graphical IK formulation problem via an equivariant network architectures achieves state-of-the-art accuracy; and

4. open source code for the model and to reproduce the experiments.

7.1.3 Learning to Place by Picking

We formulated a method called placing via picking (PvP) that leverages the symmetry of the picking and placing tasks to acquire expert placing demonstrations without human labour. In this work, we contributed:

1. PvP, a self-supervised data collection method for 6-DOF robotic object placements;

2. a compliant grasping and a tactile-driven regrasping strategy to achieve uninterrupted pick and place data collection without human intervention in contact constrained environments;

3. a language-specified grasp planning pipeline; and

4. real-world experimental validation of PvP using the collected demonstrations to train a vision-based policy capable of placing multiple plates in a dish rack and of setting a table.

7.1.4 Language-Driven Robotic Photography

Finally, we demonstrated how vision and language models trained solely by self-supervision can be integrated into a robotic photography system that produces visually appealing photos by interacting with the user and scene via the reasoning and grounding capabilities of LLMs and VLMs respectively. Our contributions were:

1. the formulation of a new imitation-based approach to robot photography, called PhotoBot, using reference image templates;

2. a novel grounded photo suggestion module building on a combination of a visual language model (VLM), an object detector, and a large language model (LLM); and

3. a user study evaluating the quality of photos taken by PhotoBot.

7.2 Future Research Directions

Learned distributions form an important part of generative SSL (Chapter 3, Chapter 4, and Chapter 5). In our work, we focused on a narrow class of distributions, that is, Gaussian or mixture of Gaussians. This limits the flexibility of the learned model—various other parametrized distribution classes could be investigated. Alternatively, diffusion models (Song and Ermon, 2019; Sohl-Dickstein et al., 2015; Ho et al., 2020), which are able to learn more flexible distributions by not requiring an explicit parametrization, are also a promising alternative.

In many of the methods that we applied, we used random sampling to generate data from which we then trained our self-supervised models (e.g.,random actions to produce trajectories in Chapter 3, or random configurations to produce poses in Chapter 4). Investigating better methods to source or to collect data for SSL in robotics remains an interesting future research direction. This need partly inspired some of the work in Chapter 5, where we leveraged a grasp planner to generate demonstration data from which we then learned a robotic placement policy. Other promising directions to tackle data generation include further studying exploration policies (Pathak et al., 2017) and simulators combined with Sim2Real methods (Valassakis et al., 2021).

Finally, computer vision and natural language processing are two fields that have experienced enormous and undeniable progress due to breakthroughs in SSL. Beyond just reframing classical robotic problems in terms of SSL, we believe that an exciting and pragmatic research avenue will involve integrating learned models from robotics-adjacent fields to complement and enhance classical robotic systems and algorithms. Our robot photography framework introduced in Chapter 6 is one example of a step in this direction.

Bibliography

Ablett, T., Zhai, Y., and Kelly, J. (2021). Seeing all the angles: Learning multiview manipulation policies for contact-rich tasks from demonstrations. In *IEEE/RSJ Intl. Conf. Intelligent Robots and Systems (IROS)*, pages 7843–7850.

Adamson, T., Lyng-Olsen, C. B., Umstattd, K., and Vázquez, M. (2020). Designing social interactions with a humorous robot photographer. In *ACM/IEEE Intl. Conf. on Human-Robot Interaction (HRI)*, pages 233–241.

Ahn, H., Kim, D., Lee, J., Chi, S., Kim, K., Kim, J., Hahn, M., and Kim, H. (2006). A robot photographer with user interactivity. In *IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS)*, pages 5637–5643.

Ahn, M., Dwibedi, D., Finn, C., Arenas, M. G., Gopalakrishnan, K., Hausman, K., Ichter, B., Irpan, A., Joshi, N., Julian, R., Kirmani, S., Leal, I., Lee, E., Levine, S., Lu, Y., Leal, I., Maddineni, S., Rao, K., Sadigh, D., Sanketi, P., Sermanet, P., Vuong, Q., Welker, S., Xia, F., Xiao, T., Xu, P., Xu, S., and Xu, Z. (2024). AutoRT: Embodied foundation models for large scale orchestration of robotic agents. *arXiv preprint arXiv:2401.12963*.

AlZayer, H., Lin, H., and Bala, K. (2021). Autophoto: Aesthetic photo capture using reinforcement learning. In *IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS)*, pages 944–951.

Ames, B., Morgan, J., and Konidaris, G. (2022). Ikflow: Generating diverse inverse kinematics solutions. *IEEE Robotics and Automation Letters*, 7(3):7177–7184.

Amini, A., Schwarting, W., Rosman, G., Araki, B., Karaman, S., and Rus, D. (2018). Variational autoencoder for end-to-end control of autonomous driving with novelty detection and training de-biasing. In *IEEE/RSJ Intl. Conf. Intelligent Robots and Systems (IROS)*.

Amir, S., Gandelsman, Y., Bagon, S., and Dekel, T. (2022). Deep vit features as dense visual descriptors. In *ECCV Workshop on What is Motion For?*

Andrychowicz, M., Wolski, F., Ray, A., Schneider, J., Fong, R., Welinder, P., McGrew, B., Tobin, J., Pieter Abbeel, O., and Zaremba, W. (2017). Hindsight experience replay. In *Advances in Neural Information Processing Systems Deep Learning Symposium (NeurIPS)*, pages 5055–5065.

Ardizzone, L., Kruse, J., Rother, C., and Köthe, U. (2018). Analyzing inverse problems with invertible neural networks. In *Intl. Conf. on Learning Representations (ICLR)*.

Aristidou, A. and Lasenby, J. (2011). FABRIK: A fast, iterative solver for the inverse kinematics problem. *Graph. Models*, 73(5):243–260.

Aristidou, A., Lasenby, J., Chrysanthou, Y., and Shamir, A. (2018). Inverse kinematics techniques in computer graphics: A survey. *Computer Graphics Forum*, 37(6):35–58.

Ba, J. L., Kiros, J. R., and Hinton, G. E. (2016). Layer normalization. In *Advances in Neural Information Processing Systems Deep Learning Symposium (NeurIPS)*.

Baltrusaitis, T., Ahuja, C., and Morency, L. (2019). Multimodal machine learning: A survey and taxonomy. *IEEE Trans. on Pattern Analysis and Machine Intelligence (PAMI)*, 41(2):423–443.

Bambade, A., El-Kazdadi, S., Taylor, A., and Carpentier, J. (2022). Prox-qp: Yet another quadratic programming solver for robotics and beyond. In *Robotics: Science and Systems (RSS)*.

Banijamali, E., Shu, R., Ghavamzadeh, M., Bui, H. H., and Ghodsi, A. (2018). Robust locally-linear controllable embedding. In *Intl. Conf. Artificial Intelligence and Statistics (AISTATS)*.

Bao, Z., Zhu, G.-N., Ding, W., Guan, Y., Bai, W., and Gan, Z. (2023). A smart interactive camera robot based on large language models. In *IEEE Intl. Conf. on Robotics and Biomimetics (ROBIO)*, pages 1–6.

Barath, D., Noskova, J., Ivashechkin, M., and Matas, J. (2020). Magsac++, a fast, reliable and accurate robust estimator. In *IEEE/CVF Conf. on Computer Vision and Pattern Recognition (CVPR)*, pages 1304–1312.

Barfoot, T. D. (2024). *State estimation for robotics*. Cambridge University Press, 2nd edition.

Beeson, P. and Ames, B. (2015). TRAC-IK: An open-source library for improved solving of generic inverse kinematics. In *Intl. Conf. on Humanoid Robots (Humanoids)*.

Bensadoun, R., Gur, S., Blau, N., and Wolf, L. (2022). Neural inverse kinematic. In *Intl. Conf. on Machine Learning (ICML)*.

Bicchi, A. and Kumar, V. (2000). Robotic grasping and contact: A review. In *IEEE Intl. Conf. Robotics and Automation (ICRA)*, pages 348–353.

Billard, A. G., Calinon, S., and Dillmann, R. (2016). Learning from humans. *Springer Handbook of Robotics*, pages 1995–2014.

Bishop, C. (1994a). Mixture density networks. Technical Report NCRG/94/004, Dept. of Computer Science and Applied Mathematics, Aston University.

Bishop, C. M. (1994b). Novelty detection and neural network validation. *IEE Proceedings - Vision, Image and Signal processing*, 141(4):217–222.

Bishop, C. M. (2006). *Pattern Recognition and Machine Learning*. Springer.

Blattmann, A., Dockhorn, T., Kulal, S., Mendelevitch, D., Kilian, M., Lorenz, D., Levi, Y., English, Z., Voleti, V., Letts, A., et al. (2023). Stable video diffusion: Scaling latent video diffusion models to large datasets. *arXiv preprint arXiv:2311.15127*.

Blei, D. M., Kucukelbir, A., and McAuliffe, J. D. (2017). Variational inference: A review for statisticians. *Journal of the American statistical Association*, 112(518):859–877.

Bócsi, B., Nguyen-Tuong, D., Csató, L., Schoelkopf, B., and Peters, J. (2011). Learning inverse kinematics with structured prediction. In *IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems*. IEEE.

Bohg, J., Morales, A., Asfour, T., and Kragic, D. (2013). Data-driven grasp synthesis — a survey. *IEEE Transactions on robotics*, 30(2):289–309.

Bousmalis, K., Vezzani, G., Rao, D., Devin, C. M., Lee, A. X., Villalonga, M. B., Davchev, T., Zhou, Y., Gupta, A., Raju, A., et al. (2023). Robocat: A self-improving generalist agent for robotic manipulation. *Trans. Machine Learning Research*.

Boyd, S. and Vandenberghe, L. (2004). *Convex Optimization*. Cambridge University Press.

Brockman, G., Cheung, V., Pettersson, L., Schneider, J., Schulman, J., Tang, J., and Zaremba, W. (2016). OpenAI gym.

Bronstein, M. M., Bruna, J., Cohen, T., and Veličković, P. (2021). Geometric deep learning: Grids, groups, graphs, geodesics, and gauges. *arXiv preprint arXiv:2104.13478*.

Bruyninckx, H. (2002). Orocos–open robot control software.

Bullock, D., Grossberg, S., and Guenther, F. H. (1993). A self-organizing neural model of motor equivalent reaching and tool use by a multijoint arm. *Journal of Cognitive Neuroscience*, 5(4):408–435.

Byers, Z., Dixon, M., Goodier, K., Grimm, C., and Smart, W. (2003a). An autonomous robot photographer. In *IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS)*.

Byers, Z., Dixon, M., Smart, W. D., and Grimm, C. (2003b). Say cheese!: Experiences with a robot photographer. In *Conf. on Innovative Applications of Artificial Intelligence (IAAI)*, pages 65–70.

Campbell, J. and Pillai, P. (2005). Leveraging limited autonomous mobility to frame attractive group photos. In *IEEE Intl. Conf. on Robotics and Automation(ICRA)*, pages 3396–3401.

Cao, Y. and Fleet., D. J. (2014). Generalized product of experts for automatic and principled fusion of Gaussian process predictions. In *NeurIPS'14 Modern Nonparametrics 3: Automating the Learning Pipeline Workshop*.

Caron, M., Touvron, H., Misra, I., Jégou, H., Mairal, J., Bojanowski, P., and Joulin, A. (2021). Emerging properties in self-supervised vision transformers. In *IEEE/CVF Intl. Conf. on Computer Vision (ICCV)*, pages 9650–9660.

Chen, T., Kornblith, S., Norouzi, M., and Hinton, G. (2020). A simple framework for contrastive learning of visual representations. In *Intl. Conf. on Machine Learning (ICML)*.

Chi, C., Feng, S., Du, Y., Xu, Z., Cousineau, E., Burchfiel, B., and Song, S. (2023). Diffusion policy: Visuomotor policy learning via action diffusion. In *Robotics: Science and Systems (RSS)*.

Chi, C., Xu, Z., Pan, C., Cousineau, E., Burchfiel, B., Feng, S., Tedrake, R., and Song, S. (2024). Universal manipulation interface: In-the-wild robot teaching without in-the-wild robots. In *arXiv preprint arXiv:2402.10329*.

Chiappa, S. and Piquet, U. (2019). Unsupervised separation of dynamics from pixels. *METRON*, 77(2):119–135.

Cho, K., van Merriënboer, B., Bahdanau, D., and Bengio, Y. (2014). On the properties of neural machine translation: Encoder–decoder approaches. In *SSST-8 Eighth Workshop on Syntax, Semantics and Structure in Statistical Translation*.

Chung, J., Gulcehre, C., Cho, K., and Bengio, Y. (2014). Empirical evaluation of gated recurrent neural networks on sequence modeling. In *NeurIPS 2014 Workshop on Deep Learning*.

Chung, J., Kastner, K., Dinh, L., Goel, K., Courville, A. C., and Bengio, Y. (2015). A recurrent latent variable model for sequential data. In *Advances in Neural Information Processing Systems Deep Learning Symposium (NeurIPS)*.

Coumans, E. and Bai, Y. (2016–2019). Pybullet, a python module for physics simulation for games, robotics and machine learning. http://pybullet.org.

Dai, H., Izatt, G., and Tedrake, R. (2019). Global inverse kinematics via mixed-integer convex optimization. *Int. J. Rob. Res.*, 38(12-13):1420–1441.

Dai, W., Li, J., Li, D., Tiong, A. M. H., Zhao, J., Wang, W., Li, B., Fung, P., and Hoi, S. (2023). Instructblip: Towards general-purpose vision-language models with instruction tuning. In *Advances in Neural Information Processing Systems (NeurIPS)*.

Dalal, M., Mandlekar, A., Garrett, C. R., Handa, A., Salakhutdinov, R., and Fox, D. (2023). Imitating task and motion planning with visuomotor transformers. In *Conf. Robot Learning (CORL)*, pages 2565–2593.

Datta, R., Joshi, D., Li, J., and Wang, J. Z. (2006). Studying aesthetics in photographic images using a computational approach. In *European Conf. on Computer Vision (ECCV)*, pages 288–301.

De Cao, N. and Kipf, T. (2018). MolGAN: An implicit generative model for small molecular graphs. *ICML Workshop on Theoretical Foundations and Applications of Deep Generative Models*.

de Jalón, J. G. (2007). Twenty-five years of natural coordinates. *Multibody Sys. Dyn.*, 18(1):15–33.

Deisenroth, M. and Rasmussen, C. E. (2011). PILCO: A model-based and data-efficient approach to policy search. In *Intl. Conf. on Machine Learning (ICML)*.

Deng, J., Dong, W., Socher, R., Li, L.-J., Li, K., and Fei-Fei, L. (2009). Imagenet: A large-scale hierarchical image database. In *IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*.

Deo, A. S. and Walker, I. D. (1993). Adaptive non-linear least squares for inverse kinematics. In *IEEE Intl. Conf. on Robotics and Automation*.

Diamond, S. and Boyd, S. (2016). CVXPY: A Python-embedded modeling language for convex optimization. *Journal of Machine Learning Research*, 17(83):1–5.

Diankov, R. (2010). *Automated Construction of Robotic Manipulation Programs*. PhD thesis, Carnegie Mellon University, Pittsburgh, PA.

Doersch, C. (2016). Tutorial on variational autoencoders. *arXiv preprint arXiv:1606.05908*.

Doersch, C., Gupta, A., and Efros, A. A. (2015). Unsupervised visual representation learning by context prediction. In *IEEE/CVF Intl. Conf. on Computer Vision (ICCV)*.

Dong, S., Jha, D. K., Romeres, D., Kim, S., Nikovski, D., and Rodriguez, A. (2021). Tactile-rl for insertion: Generalization to objects of unknown geometry. *IEEE Intl. Conf. Robotics and Automation (ICRA)*, pages 6437–6443.

Elfwing, S., Uchibe, E., and Doya, K. (2018). Sigmoid-weighted linear units for neural network function approximation in reinforcement learning. *Neural Networks*, 107:3–11.

Engell-Nørregård, M. and Erleben, K. (2009). A projected non-linear conjugate gradient method for interactive inverse kinematics. In *Intl. Conf. on Mathematical Modelling (MATHMOD)*.

Erleben, K. and Andrews, S. (2019). Solving inverse kinematics using exact Hessian matrices. *Comput. Graph.*, 78:1–11.

Finn, C., Levine, S., and Abbeel, P. (2016). Guided cost learning: Deep inverse optimal control via policy optimization. In *Intl. Conf. Machine Learning (ICML)*, pages 49–58.

Fischler, M. A. and Bolles, R. C. (1981a). Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography. *Commun. ACM*, 24(6):381–395.

Fischler, M. A. and Bolles, R. C. (1981b). Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography. *Communications of the ACM*, 24(6):381–395.

Florence, P., Lynch, C., Zeng, A., Ramirez, O. A., Wahid, A., Downs, L., Wong, A., Lee, J., Mordatch, I., and Tompson, J. (2021). Implicit behavioral cloning. In *Conf. Robot Learning (CORL)*, pages 158–168.

Florence, P., Manuelli, L., and Tedrake, R. (2019). Self-supervised correspondence in visuomotor policy learning. *IEEE Robotics and Automation Letters (RAL)*, 5(2):492–499.

Fraccaro, M., Kamronn, S., Paquet, U., and Winther, O. (2017a). A disentangled recognition and nonlinear dynamics model for unsupervised learning. In *Advances in Neural Information Processing Systems Deep Learning Symposium (NeurIPS)*.

Fraccaro, M., Kamronn, S., Paquet, U., and Winther, O. (2017b). A disentangled recognition and nonlinear dynamics model for unsupervised learning. In *Advances in Neural Information Processing Systems Deep Learning Symposium (NeurIPS)*.

Fu, L., Huang, H., Berscheid, L., Li, H., Goldberg, K., and Chitta, S. (2023). Safe self-supervised learning in real of visuo-tactile feedback policies for industrial insertion. In *IEEE Intl. Conf. Robotics and Automation (ICRA)*, pages 10380–10386.

Gadde, R. and Karlapalem, K. (2011). Aesthetic guideline driven photography by robots. In *Intl. Joint Conf. on Artificial Intelligence - Volume Three (IJCAI)*, page 2060–2065.

Gal, Y. and Ghahramani, Z. (2016). Dropout as a Bayesian approximation: Representing model uncertainty in deep learning. In *Intl. Conf. on Machine Learning (ICML)*.

Gandhi, D., Gupta, A., and Pinto, L. (2020). Swoosh! Rattle! Thump!–Actions that sound. In *Robotics: Science and Systems (RSS)*.

Ghahramani, Z. and Hinton, G. E. (2000). Variational learning for switching state-space models. *Neural Computation*, 12(4):831–864.

Ghosh, D. (2018). Kl divergence for machine learning. https://dibyaghosh.com/blog/probability/kldivergence.html.

Giamou, M., Maric, F., Rosen, D. M., Peretroukhin, V., Roy, N., Petrovic, I., and Kelly, J. (2022). Convex iteration for distance-geometric inverse kinematics. *IEEE Robotics and Automation Letters*.

Gilmer, J., Schoenholz, S. S., Riley, P. F., Vinyals, O., and Dahl, G. E. (2017). Neural message passing for quantum chemistry. In *Intl. Conf. on Machine Learning (ICML)*.

Glasserman, P. (2004). *Monte Carlo methods in financial engineering*, volume 53. Springer.

Goldfarb, D. and Idnani, A. (1983). A numerically stable dual method for solving strictly convex quadratic programs. *Mathematical programming*, 27(1):1–33.

Goodfellow, I., Bengio, Y., and Courville, A. (2016). *Deep learning*. MIT press.

Green, M. and Moore, J. B. (1986). Persistence of excitation in linear systems. *Systems and Control Letters*, 7(5):351–360.

Gregor, K., Danihelka, I., Graves, A., Rezende, D., and Wierstra, D. (2015). Draw: A recurrent neural network for image generation. In *Intl. Conf. on Machine Learning (ICML)*.

Grill, J.-B., Strub, F., Altché, F., Tallec, C., Richemond, P., Buchatskaya, E., Doersch, C., Avila Pires, B., Guo, Z., Gheshlaghi Azar, M., et al. (2020). Bootstrap your own latent-a new approach to self-supervised learning. In *Advances in Neural Information Processing Systems (NeurIPS)*.

Ha, D. and Schmidhuber, J. (2018). Recurrent world models facilitate policy evolution. In *Advances in Neural Information Processing Systems Deep Learning Symposium (NeurIPS)*.

Haarnoja, T., Ajay, A., Levine, S., and Abbeel, P. (2016). Backprop KF: Learning discriminative deterministic state estimators. In *Advances in Neural Information Processing Systems Deep Learning Symposium (NeurIPS)*.

Hafner, D., Lillicrap, T., Fischer, I., Villegas, R., Ha, D., Lee, H., and Davidson, J. (2019). Learning latent dynamics for planning from pixels. In *Intl. Conf. on Machine Learning (ICML)*.

Hafner, D., Lillicrap, T. P., Norouzi, M., and Ba, J. (2021). Mastering atari with discrete world models. In *Intl. Conf. on Learning Representations (ICLR)*.

Hamilton, W., Ying, Z., and Leskovec, J. (2017). Inductive representation learning on large graphs. *Advances in Neural Information Processing Systems (NeurIPS)*, 30.

Harada, K., Tsuji, T., Nagata, K., Yamanobe, N., and Onda, H. (2014). Validating an object placement planner for robotic pick-and-place tasks. *Robotics and Autonomous Systems*, 62(10):1463–1477.

Hartenberg, R. S. and Denavit, J. (1955). A kinematic notation for lower pair mechanisms based on matrices. *J. Appl. Mech.*, 77(2):215–221.

Hastings, W. K. (1970). Monte carlo sampling methods using markov chains and their appucations. *Biometrika*, 57(1):97–109.

Haustein, J. A., Cruciani, S., Asif, R., Hang, K., and Kragic, D. (2019a). Placing objects with prior in-hand manipulation using dexterous manipulation graphs. In *IEEE-RAS Intl. Conf. Humanoid Robots (Humanoids)*, pages 453–460.

Haustein, J. A., Hang, K., Stork, J., and Kragic, D. (2019b). Object placement planning and optimization for robot manipulators. In *IEEE/RSJ Intl. Conf. Intelligent Robots and Systems (IROS)*, pages 7417–7424.

He, K., Fan, H., Wu, Y., Xie, S., and Girshick, R. (2020). Momentum contrast for unsupervised visual representation learning. In *IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*.

He, K., Zhang, X., Ren, S., and Sun, J. (2016). Deep residual learning for image recognition. In *IEEE Conf. Computer Vision and Pattern Recognition (CVPR)*, pages 770–778.

Hinton, G. E. (2002). Training products of experts by minimizing contrastive divergence. *Neural Computation*, 14(8):1771–1800.

Ho, C.-K. and King, C.-T. (2022). Selective inverse kinematics: A novel approach to finding multiple solutions fast for high-dof robotic. *arXiv preprint arXiv:2202.07869*.

Ho, J. and Ermon, S. (2016). Generative adversarial imitation learning. In *Advances in Neural Information Processing Systems (NeurIPS)*.

Ho, J., Jain, A., and Abbeel, P. (2020). Denoising diffusion probabilistic models. In *Advances in Neural Information Processing Systems (NeurIPS)*.

Hochreiter, S. and Schmidhuber, J. (1997). Long short-term memory. *Neural computation*, 9(8):1735–1780.

Hoffman, M. D., Blei, D. M., Wang, C., and Paisley, J. (2013). Stochastic variational inference. *Journal of Machine Learning Research*.

Hogan, F. R., Tremblay, J.-F., Baghi, B. H., Jenkin, M., Siddiqi, K., and Dudek, G. (2022). Finger-sts: Combined proximity and tactile sensing for robotic manipulation. *IEEE Robotics and Automation Letters (RAL)*, 7(4):10865–10872.

Hoque, R., Seita, D., Balakrishna, A., Ganapathi, A., Tanwani, A., Jamali, N., Yamane, K., Iba, S., and Goldberg, K. (2022). Visuospatial foresight for physical sequential fabric manipulation. *Autonomous Robots*, 46.

Hornik, K., Stinchcombe, M., and White, H. (1989). Multilayer feedforward networks are universal approximators. *Neural networks*, 2(5):359–366.

Husty, M., Pfurner, M., and Schröcker, H. (2007). A new and efficient algorithm for the inverse kinematics of a general serial 6r manipulator. *Mech. Mach. Theory*, 42(1):66–81.

Ichnowski, J., Avigal, Y., Satish, V., and Goldberg, K. (2020). Deep learning can accelerate grasp-optimized motion planning. *Science Robotics*, 5(48):eabd7710.

Ichter, B., Harrison, J., and Pavone, M. (2018). Learning sampling distributions for robot motion planning. In *IEEE Intl. Conf. on Robotics and Automation (ICRA)*.

Ichter, B. and Pavone, M. (2019). Robot motion planning in learned latent spaces. *IEEE Robotics and Automation Letters (RAL)*, 4(3):2407–2414.

Jang, E., Irpan, A., Khansari, M., Kappler, D., Ebert, F., Lynch, C., Levine, S., and Finn, C. (2022). Bc-z: Zero-shot task generalization with robotic imitation learning. In *Conf. Robot Learning (CORL)*, pages 991–1002.

Johns, E. (2021). Coarse-to-fine imitation learning: Robot manipulation from a single demonstration. In *IEEE Intl. Conf. Robotics and Automation (ICRA)*, pages 4613–4619.

Johnson, M. J., Duvenaud, D., Wiltschko, A. B., Adams, R. P., and Datta, S. R. (2016). Composing graphical models with neural networks for structured representations and fast inference. In *Advances in Neural Information Processing Systems Deep Learning Symposium (NeurIPS)*.

Jordan, M. I., Ghahramani, Z., Jaakkola, T. S., and Saul, L. K. (1999a). An introduction to variational methods for graphical models. *Machine learning*, 37:183–233.

Jordan, M. I., Ghahramani, Z., Jaakkola, T. S., and Saul, L. K. (1999b). *An introduction to variational methods for graphical models*, pages 105–161. MIT Press, Cambridge, MA, USA.

Jordan, M. I. and Rumelhart, D. E. (1992). Forward models: Supervised learning with a distal teacher. *Cognitive science*, 16(3):307–354.

Kalashnikov, D., Irpan, A., Pastor, P., Ibarz, J., Herzog, A., Jang, E., Quillen, D., Holly, E., Kalakrishnan, M., Vanhoucke, V., et al. (2018). QT-opt: Scalable deep reinforcement learning for vision-based robotic manipulation. In *Conf. Robot Learning (CORL)*, pages 651–673.

Kalashnkov, D., Varley, J., Chebotar, Y., Swanson, B., Jonschkowski, R., Finn, C., Levine, S., and Hausman, K. (2021). Mt-opt: Continuous multi-task robotic reinforcement learning at scale. *arXiv preprint arXiv:2104.08212*.

Kalman, R. E. (1960). A new approach to linear filtering and prediction problems. *Transactions of the ASME–Journal of Basic Engineering*, 82(Series D):35–45.

Karl, M., Sölch, M., Bayer, J., and van der Smagt, P. (2017). Deep variational Bayes filters: Unsupervised learning of state space models from raw data. In *Intl. Conf. on Learning Representations (ICLR)*, Toulon, France.

Kenton, J. D. M.-W. C. and Toutanova, L. K. (2019). Bert: Pre-training of deep bidirectional transformers for language understanding. In *North American Chapter of the Association for Computational Linguistics (NAACL-HLT)*.

Kenwright, B. (2012). Inverse kinematics–cyclic coordinate descent (CCD). *J. Graphics Tools*, 16(4):177–217.

Khan, A., Ribeiro, A., Kumar, V., and Francis, A. G. (2020). Graph neural networks for motion planning. *arXiv preprint arXiv:2006.06248*.

Kim, M.-J., Song, T.-H., Jin, S.-H., Jung, S.-M., Go, G.-H., Kwon, K.-H., and Jeon, J.-W. (2010). Automatically available photographer robot for controlling composition and taking pictures. In *IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS)*, pages 6010–6015.

Kingma, D. P. and Ba, J. (2015). Adam: A method for stochastic optimization. In *Intl. Conf. on Learning Representations (ICLR)*.

Kingma, D. P. and Welling, M. (2014). Auto-encoding variational Bayes. In *Intl. Conf. on Learning Representations (ICLR)*.

Kipf, T. N. and Welling, M. (2016). Variational graph auto-encoders. *NeurIPS Bayesian Deep Learning Workshop*.

Kipf, T. N. and Welling, M. (2017). Semi-Supervised Classification with Graph Convolutional Networks. In *Intl. Conf. on Learning Representations (ICLR)*.

Kirillov, A., Mintun, E., Ravi, N., Mao, H., Rolland, C., Gustafson, L., Xiao, T., Whitehead, S., Berg, A. C., Lo, W.-Y., Dollár, P., and Girshick, R. (2023). Segment anything. *arXiv preprint arXiv:2304.02643*.

Kloss, A., Schaal, S., and Bohg, J. (2020). Combining learned and analytical models for predicting action effects from sensory data. *Intl. Journal of Robotics Research*.

Kornblith, S., Shlens, J., and Le, Q. V. (2019). Do better imagenet models transfer better? In *IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*.

Kostrikov, I., Agrawal, K. K., Dwibedi, D., Levine, S., and Tompson, J. (2019). Discriminator-actor-critic: Addressing sample inefficiency and reward bias in adversarial imitation learning. In *Intl. Conf. Learning Representations (ICLR)*.

Krishnan, R. G., Shalit, U., and Sontag, D. (2017). Structured inference networks for nonlinear state space models. In *Association for the Advancement of Artificial Intelligence Conf. (AAAI)*.

Kruse, J., Ardizzone, L., Rother, C., and Köthe, U. (2021). Benchmarking invertible architectures on inverse problems. *ICML Workshop on Invertible Neural Networks, Normalizing Flows, and Explicit Likelihood Models*.

Lakshminarayanan, B., Pritzel, A., and Blundell, C. (2017). Simple and scalable predictive uncertainty estimation using deep ensembles. In *Advances in Neural Information Processing Systems Deep Learning Symposium (NeurIPS)*.

Laskey, M., Lee, J., Fox, R., Dragan, A., and Goldberg, K. (2017). Dart: Noise injection for robust imitation learning. In *Conf. Robot Learning (CORL)*, pages 143–156.

Le Naour, T., Courty, N., and Gibet, S. (2019). Kinematics in the metric space. *Comput. Graph.*, 84:13–23.

LeCun, Y., Boser, B., Denker, J. S., Henderson, D., Howard, R. E., Hubbard, W., and Jackel, L. D. (1989). Backpropagation applied to handwritten zip code recognition. *Neural Computation*, 1(4):541–551.

Lee, H.-Y. and Liang, C.-G. (1988). A new vector theory for the analysis of spatial mechanisms. *Mech. Mach. Theory*, 23(3):209–217.

Lee, M., Yi, B., Martin-Martin, R., Savarese, S., and Bohg, J. (2020). Multimodal sensor fusion with differentiable filters. In *IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS)*.

Lembono, T. S., Pignat, E., Jankowski, J., and Calinon, S. (2021). Learning constrained distributions of robot configurations with generative adversarial network. *IEEE Robotics and Automation Letters*, 6(2):4233–4240.

Lepetit, V., Moreno-Noguer, F., and Fua, P. (2009). Epnp: An accurate o(n) solution to the pnp problem. *Intl. journal of Computer Vision*, 81:155–166.

Levine, S., Finn, C., Darrell, T., and Abbeel, P. (2016). End-to-end training of deep visuomotor policies. *J. Machine Learning Research*, 17(1):1334–1373.

Li, J., Sousa, M., Mahadevan, K., Wang, B., Aoyagui, P. A., Yu, N., Yang, A., Balakrishnan, R., Tang, A., and Grossman, T. (2023a). Stargazer: An interactive camera robot for capturing how-to videos based on subtle instructor cues. In *ACM CHI Conf. on Human Factors in Computing Systems (CHI)*.

Li, X., Baum, M., and Brock, O. (2023b). Augmentation enables one-shot generalization in learning from demonstration for contact-rich manipulation. In *IEEE/RSJ Intl. Conf. Intelligent Robots and Systems (IROS)*, pages 3656–3663.

Li, Y., Vinyals, O., Dyer, C., Pascanu, R., and Battaglia, P. (2018). Learning deep generative models of graphs. *arXiv preprint arXiv:1803.03324*.

Liao, R. (2022). Graph neural networks: Graph generation. In Wu, L., Cui, P., Pei, J., and Zhao, L., editors, *Graph Neural Networks: Foundations, Frontiers, and Applications*, pages 225–250. Springer Singapore, Singapore.

Liberti, L., Lavor, C., Maculan, N., and Mucherino, A. (2014). Euclidean Distance Geometry and Applications. *SIAM Rev.*, 56(1):3–69.

Limoyo, O., Ablett, T., and Kelly, J. (2023a). Learning sequential latent variable models from multimodal time series data. In *Intelligent Autonomous Systems 17*, volume 577 of *Lecture Notes in Networks and Systems*, pages 511–528. Best Paper Finalist.

Limoyo, O., Ablett, T., and Kelly, J. (2023b). Learning sequential latent variable models from multimodal time series data. In *Intelligent Autonomous Systems 17*, pages 511–528.

Limoyo, O., Ablett, T., Maric, F., Volpatti, L., and Kelly, J. (2018). Self-calibration of mobile manipulator kinematic and sensor extrinsic parameters through contact-based interaction. In *IEEE Intl. Conf. on Robotics and Automation (ICRA)*.

Limoyo, O., Chan, B., Marić, F., Wagstaff, B., Mahmood, A. R., and Kelly, J. (2020a). Heteroscedastic uncertainty for robust generative latent dynamics. *IEEE Robotics and Automation Letters (RAL)*, 5(4):6654–6661.

Limoyo, O., Chan, B., Maric, F., Wagstaff, B., Mahmood, R., and Kelly, J. (2020b). Heteroscedastic uncertainty for robust generative latent dynamics. *IEEE Robotics and Automation Letters*, 5(4):6654–6661.

Limoyo, O., Konar, A., Ablett, T., Kelly, J., Hogan, F. R., and Dudek, G. (2024a). Working backwards: Learning to place by picking. In *IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS)*, Abu Dhabi, United Arab Emirates. Submitted.

Limoyo, O., Li, J., Rivkhin, D., Kelly, J., and Dudek, G. (2024b). Reference-guided robotic photography through natural language interaction. In *Proceedings of the ACM/IEEE International Conference on Human Robot Interaction (HRI) Workshop on Human—Large Language Model Interaction: The Dawn of a New Era or the End of it All?*, Boulder, Colorado, USA.

Limoyo, O., Li, J., Rivkin, D., Kelly, J., and Dudek, G. (2024c). Photobot: Reference-guided interactive photography via natural language. In *IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS)*, Abu Dhabi, United Arab Emirates. Submitted.

Limoyo, O., Maric, F., Giamou, M., Alexson, P., Petrovic, I., and Kelly, J. (2023c). Euclidean equivariant models for generative graphical inverse kinematics. In *Proceedings of the Robotics: Science and Systems (RSS) Workshop on Symmetries in Robot Learning*, Daegu, Republic of Korea.

Limoyo, O., Maric, F., Giamou, M., Alexson, P., Petrovic, I., and Kelly, J. (2023d). Generative graphical inverse kinematics. *IEEE Transactions on Robotics*. To Appear.

Lippi, M., Poklukar, P., Welle, M. C., Varava, A., Yin, H., Marino, A., and Kragic, D. (2020). Latent space roadmap for visual action planning of deformable and rigid object manipulation. In *IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS)*.

Liu, S., Zeng, Z., Ren, T., Li, F., Zhang, H., Yang, J., Li, C., Yang, J., Su, H., Zhu, J., et al. (2023). Grounding dino: Marrying dino with grounded pre-training for open-set object detection. *arXiv preprint arXiv:2303.05499*.

Ljung, L. (1998). System identification. In Procházka, A., Uhlíř, J., Rayner, P. W. J., and Kingsbury, N. G., editors, *Signal Analysis and Prediction*, pages 163–173. Birkhäuser Boston, Boston, MA.

Loshchilov, I. and Hutter, F. (2019). Decoupled weight decay regularization. In *Intl. Conf. on Learning Representations (ICLR)*.

Lowe, D. G. (2004). Distinctive image features from scale-invariant keypoints. *Intl. Journal of Computer Vision*, 60:91–110.

Lynch, C., Wahid, A., Tompson, J., Ding, T., Betker, J., Baruch, R., Armstrong, T., and Florence, P. (2023). Interactive language: Talking to robots in real time. *IEEE Robotics and Automation Letters (RAL)*, pages 1–8.

Lynch, K. M. and Park, F. C. (2017). *Modern robotics*. Cambridge University Press.

MacKay, D. J. (1995). Probable networks and plausible predictions - a review of practical Bayesian methods for supervised neural networks. *Network: Computation in Neural Systems*, 6(3):469–505.

Mahmood, A. R., Korenkevych, D., Vasan, G., Ma, W., and Bergstra, J. (2018). Benchmarking reinforcement learning algorithms on real-world robots. In *Conf. on Robot Learning (CORL)*.

Mandlekar, A., Xu, D., Wong, J., Nasiriany, S., Wang, C., Kulkarni, R., Fei-Fei, L., Savarese, S., Zhu, Y., and Mart'in-Mart'in, R. (2021). What matters in learning from offline human demonstrations for robot manipulation. In *Conf. Robot Learning (CORL)*, pages 1678–1690.

Manocha, D. and Canny, J. F. (1994). Efficient inverse kinematics for general 6r manipulators. *IEEE Trans. Robot.*, 10(5):648–657.

Maric, F., Giamou, M., Hall, A. W., Khoubyarian, S., Petrovic, I., and Kelly, J. (2021). Riemannian optimization for distance-geometric inverse kinematics. *IEEE Transactions on Robotics*.

Mitchell, T. M. (2008). The need for biases in learning generalizations. Technical Report CBM-TR-117, Rutgers University.

Mnih, V., Kavukcuoglu, K., Silver, D., Graves, A., Antonoglou, I., Wierstra, D., and Riedmiller, M. (2013). Playing Atari with deep reinforcement learning. In *NeurIPS'13 Deep Learning Workshop*.

Moisan, L. and Stival, B. (2004). A probabilistic criterion to detect rigid point matches between two images and estimate the fundamental matrix. *Intl. Journal of Computer Vision*, 57:201–218.

Murphy, K. P. (2012a). *Machine Learning: A Probabilistic Perspective*. The MIT Press.

Murphy, K. P. (2012b). *Machine Learning: A Probabilistic Perspective*. The MIT Press.

Murray, N., Marchesotti, L., and Perronnin, F. (2012). Ava: A large-scale database for aesthetic visual analysis. In *IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*.

Murray, R. M., Li, Z., and Sastry, S. S. (2017). *A mathematical introduction to robotic manipulation*. CRC press.

Nair, A. V., Pong, V., Dalal, M., Bahl, S., Lin, S., and Levine, S. (2018). Visual reinforcement learning with imagined goals. In *Advances in Neural Information Processing Systems (NeurIPS)*.

Nair, S., Babaeizadeh, M., Finn, C., Levine, S., and Kumar, V. (2020). Trass: Time reversal as self-supervision. In *IEEE Intl. Conf. Robotics and Automation (ICRA)*, pages 115–121.

Nakamura, Y., Hanafusa, H., and Yoshikawa, T. (1987). Task-priority based redundancy control of robot manipulators. *Int. J. Rob. Res.*, 6(2):3–15.

Newbury, R., Cosgun, A., Koseoglu, M., and Drummond, T. (2020). Learning to take good pictures of people with a robot photographer. In *IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS)*, pages 11268–11275.

Ngiam, J., Khosla, A., Kim, M., Nam, J., Lee, H., and Ng, A. Y. (2011). Multimodal deep learning. In *Intl. Conf. on Machine Learning (ICML)*.

Nocedal, J. and Wright, S. J. (1999). *Numerical optimization*. Springer.

OpenAI (2023). Gpt-4 technical report. Technical report.

Opper, M. and Saad, D. (2001). *Advanced Mean Field Methods: Theory and Practice*. The MIT Press.

Oquab, M., Darcet, T., Moutakanni, T., Vo, H. V., Szafraniec, M., Khalidov, V., Fernandez, P., HAZIZA, D., Massa, F., El-Nouby, A., et al. (2023). Dinov2: Learning robust visual features without supervision. *Transactions on Machine Learning Research*.

Oron, S., Dekel, T., Xue, T., Freeman, W. T., and Avidan, S. (2017). Best-buddies similarity—robust template matching using mutual nearest neighbors. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 40(8):1799–1813.

Orsini, M., Raichuk, A., Hussenot, L., Vincent, D., Dadashi, R., Girgin, S., Geist, M., Bachem, O., Pietquin, O., and Andrychowicz, M. (2021). What matters for adversarial imitation learning? In *Advances in Neural Information Processing Systems Deep Learning Symposium (NeurIPS)*, pages 14656–14668.

Pathak, D., Agrawal, P., Efros, A. A., and Darrell, T. (2017). Curiosity-driven exploration by self-supervised prediction. In *Intl. Conf. on Machine Learning (ICML)*.

Pomerleau, D. (1989). Alvinn: An autonomous land vehicle in a neural network. In *Advances in Neural Information Processing Systems Deep Learning Symposium (NeurIPS)*, pages 305–313.

Pomerleau, D. A. (1993). Input reconstruction reliability estimation. In *Advances in Neural Information Processing Systems Deep Learning Symposium (NeurIPS)*.

Porta, J., Ros, L., Thomas, F., and Torras, C. (2005). A branch-and-prune solver for distance constraints. *IEEE Trans. Robot.*, 21:176–187.

Qureshi, A. H., Miao, Y., Simeonov, A., and Yip, M. C. (2020). Motion planning networks: Bridging the gap between learning-based and classical motion planners. *IEEE Transactions on Robotics*, 37(1):48–66.

Rafailov, R., Yu, T., Rajeswaran, A., and Finn, C. (2021). Offline reinforcement learning from images with latent space models. In *Learning for Dynamics and Control (L4DC)*.

Rauch, H. E., Tung, F., and Striebel, C. T. (1965). Maximum likelihood estimates of linear dynamic systems. *AIAA Journal*, 3(8):1445–1450.

Reimers, N. and Gurevych, I. (2019). Sentence-bert: Sentence embeddings using siamese bert-networks. In *Conf. on Empirical Methods in Natural Language Processing (EMNLP)*.

Ren, H. and Ben-Tzvi, P. (2020). Learning inverse kinematics and dynamics of a robotic manipulator using generative adversarial networks. *Robotics and Autonomous Systems*, 124:103386.

Rezaei-Shoshtari, S., Hogan, F. R., Jenkin, M., Meger, D., and Dudek, G. (2021). Learning intuitive physics with multimodal generative models. In *Association for the Advancement of Artificial Intelligence Conf. (AAAI)*.

Rezende, D. J., Mohamed, S., and Wierstra, D. (2014a). Stochastic backpropagation and approximate inference in deep generative models. In *Intl. Conf. on Machine Learning (ICML)*.

Rezende, D. J., Mohamed, S., and Wierstra, D. (2014b). Stochastic backpropagation and approximate inference in deep generative models. In *Intl. Conf. on Machine Learning (ICML)*.

Richter, C. and Roy, N. (2017). Safe visual navigation via deep learning and novelty detection. In *Robotics: Science and Systems (RSS)*.

Rivkin, D., Dudek, G., Kakodkar, N., Meger, D., Limoyo, O., Jenkin, M., Liu, X., and Hogan, F. (2023). Ansel photobot: A robot event photographer with semantic intelligence. In *IEEE Intl. Conf. on Robotics and Automation (ICRA)*.

Robbins, H. and Monro, S. (1951). A stochastic approximation method. *The annals of mathematical statistics*, pages 400–407.

Roweis, S. and Ghahramani, Z. (1999). A unifying review of linear gaussian models. *Neural computation*, 11(2):305–345.

Salimans, T. and Kingma, D. P. (2016). Weight normalization: A simple reparameterization to accelerate training of deep neural networks. In *Advances in Neural Information Processing Systems Deep Learning Symposium (NeurIPS)*.

Satorras, V. G., Hoogeboom, E., and Welling, M. (2021). E (n) equivariant graph neural networks. In *Intl. Conf. on Machine Learning (ICML)*.

Schubert, E., Sander, J., Ester, M., Kriegel, H. P., and Xu, X. (2017). Dbscan revisited, revisited: why and how you should (still) use dbscan. *ACM Transactions on Database Systems (TODS)*, 42(3):1–21.

Sciavicco, L. and Siciliano, B. (1986). Coordinate transformation: A solution algorithm for one class of robots. *IEEE Transactions on Systems, Man and Cybernetics*, 16:550 – 559.

Sermanet, P., Lynch, C., Chebotar, Y., Hsu, J., Jang, E., Schaal, S., Levine, S., and Brain, G. (2018). Time-contrastive networks: Self-supervised learning from video. In *IEEE Intl. Conf. on Robotics and Automation (ICRA)*.

Siciliano, B., Sciavicco, L., Villani, L., and Oriolo, G. (2010). *Robotics: Modelling, Planning and Control*. Springer Science & Business Media.

Simm, G. N. C. and Hernández-Lobato, J. M. (2020). A generative model for molecular distance geometry. In *Intl. Conf. on Machine Learning (ICML)*.

Sohl-Dickstein, J., Weiss, E., Maheswaranathan, N., and Ganguli, S. (2015). Deep unsupervised learning using nonequilibrium thermodynamics. In *Intl. Conf. on Machine Learning (ICML)*.

Sohn, K., Lee, H., and Yan, X. (2015a). Learning structured output representation using deep conditional generative models. In *Advances in Neural Information Processing Systems (NeurIPS)*.

Sohn, K., Yan, X., and Lee, H. (2015b). Learning structured output representation using deep conditional generative models. In *Advances in Neural Information Processing Systems - Volume 2 (NeurIPS*.

Song, Y. and Ermon, S. (2019). Generative modeling by estimating gradients of the data distribution. In *Advances in Neural Information Processing Systems (NeurIPS)*.

Spector, O. and Di Castro, D. (2021). Insertionnet-a scalable solution for insertion. *IEEE Robotics and Automation Letters (RAL)*, 6(3):5509–5516.

Spector, O., Tchuiev, V., and Di Castro, D. (2022). Insertionnet 2.0: Minimal contact multi-step insertion using multimodal multiview sensory input. In *IEEE Intl. Conf. Robotics and Automation (ICRA)*, pages 6330–6336.

Srivastava, N., Mansimov, E., and Salakhudinov, R. (2015). Unsupervised learning of video representations using lstms. In *Intl. Conf. on Machine Learning (ICML)*.

Srivastava, N. and Salakhutdinov, R. (2014). Multimodal learning with deep boltzmann machines. *Journal of Machine Learning Research*, 15(1):2949–2980.

Stewart, C. V. (1995). Minpran: A new robust estimator for computer vision. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 17(10):925–938.

Sundermeyer, M., Mousavian, A., Triebel, R., and Fox, D. (2021). Contact-graspnet: Efficient 6-dof grasp generation in cluttered scenes. In *IEEE Intl. Conf. Robotics and Automation (ICRA)*, pages 13438–13444.

Sutton, R. S., Szepesvári, C., Geramifard, A., and Bowling, M. (2008). Dyna-style planning with linear function approximation and prioritized sweeping. In *Conf. on Uncertainty in Artificial Intelligence (UAI)*.

Suzuki, M., Nakayama, K., and Matsuo, Y. (2016). Joint multimodal learning with deep generative models. *arXiv preprint arXiv:1611.01891*.

Talmor, A., Yoran, O., Bras, R. L., Bhagavatula, C., Goldberg, Y., Choi, Y., and Berant, J. (2021). CommonsenseQA 2.0: Exposing the limits of AI through gamification. In *Advances in Neural Information Processing Systems Datasets and Benchmarks Track (Round 1)*.

Tedrake, R. (2022). *Robotic Manipulation*.

Tipping, M. E. and Bishop, C. M. (1999). Probabilistic principal component analysis. *Journal of the Royal Statistical Society Series B: Statistical Methodology*, 61(3):611–622.

Torr, P. H. and Zisserman, A. (2000). Mlesac: A new robust estimator with application to estimating image geometry. *Computer Vision and Image Understanding*, 78(1):138–156.

Urain, J., Funk, N., Peters, J., and Chalvatzaki, G. (2023). Se(3)-diffusionfields: Learning smooth cost functions for joint grasp and motion optimization through diffusion. In *IEEE Intl. Conf. on Robotics and Automation (ICRA)*.

Urtasun, R., Fleet, D. J., and Fua, P. (2006). 3D people tracking with Gaussian process dynamical models. In *IEEE Conf. Computer Vision and Pattern Recognition (CVPR)*.

Valassakis, E., Di Palo, N., and Johns, E. (2021). Coarse-to-fine for sim-to-real: Sub-millimetre precision across wide task spaces. In *IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS)*.

Vedantam, R., Fischer, I., Huang, J., and Murphy, K. (2018). Generative models of visually grounded imagination. In *Intl. Conf. on Learning Representations (ICLR)*.

Velickovic, P., Cucurull, G., Casanova, A., Romero, A., Lio, P., and Bengio, Y. (2018). Graph attention networks. In *Intl. Conf. on Learning Representations (ICLR)*.

Villegas, R., Yang, J., Ceylan, D., and Lee, H. (2018). Neural kinematic networks for unsupervised motion retargetting. In *IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*.

von Oehsen, T., Fabisch, A., Kumar, S., and Kirchner, F. (2020). Comparison of Distal Teacher Learning with Numerical and Analytical Methods to Solve Inverse Kinematics for Rigid-Body Mechanisms. *arXiv:2003.00225 [cs]*.

Wächter, A. and Biegler, L. T. (2006). On the implementation of an interior-point filter line-search algorithm for large-scale nonlinear programming. *Mathematical programming*, 106:25–57.

Wahlström, N., Schön, T. B., and Desienroth, M. P. (2015). From pixels to torques: Policy learning with deep dynamical models. In *ICML'15 Deep Learning Workshop*.

Wang, Z., Novikov, A., Zolna, K., Merel, J. S., Springenberg, J. T., Reed, S. E., Shahriari, B., Siegel, N., Gulcehre, C., Heess, N., et al. (2020). Critic regularized regression. In *Advances in Neural Information Processing Systems Deep Learning Symposium (NeurIPS)*, pages 7768–7778.

Watter, M., Springenberg, J. T., Boedecker, J., and Riedmiller, M. (2015). Embed to control: a locally linear latent dynamics model for control from raw images. In *Advances in Neural Information Processing Systems Deep Learning Symposium (NeurIPS)*.

Whitney, D. E. (1969). Resolved motion rate control of manipulators and human prostheses. *IEEE Transactions on man-machine systems*, 10(2):47–53.

Williams, R. J. (1992). Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine learning*, 8:229–256.

Wu, M. and Goodman, N. (2018). Multimodal generative models for scalable weakly-supervised learning. In *Advances in Neural Information Processing Systems Deep Learning Symposium (NeurIPS)*.

Yenamandra, T., Bernard, F., Wang, J., Mueller, F., and Theobalt, C. (2019). Convex Optimisation for Inverse Kinematics. *Intl. Conf. on 3D Vision (3DV)*, pages 318–327.

Yu, K., Bauzá, M., Fazeli, N., and Rodriguez, A. (2016). More than a million ways to be pushed. A high-fidelity experimental dataset of planar pushing. In *IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS)*.

Zabarauskas, M. and Cameron, S. (2014). Luke: An autonomous robot photographer. In *IEEE Intl. Conf. on Robotics and Automation (ICRA)*, pages 1809–1815.

Zakka, K., Zeng, A., Lee, J., and Song, S. (2020). Form2fit: Learning shape priors for generalizable assembly from disassembly. In *IEEE Intl. Conf. Robotics and Automation (ICRA)*, pages 9404–9410.

Zeng, A., Song, S., Yu, K.-T., Donlon, E., Hogan, F. R., Bauza, M., Ma, D., Taylor, O., Liu, M., Romo, E., et al. (2022). Robotic pick-and-place of novel objects in clutter with multi-affordance grasping and cross-domain image matching. *Intl. J. Robotics Research*, 41(7):690–705.

Zhang, A., Lipton, Z. C., Li, M., and Smola, A. J. (2023). *Dive into Deep Learning*. Cambridge University Press.

Zhang, M., Vikram, S., Smith, L., Abbeel, P., Johnson, M., and Levine, S. (2019a). Solar: Deep structured representations for model-based reinforcement learning. In *Intl. Conf. on Machine Learning (ICML)*.

Zhang, M., Vikram, S., Smith, L., Abbeel, P., Johnson, M., and Levine, S. (2019b). Solar: Deep structured representations for model-based reinforcement learning. In *Intl. Conf. on Machine Learning (ICML)*.

Zhao, T. Z., Kumar, V., Levine, S., and Finn, C. (2023). Learning fine-grained bimanual manipulation with low-cost hardware. In *Robotics: Science and Systems (RSS)*.

Zhou, J., Wei, C., Wang, H., Shen, W., Xie, C., Yuille, A., and Kong, T. (2021a). Image bert pre-training with online tokenizer. In *Intl. Conf. on Learning Representations (ICLR)*.

Zhou, K., Chen, C., Wang, B., Saputra, M. R. U., Trigoni, N., and Markham, A. (2021b). VMLoc: Variational fusion for learning-based multimodal camera localization. In *Association for the Advancement of Artificial Intelligence Conf. (AAAI)*.

Zhou, X., Girdhar, R., Joulin, A., Krähenbühl, P., and Misra, I. (2022). Detecting twenty-thousand classes using image-level supervision. In *European Conf. on Computer Vision (ECCV)*.