THESIS FOR THE DEGREE OF LICENTIATE OF ENGINEERING

Data-Efficient Representation Learning for Grasping and Manipulation

Ahmet Ercan Tekden

Department of Electrical Engineering
Chalmers University of Technology
Gothenburg, Sweden, 2024

Data-Efficient Representation Learning for Grasping and Manipulation

Ahmet Ercan Tekden

This thesis has been prepared using LaTeX.

Department of Electrical Engineering
Chalmers University of Technology
SE-412 96 Gothenburg, Sweden
Phone: +46 (0)31 772 1000
www.chalmers.se

Printed by Chalmers Reproservice
Gothenburg, Sweden, January 2024

To my family and friends.

Abstract

General-purpose robotics require adaptability to environmental variations and, therefore, need effective representations for programming them. A common way to acquire such representations is through machine learning. Machine learning has shown great potential in computer vision, natural language processing, reinforcement learning, and robotics. However, this success relies on the availability of large datasets. It is hard to generate data for most robotic tasks; therefore, robotics requires data efficiency. This licentiate presents our initial progress on learning effective representations for robotics using a low amount of data. Specifically, we have used neural fields as our choice of generative models, and we have shown that we can learn local surface models for grasp synthesis and joint scene-motion models for motion generation. In the former work, we use local surface models as correspondences for grasp transfer. Unlike previous work, our method can transfer grasps demonstrated on a single object to novel objects, including ones from unseen categories, while acquiring higher spatial accuracy. The latter work shows we can model scenes and motions as smooth joint functions of shared embeddings. Unlike previous work, our approach requires less expert demonstration yet still generates precise motion trajectories.

Keywords: Grasping, Robot Manipulation, Robot Learning, Data-efficient Representation Learning, Learning from Demonstration, Neural Fields, Generative Modeling.

List of Publications

This thesis is based on the following publications:

[A] **Ahmet Ercan Tekden**, Marc Peter Deisenroth, Yasemin Bekiroglu, "Grasp Transfer based on Self-Aligning Implicit Representations of Local Surfaces". IEEE Robotics and Automation Letters (RA-L), vol. 8, no. 4, Oct. 2023.

[B] **Ahmet Ercan Tekden**, Marc Peter Deisenroth, Yasemin Bekiroglu, "Neural Field Movement Primitives for Joint Modelling of Scenes and Motions". IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), 2023, Detroit, USA.

Other publications by the author, not included in this thesis, are:

[C] **Ahmet Ercan Tekden**, Marc Peter Deisenroth, Yasemin Bekiroglu, "Affordance Transfer based on Self-Aligning Implicit Representations of Local Surfaces". *Workshop on implicit representations for robotic manipulation at Robotics: Science and Systems (RSS)*, 2022, New York, USA, (Hybrid).

[D] Yiting Chen, **Ahmet Ercan Tekden**, Marc Peter Deisenroth, Yasemin Bekiroglu, "Sliding Touch-based Exploration for Modeling Unknown Object Shape with Multi-finger Hands". *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2023, Detroit, USA.

[E] **Ahmet Ercan Tekden**, Marc Peter Deisenroth, Yasemin Bekiroglu, "Neural Field Movement Primitives for Joint Modelling of Scenes and Motions". *Learning meets model-based methods for manipulation and grasping workshop at IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2023, Detroit, USA.

[F] **Ahmet Ercan Tekden**, Aykut Erdem, Erkut Erdem, Tamim Asfour, Emre Ugur, "Object and relation centric representations for push effect prediction". *Robotics and Autonomous Systems*, 2024.

Acknowledgments

I would like to express my heartfelt gratitude to my supervisor, Yasemin Bekiroglu, for her unwavering support during my doctoral studies. Her valuable advice, encouragement, and motivation were instrumental in my academic success. I am also deeply grateful to Peter Marc Deisenroth for co-supervising my research and providing invaluable insights. I also would like to thank my examiner, Torsten Wik for his support throughout my PhD.

My sincerest gratitude to Albert Skegro, Attila Lischka, Remi Lacombe, Stefan Kojchev, and others for their friendship, joyful lunch breaks and fikas. Your friendship made the academic journey all the more enjoyable. I extend my appreciation to Muhammad Faris, Godwin Peprah, Gabriel Arslan Waltersson, Rita Laezza, Maximilian Diehl and Mattia de Lazzari for both academic and casual discussions and chats.

A heartfelt thank you to Alper Ahmetoglu, Alper Sarıalan, Safa Andac and Mete Tuluhan Akbulut for their enduring friendship and virtual connection. Your willingness to engage in both academic and casual conversations has been truly appreciated.

Lastly, my deepest gratitude goes to my parents and brothers for their unwavering support and encouragement throughout my entire life. Their influence has been pivotal in my educational journey, and I would not have been able to persevere without their love and guidance.

Acronyms

1-D:	One Dimensional
2-D:	Two Dimensional
3-D:	Three Dimensional
6-D:	Three Dimensional
LfD:	Learning from Demonstration
MLP:	Multilayer Perceptron

NeRF:	Neural Radiance Fields
ReLU:	Rectified Linear Unit
RGB:	Red Green Blue
SE(3):	Special Euclidean groups in three dimensions
SDF:	Signed Distance Function

Contents

Part I

Overview

CHAPTER 1

Introduction

Robots greatly benefit people and have widespread use in manufacturing, logistics, healthcare, agriculture, and many other areas. Robots can perform tasks more accurately, efficiently, and cheaply than humans. Furthermore, they do not need to rest. However, designing robot skills that can adapt to environmental variations is challenging. Because of this, general-purpose robotics is still an active research field. Currently, robots require setting up the environment specifically for their usage. This is because robot setups require programming robots with pre-defined movements, often by experienced software developers with expertise in robot programming. To operate in unstructured natural environments, such as our homes, robots need to understand the world around them and be able to adapt their actions to variations in the environment. This can be achieved by acquiring effective representations of the underlying environments and actions of the robot.

A promising approach to acquiring representations that allow the robot to generalize to different environments is through machine learning. Robots can learn from available data and through their own interactions with the environment. From the data, they can learn effective representations that allow the robot to generalize its actions according to the scene changes. For

learning effective representations, the choice of input data and how the input is processed is important. Depending on these design choices, the machine learning method may require a higher amount of data and/or show limited generalization capabilities.

In this thesis, we investigate how to learn effective representations for grasping and manipulation [1] that can generalize to various scene and sensor modalities while keeping the data requirement low.

1.1 Robot Learning

Robot learning refers to algorithms and techniques that allow a robot to learn from data, feedback, and environment interactions, often using machine learning. It deals with key robotics problems such as model learning [1], [2], imitation and apprenticeship learning [3], reinforcement learning [4], perception [5] and their various combinations to teach complex behavior to robots efficiently and robustly [6]. Robot learning has become an important topic because of the following reasons: (i) Many robotic skills are hard to program, (ii) robots have to deal with unknown and partial information, (iii) the world is dynamic, and environments keep changing between different trials [7].

Some common applications for robot learning are:

1. **Grasp Synthesis:** Grasp synthesis refers to the generation of grasp poses that allow robots to hold and safely pick up objects [8]. Robot learning provides a direct strategy for grasp synthesis where scene observations, often images or point-clouds, are mapped to grasp poses. These methods are particularly effective when object geometry is unknown or objects are densely placed in the scene [9].

2. **Robot Motion Generation:** Motion generation refers to generating trajectories or control vectors that enable robots to complete their tasks. With robot learning, this process is often performed through reinforcement learning or supervised learning [6]. Robot learning provides a flexible way to generate motions that adapt to variations in the environment.

[1] Manipulation refers to how a robot can interact with objects in its environment through grasping, moving, and rotating objects to achieve its goals.

3. **Learning Object Dynamics:** Learning object dynamics refers to learning machine learning models for predicting the effects of the robot's action on the environment. In robot learning, robots can use such models to plan the best set of actions [10].

Robot learning provides a flexible way for robots to perform their tasks. For this reason, robot learning is now widely used for programming robots.

1.2 Data-Efficiency in Robotics

The field of machine learning is growing rapidly. It has yielded impressive results in the fields of natural language processing, computer vision, and robotics. However, robotics overall lags behind in many robotic applications compared to the other major fields. The main reason for this is the lack of data since robotic data collection is challenging. Machine learning is relatively more successful for robotic applications, such as grasp synthesis, affordance learning, and other tasks where autonomous data collection is possible. However, for many robotic applications, data collection is difficult due to the following reasons:

- Robot hardware is expensive, so deploying robotics in the real world is costly. It requires constant monitoring. Yet, during data collection, robot hardware can still break down and require repair. Furthermore, it can be dangerous for robots to collect data in natural environments. In addition, exploratory policies require enforcing safety constraints, as robot actions can have irreversible consequences [11].

- Real-world environments are complex and unstructured. Therefore, capturing all possible scenarios and edge cases for different tasks is difficult.

- Data annotation requires manual labor. Furthermore, robotics often requires manual labor to be handled by experts. This makes data annotation at scale hard. Moreover, it takes a long time to collect data samples.

- Robotic data is noisy and not consistent between different setups. For example, even when two robotic labs share identical robotic setups, the data distributions collected in these labs can differ.

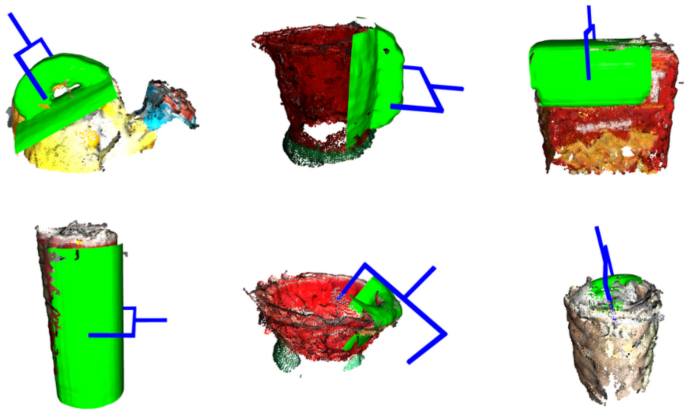

Figure 1.1: Local Surface Models for Grasp Transfer. Many objects share geometrically similar parts. We can utilize local surface models to identify such parts to transfer grasp experiences across object categories.

For these reasons, robotics requires data-efficient learning methods and smarter data generation methods. Prof. Kaelbling argues that to learn generalizable representation efficiently, inductive biases should be incorporated into learning algorithms in the form of prior knowledge and structure [12]. In this thesis, we employ two forms of inductive bias: Using local shape information, such as object parts, to learn compositional representations and using the spectral bias of neural fields to learn smooth representations.

Humans decompose environments into objects and parts and use them for physical reasoning [13], [14]. Compositional representations, *e.g.* describing object parts, allow higher transferability as they are often shared between different categories of objects; for example, the handle used for grasping is shared between many household objects. Consequently, compositionality can provide data efficiency to representation learning. We have employed this idea for modeling local surfaces to transfer grasp poses across object categories, as shown in Paper A. An illustration of our method is shown in Figure 1.1.

Spectral bias refers to the type of bias in the neural networks in which the network prioritizes learning less complex mappings between input and output features [15]. In Paper B, spectral bias has allowed us to model scenes and motions as smooth functions of shared embeddings using neural fields. The

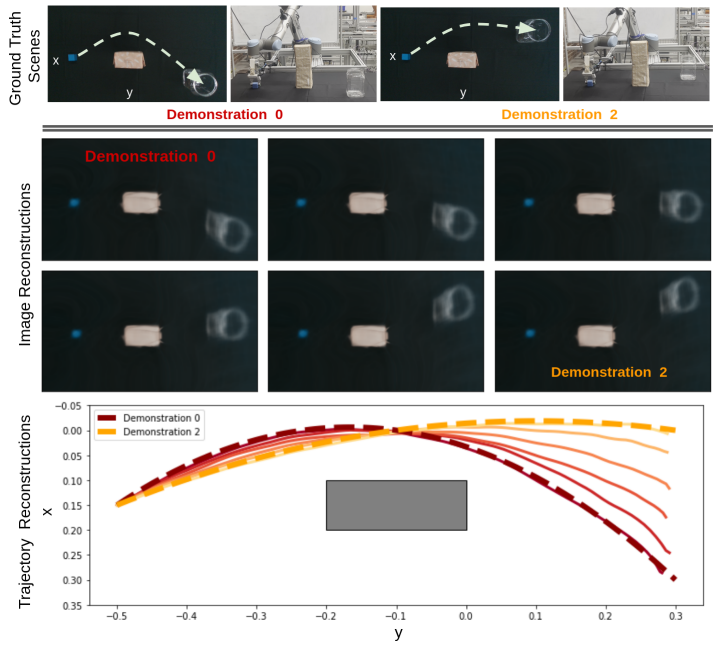

Figure 1.2: Joint Modeling of Scenes and Motions. We can model scenes and motions as smooth functions of shared embeddings. After training, we can reconstruct the images and trajectories corresponding to the unseen intermediate plastic container locations by interpolating between the learned embeddings of expert demonstrations shown on top.

acquired representations have allowed for generalizing to the scene variations in a data-efficient way and generating precise motion trajectories. This is illustrated in Figure 1.2.

1.3 Learning From Demonstration

Humans often do not learn tasks from scratch but use their prior information. Similarly, we can incorporate prior information to train robots in a more efficient way. For this, we can employ learning from demonstration (LfD). LfD involves utilizing instructions and/or demonstrations provided by humans to

ease the learning of new tasks [16], [17]. Some applications of LfD are as follows [18]:

- **Movement Primitives:** These are policy representations that encode continuous joint or state trajectories [19]–[23]. They are often used to teach robots complex motor skills efficiently using LfD data.

- **Behavioral Cloning:** Behavioral cloning (BC) refers to the supervised learning technique that learns the mapping between input states and actions using LfD data provided as state-action pairs [24], [25]. This allows the agent to mimic the expert demonstrator.

- **Correspondence Learning:** Learning from demonstration can be utilized to provide correspondences on the scene, which can be used as task waypoints to solve the tasks. Common examples of these correspondences are affordance heatmaps [26] and keypoints [27]–[29].

This project uses LfD for learning part-level shape correspondences (Paper A) and movement primitives (Paper B). In the former work, we provide a single grasp demonstration to learn part correspondences that can be used in grasping. For the latter work, we show that we can learn scene-trajectory mappings with neural fields using LfD data using a low number of demonstrations.

1.4 Generative Modeling

Generative modeling is a field within machine learning in which a model is trained to capture the underlying data distribution of a dataset to produce data samples similar to those observed in the dataset [30]. Ideally, the model will capture physical and semantic rules regarding the trained dataset and allow the generation of samples that follow these rules. For example, consider the handwritten number generation task, illustrated in Figure 1.3. We train this model only with images labeled 6. Our model can capture the underlying distribution and can generate novel images with label 6.

To understand the benefit of generative modeling for robotics, we will compare it with discriminative modeling. Discriminate models aim to find the decision boundaries between data classes in the training dataset using the

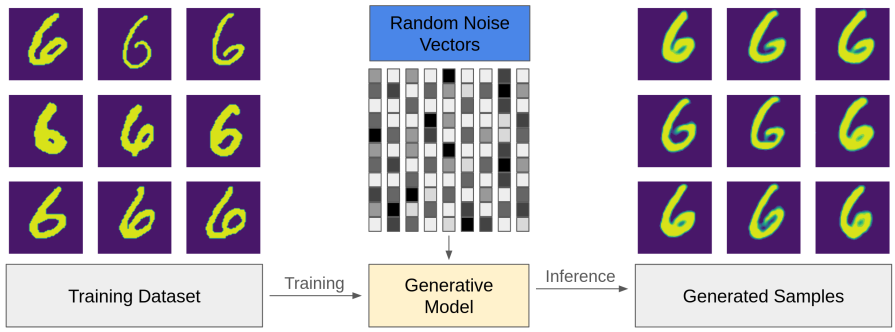

Figure 1.3: Illustration of a generative model for number generation. After training, we can generate novel images that look similar to training data by sampling random Gaussian noises.

provided class labels. A visualization of the comparison between these modeling types is shown in Figure 1.4.

To illustrate this comparison further, we compare these two approaches to modeling using classification as a benchmark. Discriminative modeling scales better than generative modeling; however, it will require more labeled data, including data samples from edge cases. Generative modeling, however, will reach a low error value faster [31]. This is mainly because generative modeling handles missing and partial data better [32], as illustrated in Figure 1.5. Note that classification using generative models typically requires iteratively finding a good fit for the given observation, making it slower than discriminative models. In this thesis, we use neural fields, a generative model, for modeling scenes and motions. Chapter 2 will give more details on neural fields.

1.5 Contributions

This licentiate presents our initial progress for the project *learning data efficient representations for robotics and manipulation*. In this project, we have extensively used **neural fields** since they provide a versatile way to model scenes and motions, which will be discussed in Chapter 2. We explore how to model scenes and motions and demonstrate our approach for grasp synthesis and modeling movement primitives.

For grasp synthesis, we have shown that we can model local surfaces and

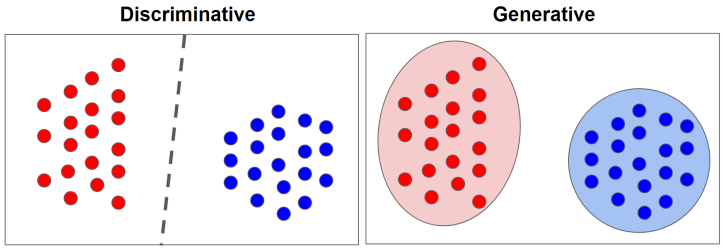

Figure 1.4: Generative and discriminative modeling of data. Generative methods are tasked to find the underlying distribution of the given data, while discriminative models are optimized to find the decision boundaries.

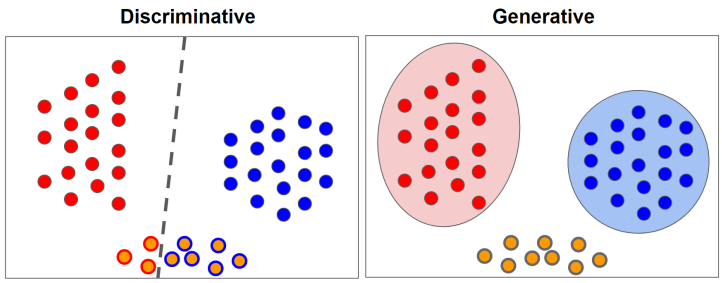

Figure 1.5: Generative and discriminative modeling for classification. Generative methods handle missing and partial data better compared to discriminative models.

use them as correspondences for grasp transfer, which will be explained in Chapter 3 and Paper A. In our experiments, we have shown that grasp poses generated by our method are more spatially precise and have higher grasp accuracy compared to the baseline. For modeling movement primitives, we have shown that we can jointly model scenes and motions as smooth functions of shared embeddings, which will be described in Chapter 4 and Paper B. Our method outperforms the baseline architecture in motion generation. Furthermore, for both of these tasks, our methods require few expert demonstrations, unlike the baseline methods.

Overall, we have shown that we can learn effective scene and motion representations that perform well in grasp synthesis and modeling movement primitives. The thesis outline is as follows:

- **Chapter 2:** Background on Neural Fields.
- **Chapter 3:** Using Local Surfaces as Correspondences for Grasp Transfer.
- **Chapter 4:** Joint Modeling of Scene and Motions.
- **Chapter 5:** Brief Summaries of the Papers Included in this Thesis.
- **Chapter 6:** Conclusion and Future work.

CHAPTER 2

Neural Fields

Visual computing requires models capable of reconstructing, synthesizing, and manipulating scenes, objects, and shapes. **Field** is a concept from physics widely used in modeling such representations to continuously parameterize them with respect to time and space [33]. Fields are scalar physical quantities defined in spatial and temporal coordinates, *e.g.* magnetic and gravitational fields. By modeling fields as functions, they can represent images by mapping pixel locations to their RGB values or 3-D shapes by mapping 3-D locations to their occupancy values. However, in practice, the field generation process does not have a known analytical form and, therefore, is performed by handcrafting or learning the underlying parameters for functions that model these fields.

One way to acquire representations for fields is through **neural fields**, which are coordinate-based neural networks that take spatial or temporal coordinates and parameterize fields [33]. These neural representations are intuitive to use and, therefore, have been used to represent scenes [34], 3-D shapes [35], [36], and images [37]. Furthermore, we have shown them to be strong representations for motion generation in Paper B. Neural fields have the following benefits:

- They can represent signals with arbitrary input/output dimensions [33].

- They can be designed to be infinitely differentiable, allowing them to be optimized and used for objectives that involve higher-order derivatives [37].

- The reconstruction process is often modality independent and, therefore, allows for representing signals from different modalities [38], [Paper B].

- They can automatically handle complex geometry processes such as filtering [39], deformation [40], and alignment [41], [Paper A].

In general, neural fields are versatile and powerful tools, and we have used them to model scenes and motions. In the rest of this section, we will first present the formulation of neural fields and explain some of their application domains. Then, we will discuss how neural fields are used as generative models. Finally, we will detail how they are trained and how spectral bias affects neural field training.

2.1 Formulation

Neural fields are neural networks that model smooth functions of spatio-temporal coordinates [33]. Given a field $\varphi : X \to Y$, they learn function mapping from coordinates $x \in X$ to scalar values $y \in Y$. These mappings can be defined in the reconstruction or sensor domain. In the reconstruction domain, the neural field maps world coordinates to the corresponding physical quantity, *e.g.* occupancy values. In the sensor domain, the neural field maps from sensor coordinates, such as pixel locations, to the corresponding measurement, such as RGB values. Due to their flexibility, neural fields can represent a wide range of scene and motion representations, as shown in Figure 2.1.

- **Scenes:** Scenes are modeled with neural fields often using radiance fields [34]. These are called neural radiance fields (NeRF). NeRF learns the $\mathbb{R}^5 \to \mathbb{R}^4$ mapping between 3-D location and 2-D viewing direction to 3-D color values, RGB, and 1-D volume density. Using differentiable volume rendering techniques, these values are composited into images. This allows optimizing NeRFs using rendering loss with images captured from different viewing directions. At inference time, NeRFs allow render-

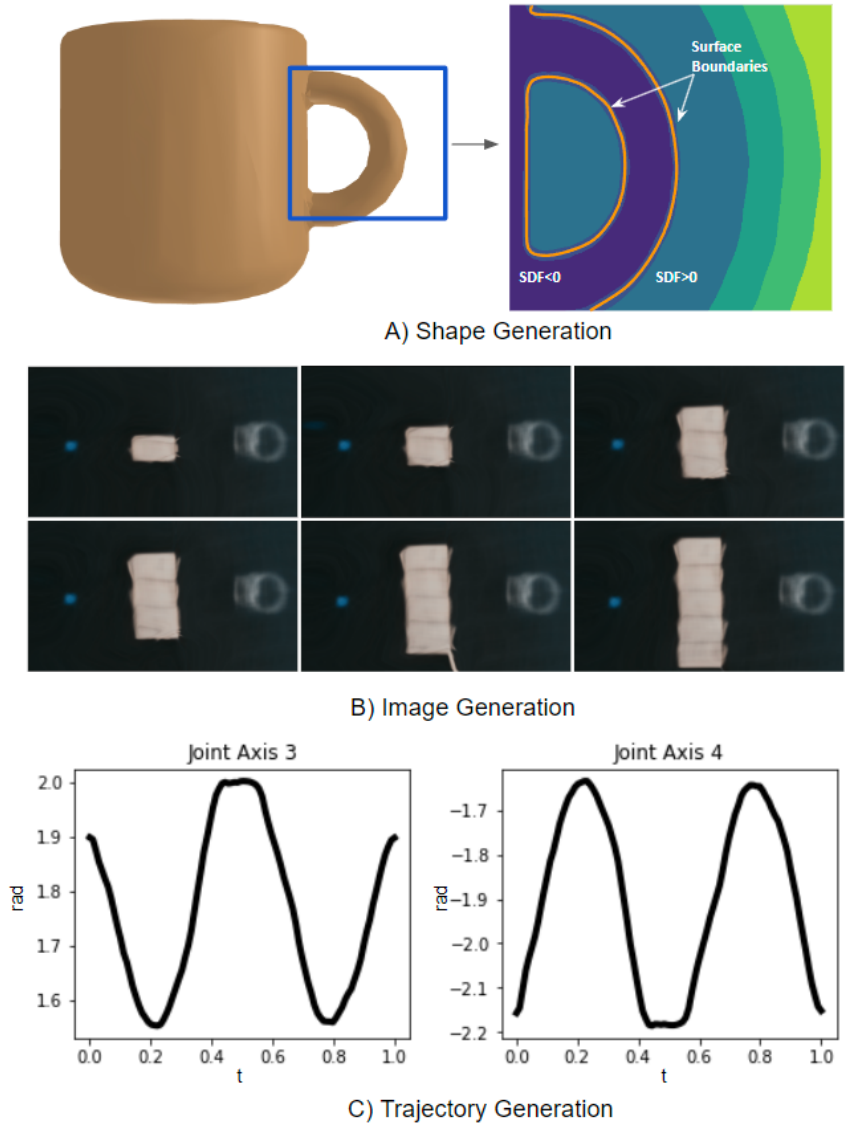

Figure 2.1: Neural Fields can model images, 3-D shapes, trajectories, and many more scene and motion representations.

ing images from novel viewing directions as long as they are trained with densely collected images. For more details, please refer to NeRF [34].

- **3-D Shapes:** A way to model 3-D shapes is by modeling their underlying implicit surface functions [35], [36], [42], *i.e.,* $f(x) = 0$ for 3-D spatial coordinates x on the modeled surface. Using marching cubes [43], we can reconstruct 3-D shapes from implicit surface functions. Implicit surfaces can represent all rigid shapes, including ones with sharp edges and holes. One of the popular implicit surface representations is signed distance functions (SDF). SDF maps each spatial coordinate $x \in X$ to the shortest signed distance value between the modeled surface and x: SDF yields negative values if x is inside the surface and positive values if x is outside the surface. This is shown in Figure 2.1A. In our work, we employ SDF for modeling shapes through neural fields.

- **Images:** Images are common visual representations easily acquired using cameras. We can model images as functions by mapping 2-D pixel locations to RGB values [37], $\mathbb{R}^2 \rightarrow \mathbb{R}^3$.

- **Motion Trajectories:** Motion trajectories are joint angles or end effector pose sequences. In this project, we model motion trajectories as mappings between temporal coordinates to corresponding n-dimensional joint or pose states, $\mathbb{R}^1 \rightarrow \mathbb{R}^n$.

- **Cost Function:** Cost function evaluates the performance of a system for given data. For robot control, the cost function evaluates the error value on the given state of the robot. Through gradient-based optimization, cost functions can be minimized to generate motion trajectories that move the robot to the goal locations.

  Cost functions have been commonly modeled as fields for robot control. For example, artificial potential fields have been used for obstacle avoidance [44]. Cost functions allow modeling motion as implicit functions, allowing multi-valued trajectory generation for tasks with multiple solutions [25]. In this project, we use neural fields to model tasks with cost functions estimated from LfD trajectories.

2.2 Generative Modeling with Neural Fields

The representations used for modeling scenes should be able to handle variations in the scenes. This is critical for robot grasping and manipulation. In traditional generative modeling, the goal is to obtain the generator function g that maps samples z, often referred to as latent vector or embedding, from a tractable distribution $\mathcal{Z}$, to samples Y [1] from an intractable distribution D, which we refer to as training data [45]. In neural fields, the spatial or temporal coordinates X of each sample Y are part of the generative modeling, which leads to the extended generator function $g(z, X) = Y$. This parameterization of the generator function allows for the generation of samples with arbitrary resolution and the application of different transformations to the input coordinates [33], such as deformation [40], [46] and alignment [41], [Paper A]. In this section, we will first describe how $z \in \mathcal{Z}$ is acquired and then detail how it is used in conditioning neural fields.

Acquisition of latent vector z

In deep learning, latent vectors $z \in \mathcal{Z}$ are generally acquired through encoder neural networks. In these architectures, the latent vector is generated by passing the observation of the scene through a neural network-based encoder $z = E(\mathcal{O})$. Depending on data type, Pointnet-based [47], [48] and ConvNet-based [49] architectures, such as ResNet [50], are used as encoder networks for point-cloud and image input, respectively. However, these methods require a large amount of data or various, often task-specific, data augmentation techniques to learn generalizable latent representations. Neural fields also employ an alternative way of acquiring latent vector z: through auto-decoders [35]. In our work, we have employed auto-decoders for modeling scene and motion representations.

In auto-decoders, at training time, each observation is assigned a latent vector often generated from a normal distribution with a small variance. These latent vectors are also called embeddings [2] as they are not intermediate representations of a network. These latent vectors are then optimized together with the network parameters with gradient descent during training through back-propagation. Network weights are fixed at inference time, and latent

[1] X is defined implicitly for Y, *e.g.* images with fixed resolution.

[2] The terms "latent vector" and "embedding" are often used interchangeably in deep learning

vectors for novel observations are found iteratively through optimization by back-propagating the reconstruction loss. In a way, the optimization step performs the encoding operation on the observations [33]. For a more detailed comparison between auto-decoders and auto-encoders, readers are referred to the supplementary material of DeepSDF [35].

Auto-decoders are slow compared to encoder-based architectures since they require iterative optimization for inference. However, they are not limited by observation format. In addition, we have found that they allow for acquiring representations that are easier to generalize through our experiments. When the changes in the modeled scene are smooth, auto-decoder-based architectures can generalize within the convex hull of the variations present within observed scenes by interpolating between the training embeddings. This is demonstrated in Paper B. Encoder-based architectures require more densely sampled training data to achieve the same results.

Conditioning of Neural Field with latent vector z

Given latent vectors z, there are multiple ways of conditioning neural fields. Two common ways are concatenation-based conditioning and using hypernetworks [51]. Note that neural fields are neural networks that take coordinates as input. Concatenation-based conditioning is through direct concatenation of coordinate and latent vectors [35], [36]. In our initial experiments for shape modeling, we have found that it is harder to optimize such neural fields. For example, the trained neural field for modeling mugs failed to capture shape details, such as holes in their handles. Instead, we have employed hypernetworks for modeling neural fields in our project.

Hypernetworks [51] are neural networks that predict the weights of another neural network. Hypernetworks allow learning smoother generator networks while requiring fewer trainable parameters to optimize than concatenation-based conditioning [52]. Hypernetworks provide additional modularity for the trained neural field by allowing each different shape, image, etc., to be represented as separate generator functions parameterized via hypernetwork. We have employed hypernetworks to condition neural fields for both papers included in this thesis. Compared to direct concatenation, hypernetworks have allowed us to model neural fields using multilayer perceptrons (MLP) with fewer layers. In Paper A, we use a 5-layer ReLU MLP for the generator network and multiple 2-layer ReLU MLPs for Hypernetwork. In comparison,

DeepSDF architecture employs a 12-layer MLP.

2.3 Training Neural Fields and Test Time Optimization

Neural field training used in our project is mostly very similar to the standard neural network training, with a few differences: (i) Due to auto-decoder architecture, it requires explicit handling of latent vector initialization, optimization, and usage; (ii) Since neural fields represent the modeled scene/motion based on coordinates, during training it requires coordinate sampling (Using all coordinates at each iteration may require a very high number of forward passes depending on the resolution of the observation. For example, an image with 256x256 resolution requires 65536 forward passes). Accordingly, this leads to the following changes in neural field training:

1. Initializing Embeddings: Initialize $z_i \in \mathcal{Z}$ for each observation i from the training dataset D.

2. Coordinate Sampling: Sample random subset of coordinates $(X_i, Y_i) \in D$ where X_i and Y_i are coordinates and their field values, respectively, for the observation with index i. For example, for shape observations, X_i corresponds to 3-D spatial coordinates, and Y_i corresponds to SDF values at these coordinates.

3. Prediction: For observation with index i, pass z_i to hypernetwork to predict the parameters of the generator network. Pass $x \in X_i$ to the generator network to predict $\hat{y} \in \hat{Y}_i$.

4. Optimization: Optimize z_i along with the hypernetwork parameters, and also the generator network parameters if it is only partially parameterized by hypernetwork, by minimizing the reconstruction loss between $\hat{y} \in \hat{Y}_i$ and $y \in Y_i$

Auto-decoder lacks encoders; therefore, it requires estimating $\hat{z}$ iteratively at inference time by minimizing the reconstruction loss, *i.e.,*

$$\hat{z} = \text{argmin}_{\hat{z}} \; L_{\text{rec}}(g_{\hat{z}}(X), Y), \tag{2.1}$$

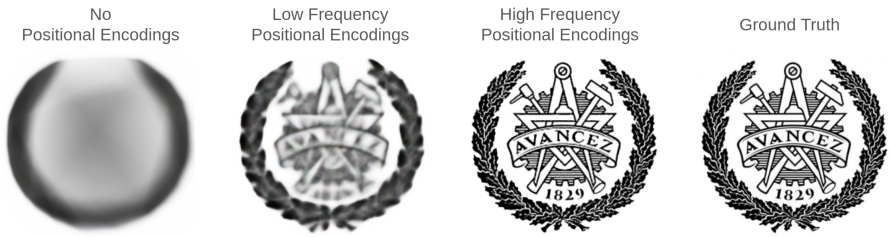

Figure 2.2: Illustration of spectral bias on image modeling. Neural fields are biased to learn low-frequency details of the observations when positional encodings are not used.

where $g_{\hat{z}}$ is the generator function parameterized by embedding $\hat{z}$. This is called test time optimization. At inference time, all network parameters are fixed, and $\hat{z}$ embedding is estimated similarly to how it is optimized during training.

2.4 Spectral Bias and Positional Encodings

Neural networks are biased towards learning simpler mappings between given input-output pairs in training, and this is called *spectral bias* [15]. ReLU MLPs have been shown to prioritize learning low-frequency features first [53]. A way to overcome this is through positional encodings [54]. Positional encodings map spatial coordinates, $x \in \mathbb{R}^n$, to higher-dimensional inputs. This allows neural fields to model complex signals more easily. A visualization of this phenomenon is shown in Figure 2.2. In this project, we have employed the positional encoding map

$$\Gamma(x) = [X, \Gamma_0(x), \Gamma_1(x), \ldots, \Gamma_{L-1}(x)] \in \mathbb{R}^{n+2nL},$$

where $\Gamma_m(x) = [\cos(2^m \pi x), \sin(2^m \pi x)] \in \mathbb{R}^{2n}$ is the m^{th} frequency band. However, there are benefits of using low-frequency positional encodings as well. Since they learn low-frequency features first, they are better at modeling smooth changes. However, high-frequency positional encodings are good at capturing fine details. Using these characteristics of positional encodings allows representations that are easier to generalize, as shown in Paper B, and handling geometry processing operations automatically, as shown in Paper A

with pose alignment. For this, in both of our works, we have employed coarse-to-fine approximation, as it allows utilizing the mentioned characteristics of positional encodings.

Coarse-to-fine Approximation

Low-frequency positional encodings are better for modeling large changes in the scene, and high-frequency positional encodings are better at capturing details. Therefore, there is a trade-off between modeling scenes with high- or low-frequency positional encodings. In our work, we use coarse-to-fine approximation to balance this trade-off.

In coarse to fine approximation, at training, low-frequency positional encodings are used first. Higher-frequency positional encodings are gradually introduced to optimization during training. This benefits our work in two ways:

- It allows the pose alignment process to be performed automatically. During initial optimization steps, it is easier to perform pose alignment of shapes with low-frequency shape information. As higher-frequency positional encodings are introduced, object poses are further refined with additional shape details. This is shown in Paper A.

- Coarse-to-fine approximation allows the learned representations to be smoother. This allows the scene representation process to be performed with less data. This is shown in Paper B.

A simple illustration of pose alignment is shown in Figure 2.3 where we train three different models on pose-perturbed images with label 6 from the MNIST dataset [55]. The model with no positional encodings reconstructs the images in an overly smooth way, and the model with high-frequency positional encodings fails to perform the pose alignment task. However, the model that utilizes coarse-to-fine approximation manages to capture image details while aligning the poses of images, acquiring the best performance among these models. Note that no additional loss function aiding the alignment of the image is utilized during the training of these models.

We adopt the coarse-to-fine approximation method proposed in bundle adjusted radiance fields [41]. Coarse-to-fine approximation is achieved by applying a smoothing mask to the positional encodings. We multiply each frequency

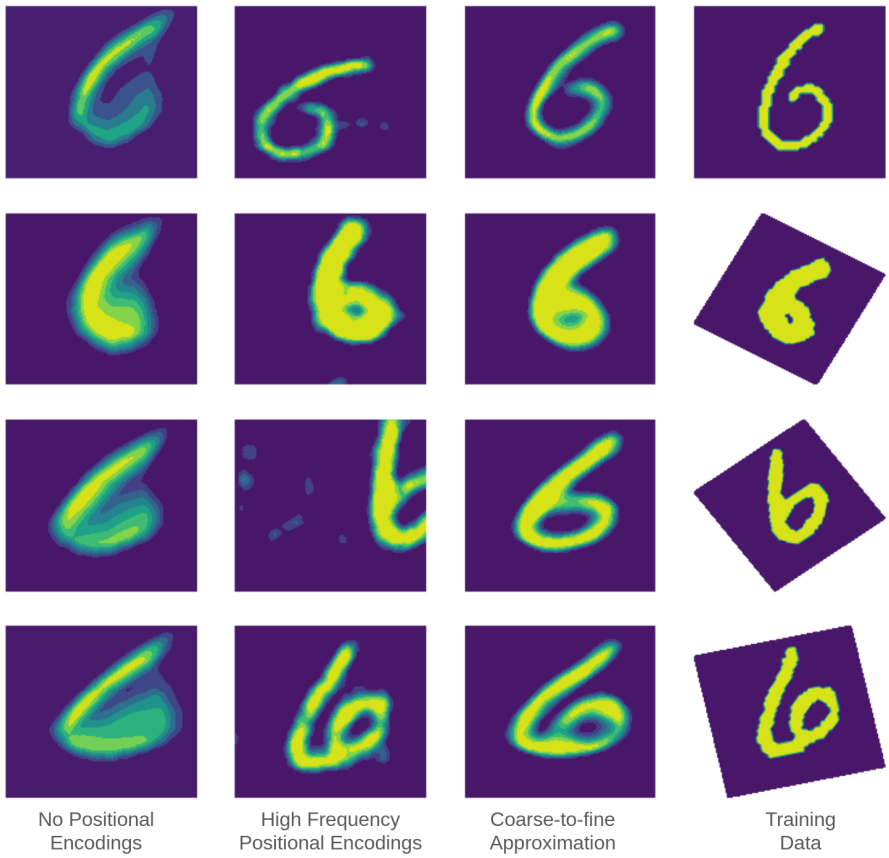

Figure 2.3: Illustration of effect of coarse-to-fine approximation for pose alignment. Coarse-to-fine approximation allows utilization of both low- and high-frequency positional encodings and acquires the best pose alignment and image reconstruction performance.

band term in the positional encoding map with a dynamically changing weight according to the current training iteration, *i.e.,*

$$\Gamma_m(x;\alpha) = w_m(\alpha) \cdot [\cos(2^m \pi x), \sin(2^m \pi x)]$$

where $\alpha \in [0, L]$ is a controllable parameter and weight w_m is given by:

$$w_m(\alpha) = \begin{cases} 0, & \text{if } \alpha < m \\ \dfrac{1 - cos((\alpha - m)\pi)}{2}, & \text{if } 0 \leq \alpha - m < 1 \\ 1, & \text{if } \alpha - m \geq 1 \end{cases}$$

During training, we slowly increase α according to the current epoch of the training. Eventually, $\alpha = L$, and the training continues with full positional encodings.

CHAPTER 3

Local Surface Models as Correspondences

In computer vision, correspondences refer to matching features across different scene observations, such as images or point-clouds. Ideally, these features should be unique and reliable to identify. They are usually used in tracking, object detection, image registration, and many more tasks.

Correspondences also play a crucial role in many robotics tasks, such as 3-D scene reconstruction, visual servoing, grasp synthesis, and simultaneous localization and mapping. However, the success of these tasks relies on the quality of the correspondence. Ideally, correspondences should be stable, reliable, distinctive, and accurate. When there is a camera pose or an illumination change, correspondences should stay stable, and their positions should not fluctuate, disappear, or change.

Correspondences are often defined in the form of key points that are estimated from images or point-clouds. Common ways to acquire such key points are through classical computer vision techniques, such as SIFT [56] or SURF [57] features, or deep learning-based ones, such as dense visual descriptors [58], [59]. The advantage of the former methods is that they do not need training; however, they generate a high number of key points, and it may be hard to identify the relevant ones. Furthermore, such methods cannot reli-

ably estimate accurate key points in poorly textured scenes. In comparison, the latter methods map each pixel in the observed image to feature vectors. By minimizing the difference between feature vectors, these methods provide a more straightforward process for identifying relevant key points. However, key points for both types of methods are defined based on pixels, leading to ambiguity in orientation estimation, which is crucial for accurate robot manipulation. Previously, this ambiguity has been resolved in two ways: by mapping each key point to multiple 3-D points [29] or by mapping key points to 3-D shapes [60]. These methods allow for more accurate key point estimation; however, they rely on category-level information, *i.e.,* they rely on object categories and cannot generalize to novel object types.

Instead, we propose defining correspondences based on local surface models and using these correspondences to transfer grasp poses. Defining key points based on local surfaces allows transferring grasp poses across object categories that have geometrically similar surfaces/parts. For example, the grasp pose defined on the handle or rim of a mug can be transferred to objects such as bags or bowls, as illustrated in Figure 3.1. Compared with previous approaches that use correspondences, estimated grasp poses are more spatially accurate and transferable across object categories, as shown in Paper A.

In the rest of this chapter, we will first give a brief background on how shape models have been utilized in robotics and the limitations of existing methods. Then, we explain how we model local surfaces in a way that allows aligning and estimating their poses automatically. Finally, we finish the chapter by discussing how local surface models can be used as correspondences in grasp transfer tasks and beyond.

3.1 Shape Models in Robotics

Shape modeling is pivotal in robotics due to its connection with 3-D perception. For example, we can identify shape models that match with a given point cloud scene observation. Since shape model and scene observation share modalities, the quality of the match is connected directly to how well the model fits the observation, and this metric is interpretable. Therefore, shape models have been utilized in grasp modeling in the form of shape primitives [61], [62], smooth differentiable functions (*e.g.* Grasp Moduli Spaces [63], [64]), and prototype parts [65]. However, these shape models provide limited gener-

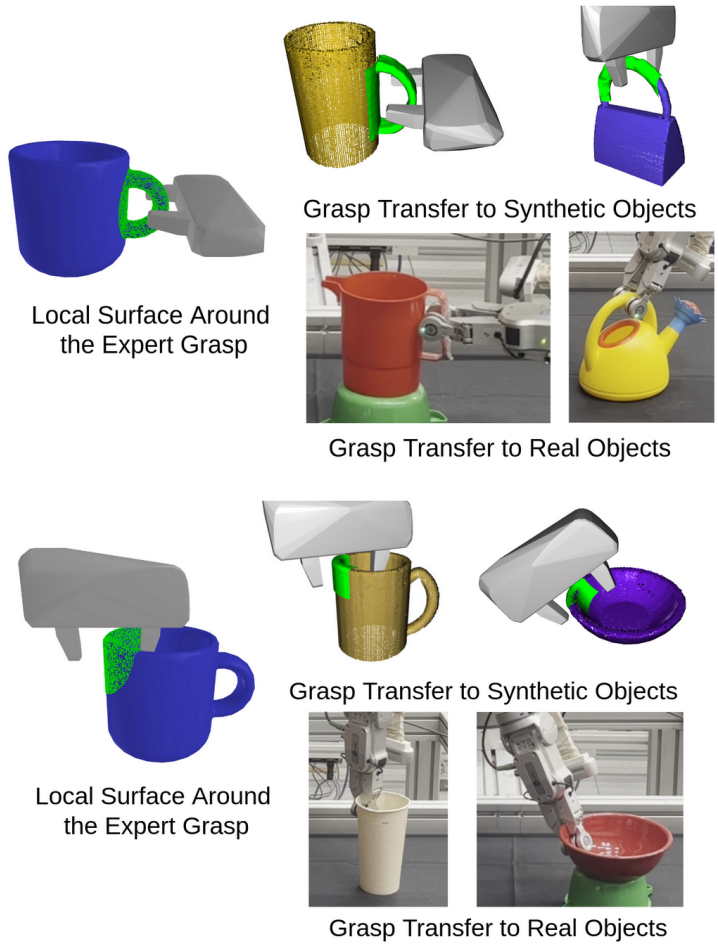

Figure 3.1: Local surface models as correspondences. We can use local surface models to identify local surfaces that are geometrically similar to the modeled ones and then use identified surfaces for grasp transfer.

alization, and they struggle when the observations deviate significantly from the provided primitives or are partial. Finally, neural object descriptors have been used for mapping point-cloud inputs to shape models [60]. However, it was limited to the object category of the training shapes.

3.2 Local Surface Modeling with Neural Fields

Due to the explained reasons, we model 3-D shapes on the local surface level using neural fields. Neural fields allow geometry processes to be integrated into the optimization procedure, as explained in Chapter 2. This is necessary, as object parts do not appear in datasets in their canonical poses. As a solution, we perform pose alignment using Special Euclidean groups in three dimensions ($\mathbf{SE}(3)$) as part of the optimization process.

$\mathbf{SE}(3)$ Group and Pose Alignment

Special Euclidean is a Lie group, and Lie Groups are topological groups that combine the structure of a group with that of a differentiable manifold. This means that its elements are smoothly parameterized and each element has an inverse. Furthermore, it allows operations such as multiplication [66].

$\mathbf{SE}(3)$ group is a smooth manifold and can be used in end-to-end optimization. Each element in $\mathbf{SE}(3)$ group, $\mathbf{SE}(3)$ transformation, has an associated Lie algebra, $\mathfrak{se}(3)$, which are 6D vectors. We can map Lie algebras to $\mathbf{SE}(3)$ transformations using the exponential map $\mathfrak{se}(3) \rightarrow \mathbf{SE}(3)$ that continuously lie on $\mathbf{SE}(3)$ manifolds [66]. Given $\beta = \langle \omega, t \rangle \in \mathfrak{se}(3)$ a vector in Lie algebra, where $\omega \in \mathbb{R}^3$ and $t \in \mathbb{R}^3$ are 3-dimensional vectors, ω and t parameterize rotation and translation, respectively. The corresponding transformation T for β is estimated using the Rodrigues formula. More specifically, $T = \left[\begin{array}{c|c} R_{3\times3} & v_{3\times1} \end{array} \right] \in \mathbf{SE}(3)$ with $v = Vt$; and R and V are written as

$$R = e^{[\omega]} = I_3 + \frac{\sin\theta}{\theta}[\omega] + \frac{1-\cos\theta}{\theta^2}[\omega]^2, \tag{3.1}$$

$$V = I_3 + \frac{1-\cos\theta}{\theta^2}[\omega] + \frac{1-\sin\theta}{\theta^3}[\omega]^2, \tag{3.2}$$

where $[\omega]$ is the skew-symmetric matrix created from the vector ω using the hat operator (Eq. 7.4 on [66]), and $\theta = \|\omega\|$. To estimate this transformation matrix T, the Taylor series expansion is used for linearization. This formulation is well defined, surjective, and allows for estimating the alignment transformation through gradient-based optimization of β. At $\beta = 0$, T corresponds to the identity transformation.

Consider the neural field-based generator function $g(z, X) = Y$. The transformation operation can be performed by applying transformation T to X by

$X' = (T \begin{bmatrix} X & 1 \end{bmatrix}^T)^T$. Then the final generator function becomes $g(z, X') = Y$ and optimization process becomes

$$(\hat{z}, \hat{\beta}) = \text{argmin}_{(\hat{z},\hat{\beta})} \, L_{\text{rec}}(g(\hat{z}, X'), Y) \tag{3.3}$$

During the training of the shape models, a β vector is assigned to each instance in the training dataset, and these vectors are optimized along with the network parameters. This leads to reconstructed surfaces that are well-aligned with each other, as shown in Paper A.

Local Surface Modeling

A Training dataset that consists of object parts is required for training local surface models. However, such datasets are not readily available. Instead, existing category-level shape datasets can be utilized to learn local surface models. The main idea is that objects from the same category will contain parts that are geometrically similar at the same relative poses with respect to the center of the object, given these objects are in their canonical poses.

We employ one reference frame given in the form of a grasp demonstration on a representative instance of an object dataset. Note that such reference frames can be provided in other ways using other waypoints (*e.g.* bottlenecks [67]) in demonstrated trajectories. The reference frame is then used to extract local surfaces from the given shape categories. We assume that the extracted surfaces will be similar, yet we consider that they may not align well with each other. These surfaces are then used for training the neural fields.

3.3 Local Surfaces as Correspondences

The learned local surface models are used as correspondence for identifying geometrically similar surfaces on novel objects for grasp transfer, which is shown in detail in Paper A. However, note that our process is generic and, therefore, can use different ways of acquiring reference frames. For example, previous works acquire waypoints from demonstration by considering when the robot slows down or makes contact with the objects [68]. Furthermore, correspondences can be used for tasks beyond pick and place. In the future, we will explore how we can use shape models to learn more complex manipulation skills.

CHAPTER 4

Joint Modeling of Scenes and Motions

Grasping is just one aspect of robot manipulation. Robots are required to perform diverse manipulation skills, including lifting, rotating, throwing, and more, while navigating through various obstacles to complete their tasks. For this reason, many robotic tasks require motion modeling in addition to perception, often by coupling perception and control. As described in Chapter 2, motion can also be modeled with neural fields.

Motion modeling is commonly achieved through task parameters and images. However, both task parameter-based and image-based modeling have their limitations. Task parameters are required to describe the scene sufficiently. They have to be provided by the user or need to be estimated, which requires further engineering effort. Modeling motion with images requires significantly more data to generalize to variations in the scene. The data need to be densely sampled compared to using task parameters.

For example, consider the wall avoidance task shown in Figure 4.1. In this task, the robot has to go from the left or right of the obstacles in the middle and drop the cube into the container. The obstacle's width and the container's location vary in different scenes. We can model this task with obstacle width and container location. However, a feature extractor model or the user must

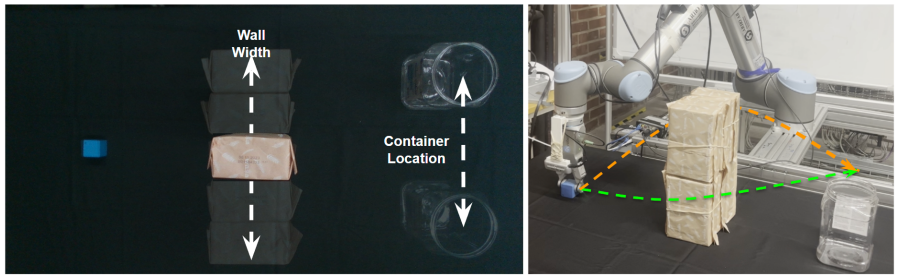

Figure 4.1: Wall avoidance task. The task parameters are on the left, and motion trajectories are shown on the right.

provide these task parameters. Alternatively, we can directly use the scene observations in the form of images to remove this requirement. However, this requires collecting additional robotic data with various obstacle widths and container locations.

We can jointly model scenes and motions using a generative approach with neural fields to get the best of both worlds. As described in Chapter 2, in generative modeling, the goal is to obtain a generator function g that maps latent vectors sampled from a tractable distribution $z \in \mathcal{Z}$ to scene observations/motion trajectories from an intractable distribution D, which is the training dataset. In addition, utilizing the spectral bias of neural fields allows learning scenes and motions as smooth functions of embeddings z. This is going to be described in more detail in upcoming sections.

In this project, we have used LfD to acquire datasets of observation-trajectory pairs for different tasks and use these datasets to model the movement primitives with neural fields. Movement primitives [19]–[23] are policy representations that encode continuous joint or state trajectories. They are often used for LfD [16], [17] to teach robots complex motor skills efficiently. In the rest of this chapter, we explain how we achieve data efficiency by modeling scenes and motions as smooth functions. Then, we discuss how to jointly model scenes and motions with shared embeddings.

4.1 Learning Scenes and Motions as Smooth Functions

We consider LfD tasks where it is possible to model corresponding scenes and motions with task parameters p. Accordingly, we can represent these scenes and motions with neural field-based generator functions $f_p(X_{\text{scene}})$ and $g_p(X_{\text{motion}})$, respectively, where the parameterization of the generator functions rely on task parameters. $X_{\text{scene}} \in \mathbb{R}^m$ corresponds to spatial coordinates of the scene observations, such as pixels of images. As for motions, we can either model trajectories directly or model them as the minimization of cost functions. X_{motion} will correspond to temporal coordinates for the former case, $X_{\text{motion}} \in \mathbb{R}^1$, joint angles/end effector positions for the latter case, $X_{\text{motion}} \in \mathbb{R}^n$, respectively. For more scene and motion modeling details, please refer to Chapter 2.

Neural fields behave differently depending on the frequency of positional encodings. Low-frequency positional encodings are good for representing the scene changes smoothly. However, they cannot capture the details necessary to reconstruct scene observations. Higher frequency positional encodings allow for representing such details. However, they require a higher number of training data to generalize. To get the best of both worlds, we separate our neural fields into deformation and template fields that use low-frequency and high-frequency positional encodings, respectively. This allows for capturing the details necessary to represent the scene while preserving the smoothness provided by the low-frequency positional encodings. In this formulation, the template neural field models the canonical scene $T(x)$, and the deformation field models the deformation in canonical scene $D_p(x) = \Delta x$:

$$f_p(x) = T(x + D_p(x)).$$

Note that only the deformation field is parameterized and not the template field. For this reason, modeling the deformation field with low-frequency positional encodings allows generalization with fewer scene observations. Furthermore, modeling the template field with high-frequency positional encodings allows for capturing fine details. This is illustrated in Figure 4.2. In this figure, the first row illustrates ground truth images, where images with red borders are not seen during training. Row b) and c) show images reconstructed without deformation fields, where the underlying models use low-frequency and

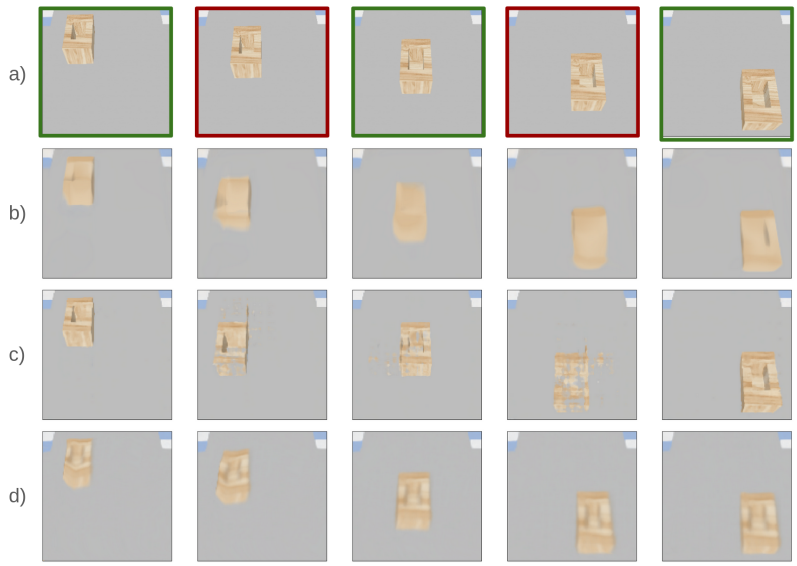

Figure 4.2: This figure illustrates the advantage of using deformation fields for scene modeling. Ground truth training images are marked with a green border, and unseen ones are marked with a red border in row a). Row b) and c) show the reconstructed images using neural fields with low- and high-frequency positional encodings, respectively. Row d) shows the images reconstructed using low-frequency deformation field and high-frequency template fields.

high-frequency positional encodings, respectively. Row b) fails to capture details, and row c) cannot model scene changes smoothly. However, using deformation fields and template fields with low- and high-frequency positional encodings, we can model smooth scene changes while still capturing fine details, as illustrated in row d).

4.2 Shared Embeddings for Joint Modeling of Scenes and Motions

Identifying and estimating task parameters for any given scene is not trivial. Finding the exact value of task parameters could be challenging even during

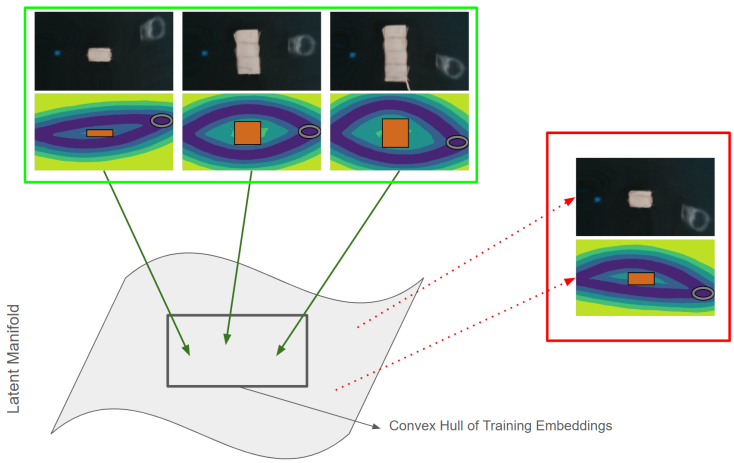

Figure 4.3: Illustration of how latent manifold maps to scene and motion observations. The trained network can correctly map to corresponding scene and motion observations within the convex hull of training embeddings. However, outside of the convex hull, this is not guaranteed.

training data collection. Instead, we can use the auto-decoder framework presented in Chapter 2. Accordingly, we assign an embedding $z \in \mathbb{R}^k$ to each demonstration. During training, these embeddings are optimized together with network parameters. This allows obtaining scene and motion functions parametrized with these embeddings. In short, we can model a task with an unknown task parameter vector p with learned task embeddings. With this new formulation, tasks are equivalently represented with neural fields $F_z(x)$ and $G_z(t)$ parameterized by z embedding:

$$(F_z(x), G_z(t)) \equiv (f_p(x), g_p(t)) \tag{4.1}$$

With this formulation, ideally, the mapping between the embeddings $z \in \mathcal{Z}$ and observations for scene and motion should be one-to-one. However, this is not guaranteed and can lead to the embeddings $z \in \mathcal{Z}$ to map to unrelated scene and motion observations. However, this can be mitigated by sampling z embeddings from the convex hull of the embeddings utilized in the training process. A visualization of this phenomena is shown in Figure 4.3. Note that regardless of where we sample $z \in \mathcal{Z}$, the neural fields generate scenes and

motions similar to the ones seen in the training dataset; therefore, this is not an issue when we are not concerned with joint modeling.

To constrain z to the convex hull of embeddings utilized in the training dataset, we change the optimization problem as follows. If there are n demonstrations in the training dataset, the optimization problem becomes:

$$\text{argmin}_{\alpha}\ L_{\text{scene}}(F_z, X) \text{ where } z = \sum_{i=0}^{n-1} \alpha_i z_i, \tag{4.2}$$

where $\alpha = [\alpha_0, ..., \alpha_{n-1}]$, $\sum_{i=0}^{i=n-1} \alpha_i = 1$, and $\alpha_i \geq 0$. z_i corresponds to the learned embedding of the expert demonstration i. With this equation, z is not directly optimized but estimated as the weighted sum of demonstration embeddings.

In summary, we can train neural fields to generate scene observations and motion trajectories from the shared embeddings of the corresponding expert demonstrations. Using these neural fields, we can then generate accurate motion trajectories for novel scenes by identifying embeddings that accurately reconstruct the scene. This is shown in Paper B.

CHAPTER 5

Summary of included papers

This chapter provides a summary of the included papers.

5.1 Paper A

Ahmet Ercan Tekden, Marc Peter Deisenroth, Yasemin Bekiroglu
Grasp Transfer based on Self-Aligning Implicit Representations of Local Surfaces
IEEE Robotics and Automation Letters, vol. 8, no. 4, Oct. 2023.
 DOI: 10.1109/LRA.2023.3306272 .

This paper presents a grasp transfer method that allows transferring a grasp experience or a demonstration to a novel object that shares local surface similarities with objects the robot has previously encountered. For this, we propose an approach based on neural fields that allow us to model local surfaces. The method is trained entirely in simulation and validated in simulation and real-world experiments. The simulation experiments show that the grasp transfer is spatially precise and has a higher grasp accuracy compared to the baseline. Furthermore, in both simulation and real-world experiments,

we show that the proposed method can perform grasp transfer to novel objects from unseen categories.

5.2 Paper B

Ahmet Ercan Tekden, Marc Peter Deisenroth, Yasemin Bekiroglu
Neural Field Movement Primitives for Joint Modelling of Scenes and Motions
IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS),2023, Detroit, USA.

This paper presents a learning from demonstration method where we propose jointly modeling scenes and motions using a generative approach with neural fields. Our model learns to generate each scene and motion trajectory from the shared embedding of the corresponding expert demonstration. At inference time, it generates accurate motion trajectories for novel scenes by identifying embeddings that accurately reconstruct the scene. Our method is evaluated in two simulation and two real-world experiments. Our simulation experiments show that our method outperforms the baseline approaches and generalizes to novel scenes. Our real-world experiments demonstrated that our method can successfully model multi-valued trajectories, is robust to the distractor objects introduced at inference time, and can generate 6D motions.

CHAPTER 6

Concluding Remarks and Future Work

This thesis presents our progress on *learning efficient data representations for robotics and manipulation*. We have used **neural fields** as our choice of representations and explored how to model scenes and motions. More specifically,

- we use a single expert demonstration to extract local surfaces from an object category. Using extracted surfaces, we trained local surface models and associated each local surface model with different grasp types. We show that we can use learned local surface models to transfer grasp poses to novel objects in a data-efficient way. This approach has proven effective in transferring grasp across object categories both in simulation and real-world environments despite being trained only with synthetic objects. Furthermore, it is shown that the predicted grasp poses are more spatially precise and have a higher grasp accuracy compared to the baseline.

- we show that we can jointly model scenes and motions to model movement primitives. The proposed method uses a small number of expert demonstrations compared to baseline approaches while not requir-

ing task parameters to be provided. Our experiments show that our method performs better than baseline approaches, is flexible and can model multi-valued trajectories, is robust to distractor objects, and can use SDF-based scene representations.

In future work, we plan to study learning multi-finger manipulation skills and utilization of different sensor and task modalities. Furthermore, we would like to investigate how we can attain even higher data efficiency. More specifically, the planned future work is as follows:

- How to learn multi-finger manipulation skills in a data-efficient way. Specifically, (i) we would like to explore how to map shape models to fingertip locations and (ii) investigate how to model dexterous manipulation skills.
- We plan to increase the data efficiency of our methods further by utilizing compositionality.
- We plan to investigate how we can use tactile feedback and effect-based conditioning for trajectory generation.

References

[1] D. Nguyen-Tuong and J. Peters, "Model learning for robot control: A survey," *Cognitive processing*, vol. 12, pp. 319–340, 2011.

[2] S. Dragiev, M. Toussaint, and M. Gienger, "Gaussian process implicit surfaces for shape estimation and grasping," in *2011 IEEE International Conference on Robotics and Automation*, IEEE, 2011, pp. 2845–2850.

[3] S. Schaal, "Is imitation learning the route to humanoid robots?" *Trends in cognitive sciences*, vol. 3, no. 6, pp. 233–242, 1999.

[4] J. Kober, J. A. Bagnell, and J. Peters, "Reinforcement learning in robotics: A survey," *The International Journal of Robotics Research*, vol. 32, no. 11, pp. 1238–1274, 2013.

[5] B. Apolloni, A. Ghosh, F. Alpaslan, and S. Patnaik, *Machine learning and robot perception*. Springer Science & Business Media, 2005, vol. 7.

[6] J. Peters, R. Tedrake, N. Roy, and J. Morimoto, "Robot learning," in *Encyclopedia of Machine Learning*, C. Sammut and G. I. Webb, Eds. Boston, MA: Springer US, 2010, pp. 865–869, ISBN: 978-0-387-30164-8.

[7] J. H. Connell and S. Mahadevan, *Robot learning*. Springer Science & Business Media, 2012, vol. 233.

[8] R. Newbury, M. Gu, L. Chumbley, *et al.*, "Deep learning approaches to grasp synthesis: A review," *IEEE Transactions on Robotics*, 2023.

[9] R. Platt, "Grasp learning: Models, methods, and performance," *Annual Review of Control, Robotics, and Autonomous Systems*, vol. 6, pp. 363–389, 2023.

[10] A. Dearden and Y. Demiris, "Learning forward models for robots," in *IJCAI*, vol. 5, 2005, p. 1440.

[11] G. Dulac-Arnold, N. Levine, D. J. Mankowitz, *et al.*, "Challenges of real-world reinforcement learning: Definitions, benchmarks and analysis," *Machine Learning*, vol. 110, no. 9, pp. 2419–2468, 2021.

[12] L. P. Kaelbling, "The foundation of efficient robot learning," *Science*, vol. 369, no. 6506, pp. 915–916, 2020.

[13] E. S. Spelke, K. Breinlinger, J. Macomber, and K. Jacobson, "Origins of knowledge.," *Psychological review*, vol. 99, no. 4, p. 605, 1992.

[14] J. B. Tenenbaum, C. Kemp, T. L. Griffiths, and N. D. Goodman, "How to grow a mind: Statistics, structure, and abstraction," *science*, vol. 331, no. 6022, pp. 1279–1285, 2011.

[15] Y. Cao, Z. Fang, Y. Wu, D.-X. Zhou, and Q. Gu, "Towards understanding the spectral bias of deep learning," in *30th International Joint Conference on Artificial Intelligence (IJCAI 2021)*, 2021, pp. 2205–2211.

[16] B. D. Argall, S. Chernova, M. Veloso, and B. Browning, "A survey of robot learning from demonstration," *Robotics and autonomous systems*, vol. 57, no. 5, pp. 469–483, 2009.

[17] S. Schaal, "Learning from demonstration," *Advances in neural information processing systems*, vol. 9, 1996.

[18] M. Tavassoli, S. Katyara, M. Pozzi, *et al.*, "Learning skills from demonstrations: A trend from motion primitives to experience abstraction," *IEEE Transactions on Cognitive and Developmental Systems*, 2023.

[19] A. J. Ijspeert, J. Nakanishi, H. Hoffmann, *et al.*, "Dynamical movement primitives: Learning attractor models for motor behaviors," *Neural computation*, vol. 25, no. 2, pp. 328–373, 2013.

[20] A. Paraschos, C. Daniel, J. R. Peters, and G. Neumann, "Probabilistic movement primitives," *Advances in neural information processing systems*, vol. 26, 2013.

[21] S. Calinon, "A tutorial on task-parameterized movement learning and retrieval," *Intelligent service robotics*, vol. 9, pp. 1–29, 2016.

[22] M. Y. Seker, M. Imre, J. H. Piater, and E. Ugur, "Conditional neural movement primitives.," in *Robotics: Science and Systems*, vol. 10, 2019.

[23] Y. Zhou, J. Gao, and T. Asfour, "Learning via-point movement primitives with inter-and extrapolation capabilities," in *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, IEEE, 2019, pp. 4301–4308.

[24] F. Torabi, G. Warnell, and P. Stone, "Behavioral cloning from observation," in *Proceedings of the 27th International Joint Conference on Artificial Intelligence*, 2018, pp. 4950–4957.

[25] P. Florence, C. Lynch, A. Zeng, *et al.*, "Implicit behavioral cloning," in *Conference on Robot Learning*, PMLR, 2022, pp. 158–168.

[26] A. Zeng, P. Florence, J. Tompson, *et al.*, "Transporter networks: Rearranging the visual world for robotic manipulation," in *Conference on Robot Learning*, PMLR, 2021, pp. 726–747.

[27] P. R. Florence, L. Manuelli, and R. Tedrake, "Dense object nets: Learning dense visual object descriptors by and for robotic manipulation," in *Conference on Robot Learning*, PMLR, 2018, pp. 373–385.

[28] L. Yen-Chen, P. Florence, J. T. Barron, *et al.*, "Nerf-supervision: Learning dense object descriptors from neural radiance fields," in *2022 International Conference on Robotics and Automation (ICRA)*, IEEE, 2022, pp. 6496–6503.

[29] A. Simeonov, Y. Du, A. Tagliasacchi, J. B. Tenenbaum, *et al.*, "Neural descriptor fields: Se (3)-equivariant object representations for manipulation," in *2022 International Conference on Robotics and Automation (ICRA)*, IEEE, 2022, pp. 6394–6400.

[30] D. Foster, *Generative deep learning.* " O'Reilly Media, Inc.", 2022.

[31] A. Ng and M. Jordan, "On discriminative vs. generative classifiers: A comparison of logistic regression and naive bayes," *Advances in neural information processing systems*, vol. 14, 2001.

[32] I. Ulusoy and C. M. Bishop, "Generative versus discriminative methods for object recognition," in *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)*, IEEE, vol. 2, 2005, pp. 258–265.

[33] Y. Xie, T. Takikawa, S. Saito, *et al.*, "Neural fields in visual computing and beyond," *Computer Graphics Forum*, 2022, ISSN: 1467-8659.

[34] B. Mildenhall, P. P. Srinivasan, M. Tancik, *et al.*, "Nerf: Representing scenes as neural radiance fields for view synthesis," in *European conference on computer vision*, Springer, 2020, pp. 405–421.

[35] J. J. Park, P. Florence, J. Straub, R. Newcombe, and S. Lovegrove, "Deepsdf: Learning continuous signed distance functions for shape representation," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2019, pp. 165–174.

[36] L. Mescheder, M. Oechsle, M. Niemeyer, *et al.*, "Occupancy networks: Learning 3d reconstruction in function space," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2019, pp. 4460–4470.

[37] V. Sitzmann, J. Martel, A. Bergman, *et al.*, "Implicit neural representations with periodic activation functions," *Advances in Neural Information Processing Systems*, vol. 33, pp. 7462–7473, 2020.

[38] Y. Du, K. Collins, J. Tenenbaum, and V. Sitzmann, "Learning signal-agnostic manifolds of neural fields," *Advances in Neural Information Processing Systems*, vol. 34, pp. 8320–8331, 2021.

[39] G. Yang, S. Belongie, B. Hariharan, and V. Koltun, "Geometry processing with neural fields," in *Thirty-Fifth Conference on Neural Information Processing Systems*, 2021.

[40] Y. Deng, J. Yang, and X. Tong, "Deformed implicit field: Modeling 3d shapes with learned dense correspondence," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2021, pp. 10 286–10 296.

[41] C.-H. Lin, W.-C. Ma, A. Torralba, and S. Lucey, "Barf: Bundle-adjusting neural radiance fields," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2021, pp. 5741–5751.

[42] Z. Murvanidze, M. P. Deisenroth, and Y. Bekiroglu, "Enhanced gpis learning based on local and global focus areas," *IEEE RA-L*, vol. 7, no. 4, pp. 11 759–11 766, 2022.

[43] W. E. Lorensen and H. E. Cline, "Marching cubes: A high resolution 3d surface construction algorithm," *ACM siggraph computer graphics*, vol. 21, no. 4, pp. 163–169, 1987.

[44] O. Khatib, "Real-time obstacle avoidance for manipulators and mobile robots," *The international journal of robotics research*, vol. 5, no. 1, pp. 90–98, 1986.

[45] L. Ruthotto and E. Haber, "An introduction to deep generative modeling," *GAMM-Mitteilungen*, vol. 44, no. 2, e202100008, 2021.

[46] K. Park, U. Sinha, J. T. Barron, *et al.*, "Nerfies: Deformable neural radiance fields," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2021, pp. 5865–5874.

[47] C. R. Qi, H. Su, K. Mo, and L. J. Guibas, "Pointnet: Deep learning on point sets for 3d classification and segmentation," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 652–660.

[48] C. R. Qi, L. Yi, H. Su, and L. J. Guibas, "Pointnet++: Deep hierarchical feature learning on point sets in a metric space," *Advances in neural information processing systems*, vol. 30, 2017.

[49] Y. LeCun, Y. Bengio, *et al.*, "Convolutional networks for images, speech, and time series," *The handbook of brain theory and neural networks*, vol. 3361, no. 10, p. 1995, 1995.

[50] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 770–778.

[51] D. Ha, A. M. Dai, and Q. V. Le, "Hypernetworks," in *International Conference on Learning Representations*, 2017.

[52] T. Galanti and L. Wolf, "On the modularity of hypernetworks," *Advances in Neural Information Processing Systems*, vol. 33, pp. 10 409–10 419, 2020.

[53] N. Rahaman, A. Baratin, D. Arpit, *et al.*, "On the spectral bias of neural networks," in *Proceedings of the 36th International Conference on Machine Learning*, vol. 97, PMLR, 2019, pp. 5301–5310.

[54] M. Tancik, P. Srinivasan, B. Mildenhall, *et al.*, "Fourier features let networks learn high frequency functions in low dimensional domains," *Advances in Neural Information Processing Systems*, vol. 33, pp. 7537–7547, 2020.

[55] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998.

[56] D. G. Lowe, "Object recognition from local scale-invariant features," in *Proceedings of the seventh IEEE international conference on computer vision*, Ieee, vol. 2, 1999, pp. 1150–1157.

[57] H. Bay, T. Tuytelaars, and L. Van Gool, "Surf: Speeded up robust features," in *Computer Vision–ECCV 2006: 9th European Conference on Computer Vision, Graz, Austria, May 7-13, 2006. Proceedings, Part I 9*, Springer, 2006, pp. 404–417.

[58] T. Schmidt, R. Newcombe, and D. Fox, "Self-supervised visual descriptor learning for dense correspondence," *IEEE Robotics and Automation Letters*, vol. 2, no. 2, pp. 420–427, 2016.

[59] L. Manuelli, W. Gao, P. Florence, and R. Tedrake, "Kpam: Keypoint affordances for category-level robotic manipulation," in *The International Symposium of Robotics Research*, Springer, 2019, pp. 132–157.

[60] E. Sucar, K. Wada, and A. Davison, "Nodeslam: Neural object descriptors for multi-view shape reconstruction," in *2020 International Conference on 3D Vision (3DV)*, IEEE, 2020, pp. 949–958.

[61] A. T. Miller, S. Knoop, H. I. Christensen, and P. K. Allen, "Automatic grasp planning using shape primitives," in *2003 IEEE International Conference on Robotics and Automation (Cat. No. 03CH37422)*, IEEE, vol. 2, 2003, pp. 1824–1829.

[62] K. Harada, K. Nagata, T. Tsuji, *et al.*, "Probabilistic approach for object bin picking approximated by cylinders," in *2013 IEEE International Conference on Robotics and Automation*, IEEE, 2013, pp. 3742–3747.

[63] F. T. Pokorny, K. Hang, and D. Kragic, "Grasp moduli spaces.," in *Robotics: Science and Systems*, 2013.

[64] F. T. Pokorny, Y. Bekiroglu, and D. Kragic, "Grasp moduli spaces and spherical harmonics," in *2014 IEEE International Conference on Robotics and Automation (ICRA)*, IEEE, 2014, pp. 389–396.

[65] R. Detry, C. H. Ek, M. Madry, and D. Kragic, "Learning a dictionary of prototypical grasp-predicting parts from grasping experience," in *2013 IEEE International Conference on Robotics and Automation*, IEEE, 2013, pp. 601–608.

[66] J.-L. Blanco, "A tutorial on se (3) transformation parameterizations and on-manifold optimization," *University of Malaga, Tech. Rep*, vol. 3, p. 6, 2010.

[67] E. Johns, "Coarse-to-fine imitation learning: Robot manipulation from a single demonstration," in *2021 IEEE international conference on robotics and automation (ICRA)*, IEEE, 2021, pp. 4613–4619.

[68] M. Shridhar, L. Manuelli, and D. Fox, "Perceiver-actor: A multi-task transformer for robotic manipulation," in *Conference on Robot Learning*, PMLR, 2023, pp. 785–799.