



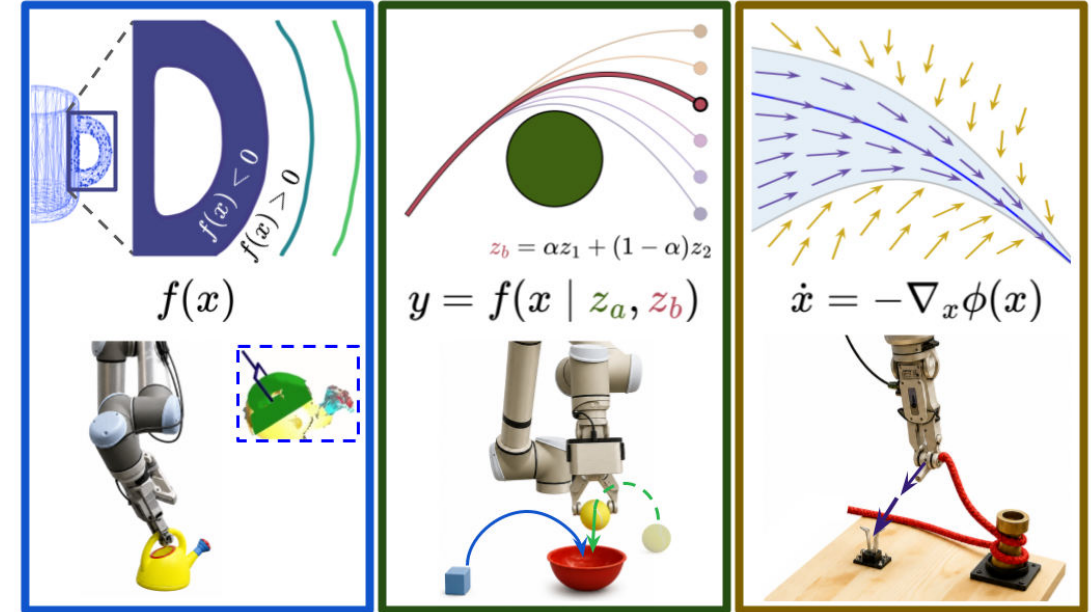
General-purpose robotic manipulation requires adaptability to diverse environments. To achieve this, robots need effective representations that allow them to understand their environment and perform correct actions. Machine learning provides a strong basis for designing and teaching robots such representations. While machine learning has driven remarkable progress in areas such as computer vision and natural language processing, its application to robotics is constrained by the high cost and complexity of collecting real-world training data. In robotics, collecting real-world data requires physical interaction with the environment and

often involves expert-designed task setups, human demonstrations, careful supervision, and extensive data curation, making the process expensive and difficult to scale. This creates a critical need for data-efficient learning approaches. This thesis presents a framework for data-efficient robotic manipulation through structured, task-aware representations. We investigate part-based representations for transferring grasp information, scene representations that capture essential structure for motion modeling, and reactive motion models that enable adaptive execution. Collectively, these contributions demonstrate that embedding task-relevant structure into robot representations provides a foundation for teaching robots diverse manipulation skills, enabling new capabilities, and substantially improving data efficiency.

AHMET ERCAN TEKDEN

Data-Efficient Representation Learning for Grasping and Manipulation

2026



Data-Efficient Representation Learning for Grasping and Manipulation

AHMET ERCAN TEKDEN

DEPARTMENT OF ELECTRICAL ENGINEERING

CHALMERS UNIVERSITY OF TECHNOLOGY

Gothenburg, Sweden 2026

www.chalmers.se

THESIS FOR THE DEGREE OF DOCTOR OF PHILOSOPHY

Data-Efficient Representation Learning for Grasping and Manipulation

AHMET ERCAN TEKDEN

Department of Electrical Engineering
CHALMERS UNIVERSITY OF TECHNOLOGY
Gothenburg, Sweden, 2026

Data-Efficient Representation Learning for Grasping and Manipulation

AHMET ERCAN TEKDEN
ISBN 978-91-8103-457-8

Acknowledgements, dedications, and similar personal statements in this thesis, reflect the author's own views.

© AHMET ERCAN TEKDEN 2026 except where otherwise stated.

Selected material from the author's licentiate thesis: Ahmet Ercan Tekden, "Data-Efficient Representation Learning for Grasping and Manipulation", Gothenburg, Sweden, Jan. 2024, is republished in this Ph.D. thesis.

Doktorsavhandlingar vid Chalmers tekniska högskola
Ny serie nr 5914
ISSN 0346-718X

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

Printed by Chalmers Digital Printing
Gothenburg, Sweden, August 2026

Data-Efficient Representation Learning for Grasping and Manipulation

AHMET ERCAN TEKDEN

Department of Electrical Engineering

Chalmers University of Technology

Abstract

Robotic manipulation requires models that can generalize across variations in objects, scenes, and task conditions. However, collecting large-scale datasets that capture such variations in real-world robotic settings remains costly and time-consuming, making data-efficient learning an important challenge. This thesis investigates how the choice of representation can influence data-efficient generalization in robotic grasping and manipulation. First, we introduce local shape descriptors that allow grasp poses to transfer across object categories by exploiting shared geometric structure. Second, we develop neural field models that represent scenes and motions as smooth functions of latent variables learned from demonstrations. This formulation organizes demonstrations in a structured latent space, enabling motion generation from a small number of demonstrations and generalization across scene variations through interpolation. Third, we propose a potential-function-based framework for reactive motion generation, where neural fields model smooth energy functions whose gradients generate well-behaved vector fields for control. A state dependent phase formulation further enables the representation of complex motion patterns while preserving reactivity. Together, these approaches demonstrate how representation choices can improve data efficiency and generalization in robotic grasping and manipulation.

Keywords: Grasping, Robot Manipulation, Robot Learning, Data-efficient Representation Learning, Learning from Demonstration, Neural Fields, Reactive Motion Generation.

To my family and friends.

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.

[C] **Ahmet Ercan Tekden**, Dimitrios Kanoulas, Aude Billard, Yasemin Bekiroglu, “Reactive Motion Generation via Phase-varying Neural Potential Functions”. *IEEE Robotics and Automation Letters (RA-L)*, vol. 11, no. 7, July 2026.

[D] **Ahmet Ercan Tekden**, Yasemin Bekiroglu, “Compositional Motion Generation from Demonstration with Object-Centric Neural Fields”. *IEEE Robotics and Automation Letters (RA-L)*, vol. 11, no. 9, September 2026.

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

[E] **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).

[F] 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.

[G] **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.

[H] **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.

[I] Yiting Chen, **Ahmet Ercan Tekden**, Miao Li, Dimitrios Kanoulas, Yasemin Bekiroglu, “Learning Pixel-Wise Suction Grasp Representation for Cluttered Environments”. *IEEE International Conference on Automation Science and Engineering (CASE)*, 2025, Los Angeles, USA.

[J] **Ahmet Ercan Tekden***, Francesco Gigante*, Shijie Zhou*, Dongxiang Li*, Yasemin Bekiroglu, “Reactive Whole-Body Control for Grasping Moving Objects with Mobile Manipulators”. *IEEE ICRA Workshop on Act to Sense to Act Better: Learning and Representations for Active Perception in Manipulation*, 2026, Vienna, Austria.

[K] **Ahmet Ercan Tekden**, Dimitrios Kanoulas, Aude Billard, Yasemin Bekiroglu, “Reactive Motion Generation via Phase-varying Neural Potential Functions”. *IEEE ICRA Workshop on Geometry in the Age of Data-Driven Robotics*, 2026, Vienna, Austria.

[L] **Ahmet Ercan Tekden**, Yasemin Bekiroglu, “Compositional Neural Field Movement Primitives”. *IEEE ICRA Workshop on Open Challenges for Rigorous Robot Perception*, 2026, Vienna, Austria.

Contents

Abstract	i
List of Papers	v
Acknowledgements	xiii
Acronyms	xiv
I Overview	1
1 Introduction	3
1.1 Grasping and Manipulation	4
1.2 Robot Learning	5
1.3 Data-Efficiency in Robotics	6
1.4 Coordinate-Based Representations	10
1.5 Learning From Demonstration	12
1.6 Research Scope	14
1.7 Contributions	14
2 Neural Fields	17
2.1 Formulation	18

2.2	Generative Modeling with Neural Fields	21
	Acquisition of latent vector z	22
	Conditioning of Neural Fields with latent vector z	23
2.3	Training Neural Fields and Test Time Optimization	24
2.4	Spectral Bias and Positional Encodings	25
	Coarse-to-fine Approximation	26
3	Grounding Grasp Generation in 3D Structure	29
3.1	Representing 3D Structure in Robotics	30
3.2	Descriptors for Action Grounding	31
3.3	Local Surface Modeling with Neural Fields	32
	SE(3) Group and Pose Alignment	34
	Local Surface Modeling	35
3.4	Extending Implicit Representations Beyond Grasping	35
4	Structured Scene and Motion Representations	39
4.1	Movement Primitives	41
4.2	Learning Scenes and Motions as Smooth Functions	43
	Smoothness and Coarse-to-Fine Modeling in Neural Fields . . .	43
	Lipschitz Regularization for Smooth Latent Representations . .	44
4.3	Shared Embeddings for Joint Modeling of Scenes and Motions .	47
	Convex Hull Constraints for Latent Inference	48
	Latent Relabeling for Compact Convex Support	50
4.4	Compositional Scene and Motion Generation	51
5	Reactive Motion Generation	55
5.1	Dynamical Systems for Motion Generation	58
5.2	Data-Driven Energy Landscape Design	62
	Decomposing Demonstrated Behavior	63
	Encoding Nominal Behaviour	63
	Modeling Support Regions of Demonstrations	64
5.3	Neural Potential Functions for Motion Generation	66
	Trajectory Synthesis for Nominal Trajectory and Admissible	
	Region Estimation	69
	Phase-driven Potential Functions	69
	Modeling Periodic Motions and 6D Tasks	72
5.4	Stability Analysis of PNPF	74

5.5	Smoothness of Energy Functions	76
6	Summary of included papers	79
6.1	Paper A	79
6.2	Paper B	80
6.3	Paper C	81
6.4	Paper D	81
7	Concluding Remarks and Future Work	83
	References	87
II	Papers	97
A	Grasp Transfer based on Self-Aligning Implicit Representations of Local Surfaces	A1
1	Introduction	A3
2	Related Work	A6
3	Method	A7
3.1	Preliminaries	A8
3.2	Pose Alignment of Shapes	A9
3.3	Local Surface Modeling	A10
3.4	Grasp Transfer	A11
3.5	Local Surface Prediction on Point clouds	A12
4	Experiments	A12
4.1	Shape Alignment	A13
4.2	Spatial Precision of Grasp Alignments	A15
4.3	Grasp Transfer	A16
4.4	Real Robot Experiments	A19
5	Conclusion	A20
	References	A20
B	Neural Field Movement Primitives for Joint Modelling of Scenes and Motions	B1
1	Introduction	B3
2	Related Work	B7

3	Method	B9
3.1	Problem Formulation	B9
3.2	Neural Fields	B9
3.3	Positional Encodings	B10
3.4	Scene-motion Embeddings	B10
3.5	Deformation and Template Fields	B11
3.6	Test Time Optimization	B12
4	Implementation Details	B13
5	Experiments	B14
5.1	Shape and Position Based Generalization	B15
5.2	Multi-valued Trajectory Generation	B18
5.3	Shape-based motion generation	B20
6	Conclusion	B20
	References	B21

C Reactive Motion Generation via Phase-varying Neural Potential Functions C1

1	Introduction	C3
2	Related work	C6
3	Motion generation with Potential functions	C8
3.1	Trajectory synthesis through generative modeling	C9
3.2	Modeling energy functions for task representation	C10
3.3	Phase-varying potential function design	C11
3.4	Stability of PNPF	C13
3.5	Modeling smooth potential functions with neural fields	C13
3.6	Modeling periodic motions	C14
3.7	Modeling 6D motions	C15
4	Simulation-based evaluations	C15
5	Real world experiments	C20
6	Conclusion	C22
	References	C22

D Compositional Motion Generation from Demonstration with Object-Centric Neural Fields D1

1	Introduction	D3
2	Related Work	D6
3	Problem Formulation	D7

4	Method	D8
4.1	Compositional Image Modeling	D9
4.2	Object-Centered Training of Experts	D10
4.3	Latent Relabeling	D12
4.4	Motion Generation	D13
4.5	Latent Conditioning with FiLM	D14
4.6	Inference	D15
5	Experiments	D16
5.1	Simulation Experiments	D17
5.2	Real-Robot Experiments	D19
6	Conclusion	D24
	References	D25

Acknowledgments

I want to express my heartfelt gratitude to my supervisor, Yasemin Bekiroglu, for her support throughout my doctoral studies. Her valuable advice, encouragement, and motivation were instrumental in my academic development. Thank you for trusting me and giving me the freedom to pursue my own research directions, even when they were unconventional. I am also grateful to my co-authors for their collaboration, insightful discussions, and valuable feedback.

My sincerest gratitude to Albert for his friendship, joyful lunch breaks, and fikas. Thanks for being there when I needed it most and supporting me. Your friendship made the academic journey all the more enjoyable. I would also like to thank my office mates Filip and Isac for their friendship and for enduring my yapping. I extend my appreciation to Remi, Stefan, Muhammad, Mattia, Godwin, Anand, Francesco, Atilla, Quentin, Rita, Gabriel, Maxi, Daniel, Lasse, Adam, Kilian, Erik, Sten, Amal, Ahmet, Rikard, Alvin, Yara, Nishant, Sabino, Lorenzo, Francesco, Martina, and Kristian for both academic and casual discussions and chats.

A heartfelt thank you to Alper, Safa, Alper, Enes, Furkan, Hacı, Ömer, İbrahim, Muhammet, Erkan, Oğuz, Ziya, Serkan, Melisa, Selçuk, Yunus, Kaan, Tuluhan, and Ali 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

LfD:	Learning from Demonstration
RGB:	Red Green Blue
2D:	Two Dimensional
3D:	Three Dimensional
6D:	Six Dimensional
NeRF:	Neural Radiance Fields
FiLM:	Feature-wise Linear Modulation
MLP:	Multilayer Perceptron
ReLU:	Rectified Linear Unit
SE (3):	Special Euclidean Group in Three Dimensions
SDF:	Signed Distance Function
INR:	Implicit Neural Representations
MP:	Movement Primitives
MoE:	Mixture-of-Experts
DS:	Dynamical Systems
PNPF:	Phase-varying Neural Potential Functions
ODE:	Ordinary Differential Equations
SEDS:	Stable Estimator of Dynamical Systems
LPV-DS:	Linear Parameter-Varying Dynamical Systems
P-QLF:	Parameterized Quadratic Lyapunov Function
DTW:	Dynamic Time Warping

Part I

Overview

CHAPTER 1

Introduction

Robots play an increasingly important role in manufacturing, logistics, health-care, agriculture, and many other domains. They can perform tasks with high precision, efficiency, and consistency. However, designing robot skills that operate robustly in diverse and changing environments remains challenging. Many robotic systems still require carefully structured setups and manually programmed motions tailored to specific scenarios. As a result, developing robotic systems that can adapt to environmental variations is an ongoing research challenge.

To operate in unstructured environments, such as homes or dynamic workspaces, robots must perceive their surroundings and adjust their actions accordingly. This requires acquiring effective representations of both the environment and the robot's actions, allowing the system to generalize its behavior under variation.

Machine learning provides a powerful framework for acquiring such representations from data and interaction. By learning from demonstrations or autonomous exploration, robots can model patterns in observations and actions rather than relying solely on manually specified rules. However, the ability of learning-based systems to generalize depends strongly on how inputs are

represented and processed. Certain representation choices may require large amounts of data, while others can promote more efficient generalization.

Recent advances in robot learning have achieved impressive performance through large-scale data collection and task-specific engineering. Nevertheless, in many real-world manipulation settings, collecting extensive datasets is costly and time-consuming. Achieving reliable performance, therefore, benefits from representations that generalize effectively from limited data. This thesis investigates how such data-efficient representations can be designed for robotic grasping and manipulation, with the aim of improving generalization in real-world scenarios.

1.1 Grasping and Manipulation

Robot manipulation refers to how a robot can interact with objects in its environment through grasping, moving, rotating, or placing them in order to achieve task-specific goals. Such tasks require coordinated perception, planning, and control, and often involve continuous physical interaction with the environment. As a result, robotic manipulation is highly sensitive to uncertainty in sensing, geometry, contact interactions, and dynamics.

A fundamental component of robotic manipulation is grasping. Before interacting with an object, a robot must establish stable and functional contact with it. This problem, commonly referred to as grasp synthesis, involves generating grasp poses that allow robots to hold and safely manipulate objects [1]. Grasp synthesis is particularly challenging in unstructured environments where object geometry, pose, and category may vary significantly across tasks. Robots also frequently operate in cluttered scenes, where objects may be partially occluded or densely arranged, introducing additional uncertainty in perception. Consequently, grasping methods must remain robust to such uncertainty while also generalizing to previously unseen objects.

Manipulation additionally requires generating motions that move the robot and manipulated objects toward desired task configurations while satisfying task constraints and remaining physically feasible. Such motions must adapt to variations in object placement, scene configuration, and task conditions. Traditional approaches often rely on explicit geometric models and carefully engineered planning pipelines, which can become difficult to scale in complex and dynamic environments.

Another important challenge is the reactive execution of such motions. During task execution, robots may encounter disturbances arising from sensing errors, object motion, or physical interaction with the environment. Manipulation policies must therefore continuously adapt online while preserving stable task progression.

These challenges are closely interconnected. Grasp selection influences feasible motions, environmental variations affect trajectory generation, and control policies must continuously adapt to changing conditions during execution. Developing representations that generalize across such variations while remaining data-efficient therefore remains a central challenge in robotic manipulation research.

1.2 Robot Learning

Robot learning refers to algorithms and techniques that enable robots to acquire skills from data, feedback, and interaction with their environment. It encompasses learning paradigms such as model learning [2], [3], imitation and apprenticeship learning [4], reinforcement learning [5], perception learning [6], and their combinations to teach robotic behavior efficiently and robustly [7]. Robot learning has become increasingly important because many robotic skills are difficult to program explicitly, robots must operate under uncertainty and partial information, and real-world environments vary significantly across tasks and trials [8].

These learning paradigms are commonly applied to robotic manipulation problems such as grasp synthesis, motion generation, and interaction modeling:

1. **Grasp Synthesis:** 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 involves producing trajectories or control signals that enable robots to complete tasks. Learning-based approaches, including reinforcement learning and supervised learning [7], provide flexible mechanisms for adapting motion to environmental variation.

3. **Model Learning:** Model learning aims to acquire representations and predictive models of the robot, its environment, and their interactions from data. These models can support perception, planning, and control by capturing scene structure, robot motion, and the effects of robot actions [10].

Robot learning enables flexible and adaptive behavior, but its practical deployment in real-world settings is often constrained by the amount of data required to achieve reliable performance. This challenge is particularly important in robotic manipulation, where collecting interaction data is costly and time-consuming. Consequently, improving generalization from limited demonstrations and interactions remains an important research direction.

1.3 Data-Efficiency in Robotics

Machine learning has achieved impressive results in domains such as natural language processing and computer vision, largely enabled by large-scale curated datasets. In robotics, collecting comparable datasets in real-world settings is often more challenging. Robotic data typically requires physical interaction or expert demonstrations, which are constrained by safety, cost, and hardware limitations. Although significant progress has been made through simulation, autonomous data collection, improved demonstration interfaces, including teleoperation systems [11] and handheld demonstration tools [12], as well as task-specific engineering, the scale and diversity of data available for many real-world manipulation tasks remain limited. This limitation arises from several practical factors:

- **Cost and safety of robotic hardware:** Real-world data collection requires expensive hardware and constant supervision. During exploration, robots can damage themselves or their environments, making unrestricted data collection unsafe. As a result, exploratory policies often require enforcing safety constraints, since robot actions can have irreversible consequences [13].
- **Complexity of real-world environments:** Robotic environments are complex and unstructured, and tasks can vary across different trials and settings. As a result, achieving robust task execution is difficult when

the collected data does not adequately cover the wide range of scenarios and edge cases encountered in real-world environments.

- **Human effort and scalability of data collection:** Data collection and annotation in robotics often require expert knowledge or supervision rather than simple labeling. This makes data collection costly, time-consuming, and difficult to scale in practice.
- **Variability across robotic setups:** Robotic data is often noisy and sensitive to differences in hardware, sensors, calibration, and environment. As a result, data collected with similar robots in different laboratories or settings may differ slightly, limiting how well learned models transfer across setups.

Given these practical constraints, improving generalization from limited data is an important consideration in robotic manipulation. One approach to addressing this challenge is to incorporate inductive biases into learning algorithms in the form of prior knowledge and structure [14]. In this thesis, we exploit multiple, complementary inductive biases to achieve data-efficient learning, including geometry-aware representations that capture spatial structure, smooth latent representations that encode continuity in scenes and motions, and compositional structure built on object-centric representations.

A first source of data efficiency explored in this thesis is the use of geometry-aware representations that exploit local spatial structure. Humans naturally decompose their environments into objects and parts and use these abstractions for physical reasoning [15], [16]. Similarly, many geometric features relevant for manipulation, such as graspable surface patches, recur across different object categories. For example, similar handle geometries appear across many household objects and provide consistent grasp affordances. This shared local geometry enables grasping strategies learned from one object to generalize to others. Representations that capture such recurring local structure can therefore transfer knowledge across objects without requiring task-specific retraining. In this thesis, we leverage this principle by modeling local surface geometry and learning shared spatial descriptors that enable the transfer of grasp poses across object categories, as presented in Paper A. An illustration of this approach is shown in Figure 1.1. By grounding learning in spatial structure shared across objects, this geometry-aware representation supports generalization from limited data.

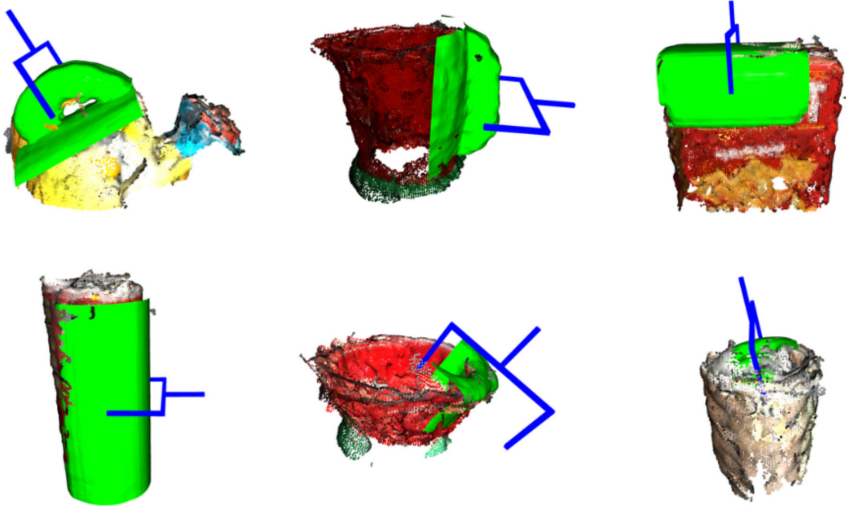


Figure 1.1: Local Surface Models for Grasp Transfer. Many objects share geometrically similar parts. We can utilize local surface models to identify such parts to transfer grasp experiences across object categories.

In addition to spatial structure, smoothness is a powerful inductive bias for learning representations in robotics. Many properties of object geometry, scenes, and robot motion vary smoothly with respect to spatial coordinates, time, and task parameters. Representations that explicitly capture this smooth structure can generalize effectively from limited data. In this thesis, we leverage smooth latent representations to model scenes and motions as continuous functions of shared embeddings, enabling data-efficient generalization and precise motion generation. This principle is demonstrated in the context of scene and motion representation using neural fields in Papers B and D. An illustration of smooth representations learned using neural fields is shown in Figure 1.2.

Beyond scene and motion representation, smoothness also plays a central role in reactive motion generation. We exploit this property in Paper C, where we learn potential functions whose gradients define vector fields for control. These potential functions are conditioned on a phase variable estimated from the system state, rather than on open-loop time. This formulation enables the

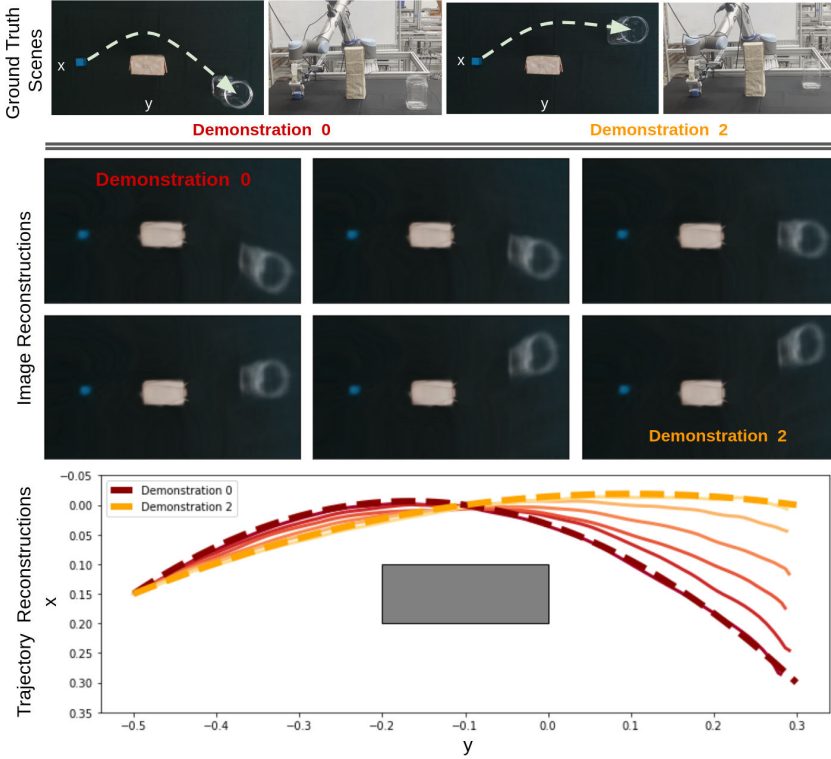


Figure 1.2: Joint modeling of scenes and motions. Each demonstration consists of a scene image and its corresponding robot trajectory. The model learns a smooth shared latent representation of both modalities, preserving the correspondence between scenes and motions. As a result, interpolation between demonstrations generates consistent scene–motion pairs, where each reconstructed trajectory matches its reconstructed scene.

representation of intersecting or repeating motion patterns while preserving reactivity to disturbances, as illustrated in Figure 1.3.

Beyond smoothness, data efficiency can also arise from exploiting the compositional structure of manipulation tasks. Compositional structure provides an additional source of data efficiency by factorizing complex tasks into object-centric components. We explore such object-centric compositional representa-

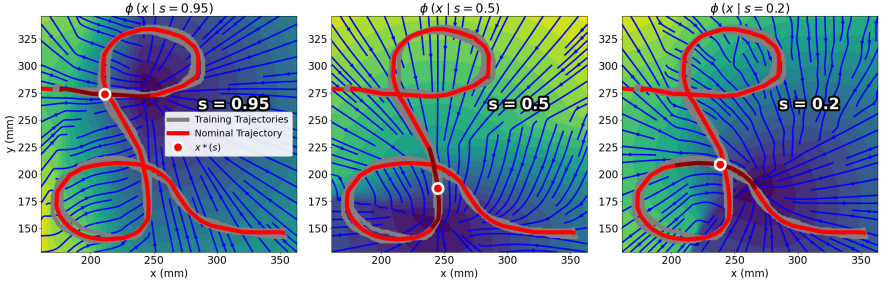


Figure 1.3: Smooth potential functions define continuous vector fields that guide robot motion. As the robot follows the vector field, it can continuously react to changes in its state and the environment. Conditioning the potential function on a state-dependent phase variable enables the representation of complex motion patterns, including intersecting trajectories, that cannot be represented by a single static vector field.

tions at both the scene and motion levels. Scenes are decomposed into object-specific components that can vary independently, while motion is generated by automatically associating different task phases with individual objects and learning object-conditioned variations in motion, as presented in Paper D. An illustration of this concept is shown in Figure 1.4, highlighting the object-centric decomposition of the scene and its relation to different phases of the task.

1.4 Coordinate-Based Representations

Coordinate-based representations model objects, scenes, or other signals as continuous functions that map spatial or spatiotemporal coordinates to signal values, rather than storing information on a fixed grid or as an explicit collection of samples. Depending on the output, these functions can represent geometry, appearance, occupancy, or other properties of the environment. For example, a shape can be represented by a scalar-valued function $f : \mathbb{R}^3 \rightarrow \mathbb{R}$ whose zero level set defines its surface, instead of an explicit mesh or point cloud. This continuous formulation supports queries at arbitrary resolution and naturally exposes smooth structure that is useful for optimization and generalization.

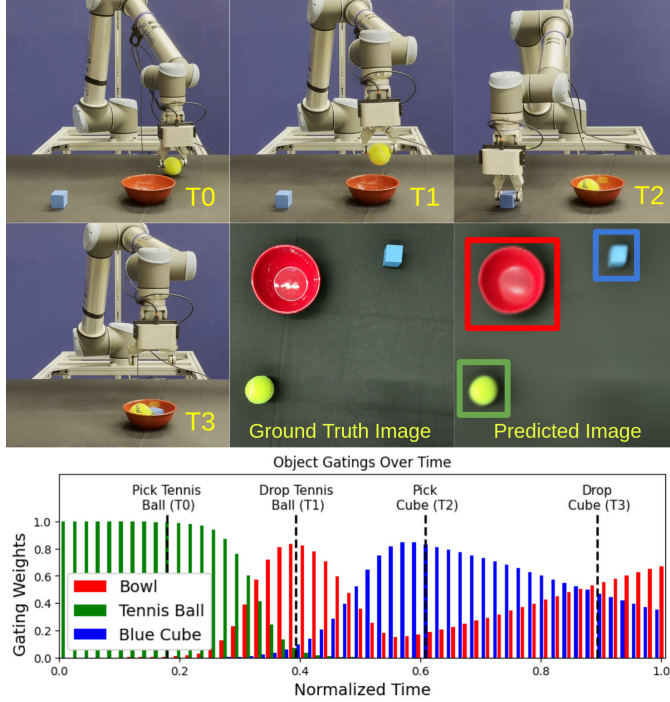


Figure 1.4: Compositional representations for scene and motion generation. The scene is decomposed into object-centric components, and motion is generated by associating different stages of the task with individual objects, enabling object-conditioned trajectory prediction.

Historically, coordinate-based representations were primarily developed for geometric modeling, where implicit functions such as signed distance fields, occupancy functions, and Gaussian process implicit surfaces [17] were used to represent object geometry through level sets. More recently, neural fields [18] have generalized this paradigm beyond geometry by representing a wide variety of continuous signals, including radiance, appearance, semantic features, and robot motions, using neural networks parameterized over coordinates.

Generative modeling provides a natural framework for learning such representations. Rather than directly predicting outputs from inputs, generative models capture the structure of data by modeling the underlying data man-

ifold [19]. This contrasts with discriminative modeling, which emphasizes learning decision boundaries between classes.

Generative modeling is particularly useful in robotics, where datasets often consist primarily of positive examples, such as successful demonstrations, and collecting samples representative of negative or unsupported cases is difficult or impractical. In this setting, generative models can learn the structure of the data manifold directly from positive observations, without requiring explicit contrastive supervision.

In this thesis, neural fields are used within a generative framework to model geometry, scenes, and motions as continuous functions conditioned on latent variables. Rather than learning direct input-output mappings, these models learn structured representations of the underlying data that support smooth interpolation and principled generalization across variations.

Neural fields are a particularly suitable choice for implementing such representations. Compared to earlier coordinate-based representations, neural fields scale naturally to high-dimensional inputs and large datasets while supporting flexible conditioning on latent variables, making them well-suited for modeling scenes and motions in robotic manipulation. As coordinate-based neural networks, they are differentiable with respect to their inputs, allowing their gradients to be used directly for optimization and control. They can be naturally conditioned on latent variables, supporting generative use across varying scenes, motions, and task configurations. Furthermore, their tendency to favor smooth functions over high-frequency ones provides a useful inductive bias for data-efficient learning: many properties relevant to manipulation, where object geometry, scene configurations, and motion trajectories vary smoothly in their natural coordinates, can be effectively captured by neural fields from limited demonstrations. The technical details of neural fields and their specific instantiations in this thesis are discussed in Chapter 2.

1.5 Learning From Demonstration

Learning from demonstration (LfD) provides a data-efficient framework for incorporating structured supervision into robot learning. Instead of relying solely on trial-and-error interactions, LfD leverages demonstrations or instructions to guide the learning process and constrain the solution space. By doing so, LfD can significantly reduce the amount of data required to acquire new

skills, particularly in complex manipulation tasks [20], [21]. In this thesis, we investigate the following topics within LfD:

Task-Conditioned Target Pose Learning: LfD can be used to specify key poses or waypoints, such as those required for grasping or placing objects in a particular manner, that are essential for solving a given task. These targets are often represented as task-conditioned local coordinate frames and are commonly learned using descriptors to enable generalization across objects within an object category [22], [23], [24]. In this thesis, we employ descriptors defined over local object surfaces or parts to enable the transfer of grasp poses beyond the demonstrated object category. This is possible because many objects share similar local parts, allowing grasp poses associated with these parts to generalize across categories.

Movement Primitives: Movement primitives are policy representations that encode continuous joint or state trajectories, typically parameterized by time or a phase variable [25], [26], [27], [28], [29]. By imposing structured parameterizations on motion, they enable robust generalization and data-efficient learning from a small number of demonstrations. For example, dynamical movement primitives [25] encode motions using dynamical systems with attractor behavior and probabilistic movement primitives [26] represent trajectories probabilistically through distributions over basis functions. In this thesis, we learn smooth scene and motion representations for modeling movement primitives. These representations implicitly capture a low-dimensional task parametrization by interpolating latent embeddings acquired during training, enabling accurate motion generation from a small number of demonstrations (Papers B and D).

Reactive Motion Generation LfD can also be used with state-based dynamical system formulations to generate vector fields that map the current robot state directly to velocity commands. Because control actions are computed as a function of the instantaneous state, such vector-field-based representations are inherently reactive and can adapt online to disturbances. In contrast to movement primitives, reactive motion generation methods avoid explicit temporal parameterization, as temporal variables are typically open-loop and reduce reactivity to perturbations. However, explicit temporal parameteri-

zations can be useful for representing complex motion patterns, particularly when different stages of the motion intersect in state space and therefore cannot be distinguished from the instantaneous state alone. In this thesis, we propose a novel state-dependent phase variable that resolves this ambiguity while preserving reactive behavior (Paper C).

1.6 Research Scope

The overarching goal of this thesis is to investigate how structured and continuous representations can improve data efficiency and generalization in robotic grasping and manipulation. In particular, this thesis studies how neural-field-based representations can be used to model geometry, scenes, motions, and reactive control policies from limited demonstrations.

To address this objective, the thesis investigates the following research questions:

- **RQ1:** How can local geometric structure be exploited for data-efficient grasp generation and transfer across objects?
- **RQ2:** How can manipulation trajectories be represented as structured and continuous functions that generalize across scene and object variations?
- **RQ3:** How can reactive motion policies be learned for complex motions involving repeated or intersecting states while preserving robustness to perturbations?

These research questions correspond to the three complementary stages of robotic manipulation explored throughout Chapters 3–5, and are summarized in Figure 1.5.

1.7 Contributions

The main contributions of this thesis, addressing the research questions introduced in the previous section, are summarized as follows:

- **(RQ1)** We introduce local surface-based representations and show how they can be used to transfer grasp poses across objects. By grounding

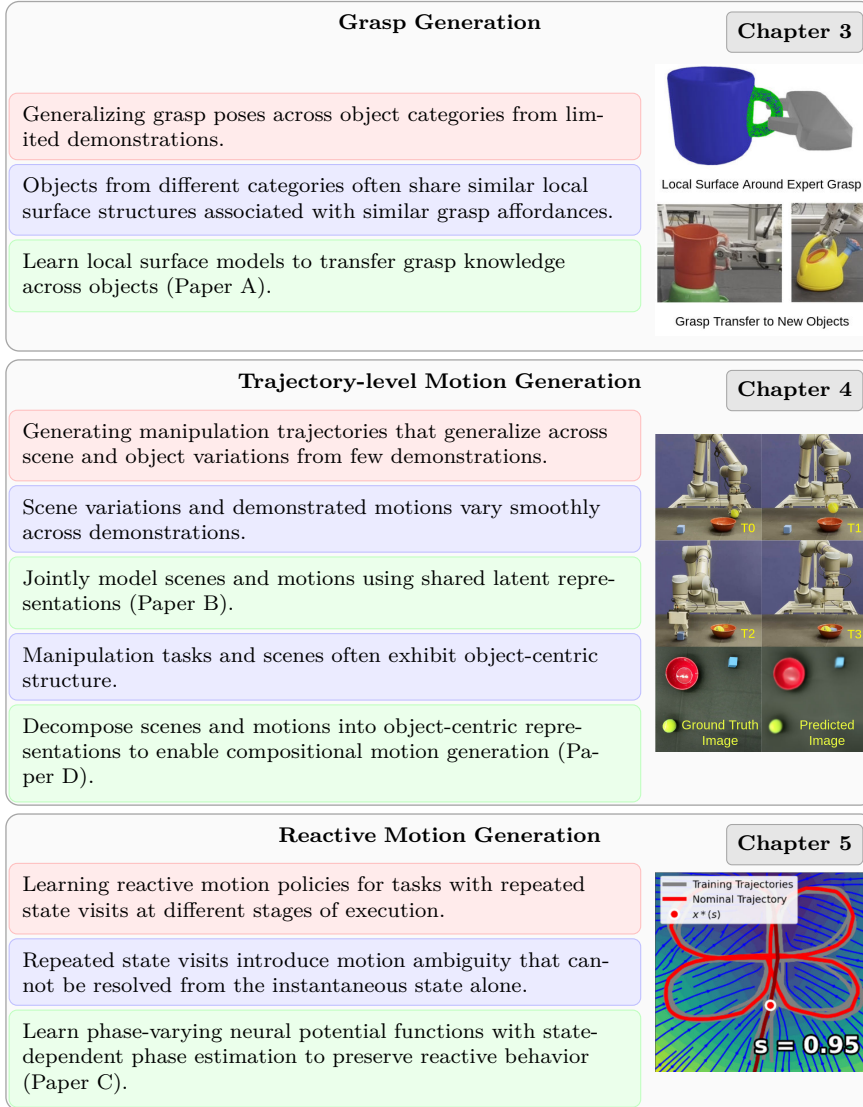


Figure 1.5: Overview of the three manipulation stages and corresponding research questions addressed in this thesis: grasp generation, trajectory-level motion generation, and reactive motion generation. For each stage, the figure summarizes the underlying research problem (red), the corresponding representation insight (blue), and the proposed data-efficient learning framework (green).

learning in local geometric structure shared across object categories, this formulation enables precise grasp synthesis from a single demonstration per category, without requiring category-level supervision (Chapter 3, Paper A).

- **(RQ2)** We propose neural field-based representations that jointly model scenes and motions using shared latent representations. By coupling scene observations and motion trajectories through a common embedding, this formulation enables data-efficient trajectory generation from visual input and generalization across scene variations from a small number of demonstrations, and is later extended to object-centric and compositional motion generation (Chapter 4, Papers B and D).
- **(RQ3)** We propose a reactive motion generation framework based on phase-varying neural potential functions. By combining potential functions with state-dependent phase estimation, this formulation models complex motions with repeated state visits while preserving reactivity to disturbances, without requiring large trajectory datasets (Chapter 5, Paper C).

The thesis is organized as follows:

- **Chapter 2: Neural Fields.**
Neural fields as continuous implicit representations: conditioning, training, and positional encodings.
- **Chapter 3: Grounding Grasp Generation in 3D Structure.**
Local surface-based representations for data-efficient grasp transfer across object categories.
- **Chapter 4: Structured Scene and Motion Representations.**
Neural field-based representations for data-efficient trajectory generation, extended to object-centric and compositional motion.
- **Chapter 5: Reactive Motion Generation.**
Phase-varying neural potential functions for reactive motion generation under perturbations and repeated state visits.
- **Chapter 6: Summaries of Included Papers.**
- **Chapter 7: Concluding Remarks and Future Work.**

CHAPTER 2

Neural Fields

Visual computing requires models capable of reconstructing, synthesizing, and manipulating scenes, objects, and shapes. A field is a concept from physics that describes quantities varying continuously over space and time, such as magnetic or gravitational fields [18]. Mathematically, a field is represented as a function that maps coordinates to values. This formulation can also be used to represent visual signals, for example by mapping pixel coordinates to RGB values in an image or 3D coordinates to occupancy values for shape representation. In practice, however, the analytical form of these functions is generally unknown. Instead, they are approximated either through handcrafted models or by learning their parameters from data.

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 [18]. These neural representations have been widely used for modeling scenes [30], 3D shapes [31], [32], and images [33]. In this thesis, we also demonstrate that neural fields provide flexible representations for modeling robot motion, scene–motion relationships, and potential functions for reactive control (Papers B, C, and D). Neural fields have the following benefits:

- They can represent signals with arbitrary input/output dimensions [18].
- They can be designed to be differentiable with respect to their inputs, enabling analytic gradients and objectives that involve higher-order derivatives [33]. This property is employed in our work for reactive motion generation (Paper C).
- The reconstruction process is often modality independent and, therefore, allows for representing signals from different modalities [34], (Paper B and D).
- They can automatically handle complex geometry processes such as filtering [35], deformation [36], and alignment [37], (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 chapter, we first present the formulation of neural fields and discuss representative application domains. Then, we describe how neural fields can be used as generative models. Finally, we detail common training procedures and discuss how spectral bias affects neural field optimization.

2.1 Formulation

Neural fields are neural networks that model continuous functions of spatial or temporal coordinates [18]. Given a field $\varphi : X \rightarrow Y$, neural fields learn a function that maps coordinates $x \in X$ to corresponding signal values $y \in Y$. These mappings can be defined either in the reconstruction domain or in the sensor domain. In the reconstruction domain, the neural field maps spatial coordinates in the scene to a physical quantity, such as occupancy or signed distance values. In the sensor domain, the neural field maps sensor coordinates, such as pixel locations, to the corresponding measurements, such as RGB values. Due to their flexibility, neural fields can model a wide range of scene and motion signals, as illustrated in Figure 2.1.

- **Scenes:** Scenes are often modeled with neural fields using radiance fields [30], commonly referred to as neural radiance fields (NeRF). NeRF learns a mapping $\mathbb{R}^5 \rightarrow \mathbb{R}^4$ between a 3D spatial location and a 2D

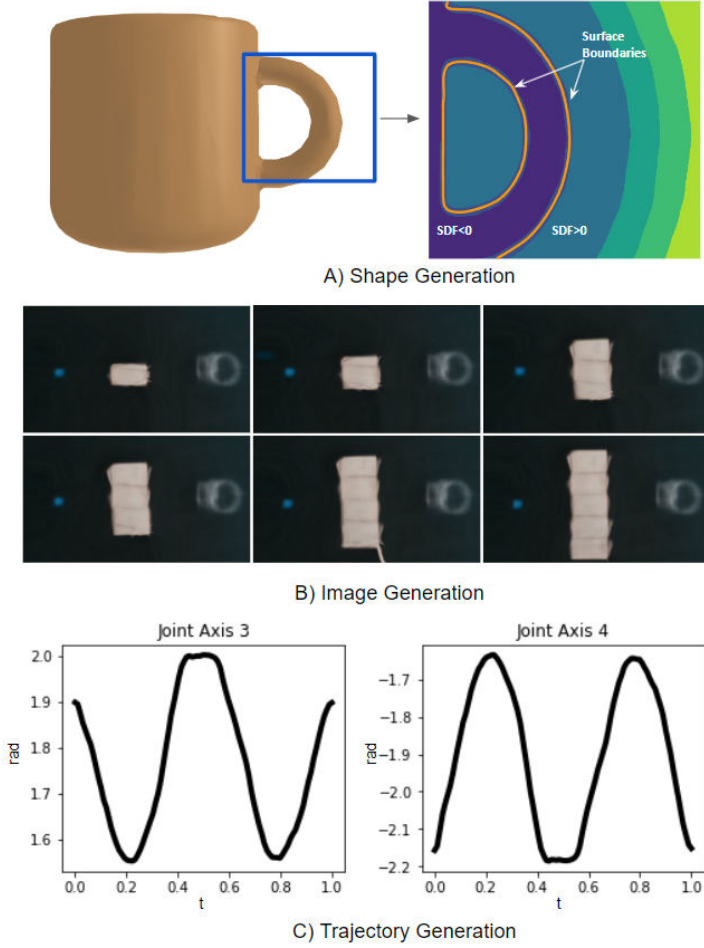


Figure 2.1: Examples of neural field representations. Neural fields map continuous coordinates to signals: (A) 3D shapes by mapping Cartesian coordinates to signed distance values, whose zero level set defines the object surface; (B) images by mapping pixel coordinates to RGB values; and (C) trajectories by mapping time to robot states, producing smooth motion trajectories.

viewing direction to RGB color values and volume density. Using differentiable volume rendering, these values are composited into images, enabling optimization using images captured from multiple viewpoints. Once trained, NeRFs allow rendering images from novel viewing directions. For more details, please refer to NeRF [30].

- **3D Shapes:** 3D shapes can be modeled through implicit surface functions [31], [32], [38], *i.e.*, $f(x) = 0$ for spatial coordinates x on the surface. Using marching cubes [39], these implicit representations can be converted into explicit meshes. Implicit surfaces can represent complex geometries, including shapes with sharp edges and holes. One widely used representation is the signed distance function (SDF), which maps each spatial coordinate $x \in X$ to the signed distance between x and the modeled surface. SDF values are negative inside the surface and positive outside. This representation is illustrated in Figure 2.1A. In this thesis, we employ SDF-based neural fields for modeling object geometry.
- **Images:** Images are common visual representations that can be easily acquired using cameras. Neural fields can model images as functions mapping 2D pixel coordinates to RGB values [33], $\mathbb{R}^2 \rightarrow \mathbb{R}^3$.
- **Motion Trajectories:** Motion trajectories represent sequences of robot states such as joint angles or end-effector poses. These trajectories can be modeled as functions mapping temporal coordinates to corresponding n -dimensional robot states, $\mathbb{R}^1 \rightarrow \mathbb{R}^n$.
- **Implicit Distance and Energy Fields:** In robotics, scalar fields are often used to represent task objectives or geometric constraints as distance, cost, or potential functions defined over the robot state or configuration space. Classical examples include artificial potential fields for obstacle avoidance [40], where gradients of the potential function generate control commands. Recent work extends this idea using neural implicit representations, such as grasp distance fields and robot distance functions, which model optimization landscapes over grasp poses [41] or robot configurations and enable gradient-based motion generation while avoiding collisions [42]. In this thesis, we similarly learn smooth potential functions from demonstrations using neural fields, whose gradients generate reactive robot motion (Paper C).

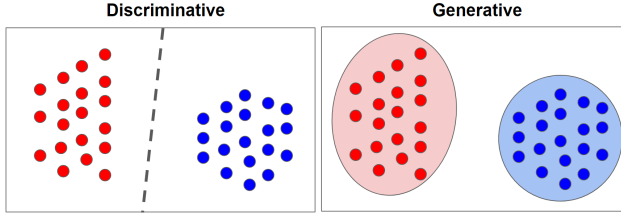


Figure 2.2: Generative and discriminative modeling of data. Generative methods aim to model the underlying data distribution, while discriminative models focus on learning decision boundaries between classes.

2.2 Generative Modeling with Neural Fields

Generative models aim to capture the underlying distribution of data such that new samples with a similar structure can be generated. In contrast to discriminative models, which learn decision boundaries between classes, generative models attempt to model the data distribution itself. This distinction is illustrated in Figure 2.2. Because they model the underlying data distribution, generative models can also handle missing or partial observations more naturally than purely discriminative approaches [43].

Formally, many generative models define a generator function g that maps latent variables z , sampled from a tractable distribution $\mathcal{Z}$, to samples Y ¹ from an intractable data distribution D , $Y = g(z)$, $z \sim \mathcal{Z}$. The latent variable z captures the underlying factors of variation present in the training data, allowing new samples to be generated by drawing different latent values [44].

Neural fields extend this formulation by explicitly modeling the spatial or temporal coordinates associated with each signal. Instead of generating an entire sample directly, neural fields learn a function that maps both a latent variable and coordinates to signal values, $Y = g(z, X)$, where X denotes spatial or temporal coordinates. This formulation enables continuous signal representations that can be queried at arbitrary resolution and allows transformations to be applied directly in the coordinate domain, such as deformation [36], [45] and alignment [37], (Paper A). In this section, we first describe how the latent variable $z \in \mathcal{Z}$ is obtained and then explain how it is used to condition neural fields.

¹In traditional generative models, the coordinates X are implicit in the data representation, e.g., pixel locations in images with fixed resolution.

Acquisition of latent vector z

In deep learning, latent vectors $z \in \mathcal{Z}$ are typically obtained through encoder neural networks. In such architectures, the latent representation is generated by passing an observation $\mathcal{O}$ through an encoder network E , yielding $z = E(\mathcal{O})$. Depending on the input modality, different architectures are commonly used. For example, PointNet-based networks [46], [47] are often used for point-cloud inputs, while convolutional networks [48], such as ResNet [49], are widely used for images. However, encoder-based approaches generally require large training datasets or extensive data augmentation to learn representations that generalize well.

Neural fields also employ an alternative approach for acquiring latent vectors through *auto-decoders* [31]. In this formulation, each observation is associated with a learnable latent vector, typically initialized from a normal distribution with small variance. These vectors are often referred to as embeddings² because they are optimized directly rather than produced as intermediate activations of an encoder network. During training, the latent vectors and network parameters are optimized jointly using gradient descent through backpropagation. At inference time, the network parameters remain fixed, and the latent vector corresponding to a new observation is obtained through iterative optimization that minimizes the reconstruction loss. In this sense, the optimization procedure implicitly performs the encoding step [18]. For a detailed comparison between auto-encoders and auto-decoders, readers are referred to the supplementary material of DeepSDF [31].

Auto-decoders are slower at inference than encoder-based approaches because they require iterative optimization. However, they are not restricted by the format of the observation used for reconstruction. In addition, our experiments suggest that auto-decoder-based representations can generalize more effectively in certain settings. When variations in the modeled scene are smooth, auto-decoder architectures can generalize within the convex hull of the training data by interpolating between learned embeddings. This behavior is demonstrated in Paper B. In contrast, encoder-based methods typically require more densely sampled training data to achieve similar generalization.

²The terms “latent vector” and “embedding” are often used interchangeably in deep learning.

Conditioning of Neural Fields with latent vector z

Given latent vectors z , neural fields can be conditioned on these representations in multiple ways. Since neural fields are neural networks that take spatial or temporal coordinates as input, conditioning mechanisms determine how the latent representation interacts with the coordinate-based generator network.

A common approach is *concatenation-based conditioning*, where the latent vector is concatenated directly with the input coordinates [31], [32]. While simple and widely used, this approach can make optimization more difficult when modeling complex geometric structures. In our preliminary experiments with shape representations, concatenation-based conditioning struggled to capture fine geometric details such as holes in object handles.

Another approach is to use *hypernetworks* [50], which generate the parameters of the neural field itself. In this formulation, a hypernetwork takes the latent vector as input and predicts the weights of the coordinate-based generator network. This idea has been applied to coordinate-based neural representations to generate families of implicit functions [33]. Hypernetworks provide a flexible conditioning mechanism and can be interpreted as generating a different neural field for each latent code. They also introduce additional modularity in neural network design and can improve parameter efficiency in conditional architectures [51].

An alternative is *feature modulation*, where the latent vector modifies intermediate activations of the network. A common realization of this idea is Feature-wise Linear Modulation (FiLM) [52], which applies learned affine transformations to hidden-layer activations conditioned on the latent vector. Similar modulation-based conditioning has been used to represent families of implicit neural representations through shared base networks and learned modulation vectors [53]. These mechanisms allow the latent representation to influence the generator network more directly while maintaining a fixed architecture.

In this thesis, we employ hypernetworks and FiLM-style modulation for conditioning neural fields in different settings across the included papers. These mechanisms allow the generator networks to remain compact while enabling flexible conditioning on latent representations.

2.3 Training Neural Fields and Test Time Optimization

Training neural fields in our work largely follows standard neural network optimization, with several important differences. First, when using an auto-decoder architecture, latent embeddings must be explicitly initialized and optimized during training. Second, neural fields represent signals as functions defined over coordinates. As a result, training typically relies on sampling coordinates from the observation rather than evaluating the entire signal at each iteration. This is particularly important for high-resolution observations; for example, representing a 256×256 image requires evaluating the neural field at 65,536 spatial locations. Accordingly, training proceeds as follows:

1. **Initializing embeddings:** Initialize a latent embedding $z_i \in \mathcal{Z}$ for each observation i in the training dataset D .
2. **Coordinate sampling:** Instead of evaluating the neural field at all coordinates of an observation, a subset of coordinates (X_i, Y_i) is sampled during each training iteration, where X_i denotes coordinates and Y_i the corresponding field values. For example, in shape modeling X_i corresponds to 3D spatial coordinates and Y_i to the signed distance values at those locations.

Sampling strategies can vary depending on the application. While uniform random sampling is commonly used, task-specific strategies such as importance sampling can improve training efficiency by focusing on informative regions of the signal. In our work, importance sampling strategies are employed in Paper D.

3. **Prediction:** For observation i , the latent embedding z_i is used to condition the generator network. Depending on the conditioning mechanism, this may involve predicting network parameters using a hypernetwork or modulating intermediate activations of the generator network. The generator network then evaluates the neural field at coordinates $x \in X_i$ to produce predictions $\hat{y} \in \hat{Y}_i$.
4. **Optimization:** The embedding z_i is optimized jointly with the parameters of the conditioning network and the generator network by minimiz-

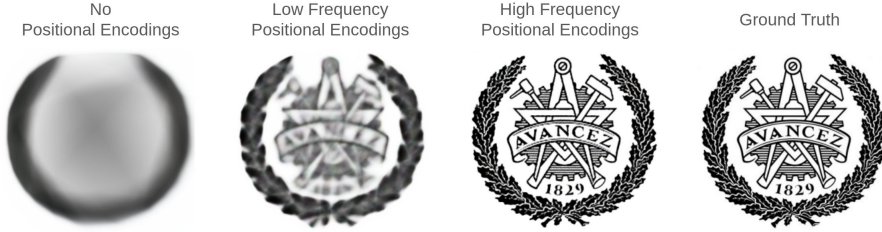


Figure 2.3: Illustration of spectral bias in image modeling. Neural fields are biased toward learning low-frequency components of signals when positional encodings are not used.

ing the reconstruction loss between predicted and observed field values $\hat{Y}_i$ and Y_i .

Unlike encoder-based architectures, auto-decoders do not directly infer latent embeddings from observations. Instead, the embedding for a new observation must be estimated through iterative optimization. This process is commonly referred to as *test-time optimization*. Formally, the latent embedding $\hat{z}$ is obtained by minimizing the reconstruction loss:

$$\hat{z} = \arg \min_z L_{\text{rec}}(g_z(X), Y), \quad (2.1)$$

where g_z denotes the generator function conditioned on latent embedding z . During inference, all network parameters remain fixed, and only the embedding $\hat{z}$ is optimized in order to reconstruct the observed data.

2.4 Spectral Bias and Positional Encodings

Neural networks exhibit an inductive bias toward learning simpler functions during training, a phenomenon known as *spectral bias* [54]. In particular, ReLU-based multilayer perceptrons (MLP) tend to learn low-frequency components of a signal before fitting high-frequency details [55]. This behavior can limit the ability of neural fields to represent signals containing fine spatial detail.

A common approach to address this limitation is to use positional encodings [56]. Positional encodings map spatial coordinates $x \in \mathbb{R}^n$ to higher-

dimensional inputs containing sinusoidal basis functions at multiple frequencies. This transformation enables neural networks to represent complex signals more easily. A visualization of this phenomenon is shown in Figure 2.3. In our work, we employ the positional encoding map

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

where $\Gamma_m(x) = [\cos(2^m \pi x), \sin(2^m \pi x)] \in \mathbb{R}^{2n}$ denotes the m^{th} frequency band.

While high-frequency positional encodings improve the ability of neural fields to capture fine details, low-frequency representations are often beneficial for modeling smooth structure and improving generalization. Smooth representations are also advantageous when gradients of the learned field are used for optimization or control. In this thesis, we exploit this property when learning potential functions for reactive motion generation, in which the gradients of the learned potential function generate velocity commands for the robot (Paper C).

To exploit the complementary benefits of low- and high-frequency representations, a coarse-to-fine approximation strategy can be used in which higher-frequency positional encodings are gradually introduced during training. This allows the model to first capture global structure and later refine finer details, improving generalization in scene and motion representations (Paper B) and enabling geometry processing operations such as pose alignment to emerge during training (Paper A).

Coarse-to-fine Approximation

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

In the coarse-to-fine approximation, low-frequency positional encodings are used first. Higher-frequency positional encodings are gradually introduced as training progresses. This scheduling strategy effectively controls the frequency spectrum that the network can represent during training. This strategy 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 2.4 where we train three different models on pose-perturbed images with label 6 from the MNIST dataset [57]. 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 [37]. Coarse-to-fine approximation is achieved by applying a smoothing mask to the positional encodings. We multiply each frequency 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:

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

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

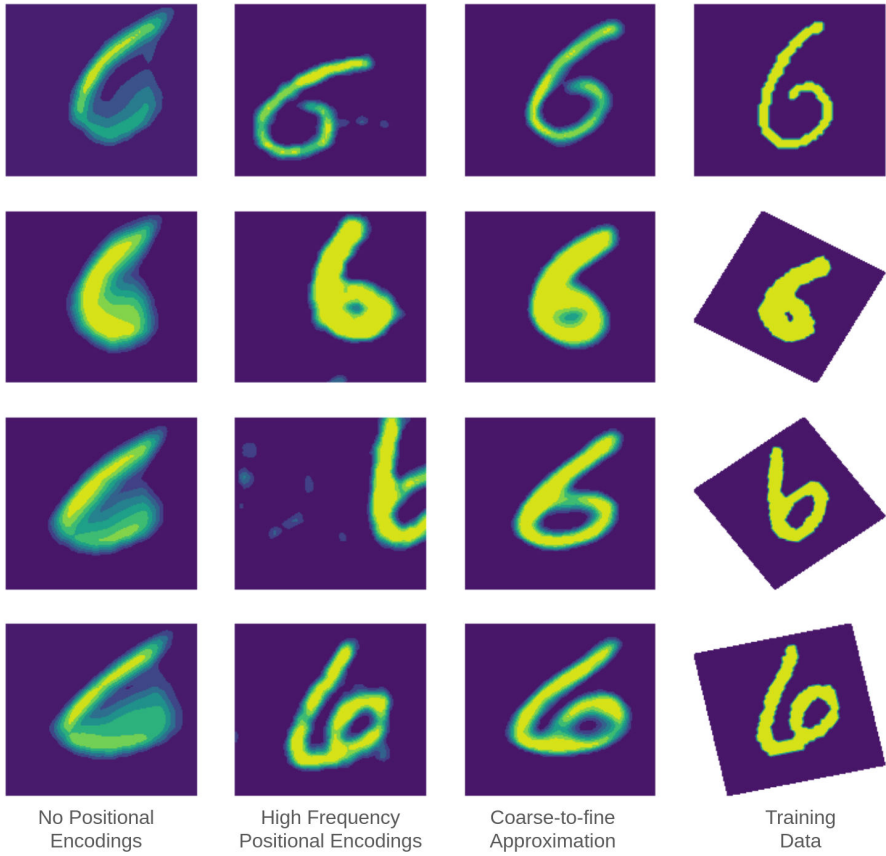


Figure 2.4: Illustration of effect of coarse-to-fine approximation for pose alignment. Coarse-to-fine approximation utilizes both low- and high-frequency positional encodings, enabling accurate pose alignment while preserving image reconstruction quality.

CHAPTER 3

Grounding Grasp Generation in 3D Structure

Robotic manipulation often requires interacting with objects whose shapes vary significantly across instances and environments. For this reason, representations that capture the 3D structure of objects provide a natural basis for reasoning about robot actions, as they expose geometric features such as surfaces, edges, and contact regions. Consequently, 3D shape models have been widely used in robotics for tasks including perception, scene reconstruction, grasp synthesis, and motion planning. By representing objects in terms of their geometric structure, robots can identify relevant regions of objects and relate actions such as grasp poses to those regions. Because many objects share similar local geometric structures, such representations can also enable the transfer of manipulation strategies across different object categories.

Representing objects through their 3D structure also provides an interpretable grounding for robot actions. When actions such as grasp poses are defined relative to geometric features of an object, the resulting representations are directly connected to the physical structure of the environment. This grounding makes it easier to reason about the behavior of learned models. For example, when a motion generation method relies on a learned shape representation, the correctness of the generated motion is closely tied to how well

the underlying geometry is modeled. If the reconstructed shape accurately captures the relevant object surfaces, the resulting motion or grasp pose is likely to be appropriate. Conversely, errors in the geometric representation often directly explain failures in the resulting actions. This connection between geometry and action provides a level of interpretability that is often absent in purely end-to-end learning approaches.

In many existing approaches, 3D information is primarily used as an input representation for robot learning. Robots perceive their environment through geometric observations such as point clouds or depth images, which are then used as inputs to models that predict grasps, actions, or policies. In this chapter, we instead explore how robot actions can be directly associated with predicted 3D geometry. By linking grasp poses and motions to geometric structures represented by learned shape models, actions become grounded in the underlying physical structure of objects rather than being inferred solely from sensory inputs.

3.1 Representing 3D Structure in Robotics

3D representations are widely used in grasp synthesis and robot manipulation because they provide geometric information about the objects in the environment. A common approach is to represent the scene using point-cloud observations obtained from depth sensors and directly predict grasp poses using neural networks [58], [59], [60], [61]. These methods can generate multiple grasp candidates directly from the point cloud of the observed scene. However, learning such grasp predictors typically relies on large-scale datasets that are often generated automatically in simulation, making it difficult to learn task-oriented grasp poses.

Beyond grasp prediction, several works have also learned manipulation policies directly from 3D input representations, including point-cloud and voxel-based representations. These approaches learn to map geometric observations of the scene to actions or control commands, enabling end-to-end learning of manipulation behaviors [62], [63]. While these methods demonstrate that manipulation policies can be learned directly from 3D input structure, they typically require large amounts of training data to capture the variability present in object geometry and task configurations.

To reduce this data requirement, some recent approaches introduce struc-

tured 3D representations, such as learned feature fields [64], [65], [66], [67]. These representations allow policies to reason about the scene more efficiently by predicting intermediate key poses that guide the execution of manipulation tasks. While this improves data efficiency, it typically constrains motion generation to keypose-based policies, limiting the flexibility of the resulting behaviors.

An alternative approach is to explicitly model object shapes rather than using geometric observations only as input representations. Shape representations are interpretable because their quality can be evaluated directly through geometric consistency with the observed scene. Various forms of shape models have been explored in robotics, including shape primitives [68], [69], smooth differentiable functions such as Grasp Moduli Spaces [70], [71], and prototype parts [72]. However, these models often provide limited generalization when observations deviate significantly from the predefined primitives or when the observations are partial. More recent approaches employ learned descriptors that map point-cloud observations to implicit shape representations [73]. While these methods improve flexibility, they often rely on category-level supervision and therefore struggle to generalize beyond the object categories present during training.

3.2 Descriptors for Action Grounding

Geometric descriptors provide a compact way to represent structures in the environment that can serve as anchors for robot actions. By associating actions such as grasp poses with identifiable geometric descriptors, manipulation strategies can be transferred across different objects and scenes.

In computer vision and robotics, such descriptors are commonly used to identify matching geometric structures across different observations of a scene, such as images or point clouds ¹. These descriptors enable identifying consistent structures across observations and are widely used in tasks such as tracking, scene reconstruction, visual servoing, grasp synthesis, and simultaneous localization and mapping. The success of these tasks relies heavily on the quality of the descriptors. Ideally, descriptors should be stable, reliable, distinctive, and accurate. When there is a camera pose or an illumination

¹Descriptors are typically used to establish *correspondences*, i.e., matches between features that represent the same physical structure across different observations.

change, the identified structures should remain stable and their estimated positions should not fluctuate, disappear, or change.

Descriptors are often defined at key points that are estimated from images or point clouds. Common ways to acquire such key points include classical computer vision techniques, such as SIFT [74] or SURF [75], as well as deep learning-based approaches such as dense visual descriptors [76], [77]. Classical methods do not require training; however, they typically generate a large number of key points, making it difficult to identify those relevant for manipulation. Furthermore, such methods cannot reliably estimate accurate key points in poorly textured scenes.

In comparison, learned visual descriptors map each pixel in an observation to a feature vector. Matching structures across observations can then be identified by minimizing the distance between feature vectors. While this improves robustness, descriptors defined at the pixel level introduce ambiguity in estimating the orientation of the corresponding 3D structure, which is crucial for accurate robot manipulation. Previously, this ambiguity has been resolved in two ways: by mapping each key point to multiple 3D points [24] or by mapping key points to 3D shapes [73]. 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 descriptors based on local surface models and using them to transfer grasp poses. By associating grasp poses with local surface descriptors, grasps can be transferred across object categories that share similar geometric structures. For example, a grasp defined on the handle or rim of a mug can be transferred to objects such as bags or bowls that exhibit similar surface geometry, as illustrated in Figure 3.1. Compared with previous descriptor-based approaches, the resulting grasp poses are more spatially accurate and transferable across object categories, as shown in Paper A.

3.3 Local Surface Modeling with Neural Fields

To represent local surface descriptors in a continuous and geometry-aware manner, we model local surfaces using neural fields. Neural fields allow geometric reasoning to be integrated directly into the optimization procedure, as explained in Chapter 2. This is particularly important because object parts typically do not appear in datasets in consistent canonical poses. To address

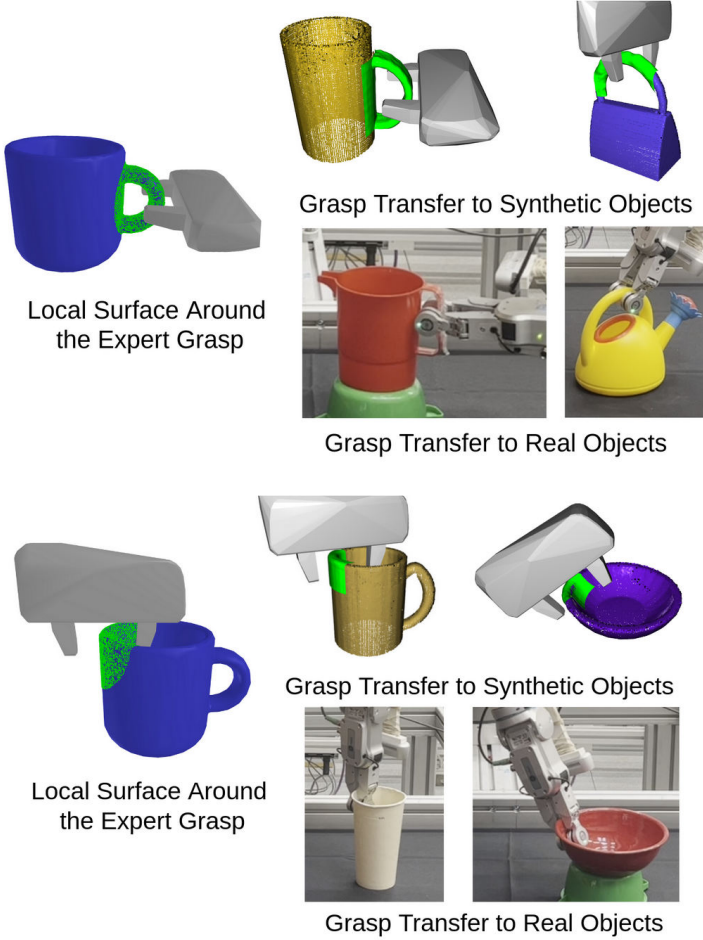


Figure 3.1: Local surface models as descriptors. We can use local surface models to identify local surfaces that are geometrically similar to the modeled ones and then use identified surfaces for grasp transfer.

this issue, we perform pose alignment during optimization using transformations from the Special Euclidean group in three dimensions ($\mathbf{SE}(3)$).

SE(3) Group and Pose Alignment

The Special Euclidean group $\mathbf{SE}(3)$ is a Lie group, and Lie groups combine the structure of a group with that of a differentiable manifold. This means that their elements are smoothly parameterized and each element has an inverse. Such a structure enables operations such as composition and differentiation, which are useful for optimization [78].

Each element of the $\mathbf{SE}(3)$ group represents a rigid body transformation. These transformations have an associated Lie algebra representation in $\mathfrak{se}(3)$, which can be represented as a 6D vector. Using the exponential map $\mathfrak{se}(3) \rightarrow \mathbf{SE}(3)$, Lie algebra elements can be mapped to valid rigid body transformations that lie on the $\mathbf{SE}(3)$ manifold [78].

Given $\beta = \langle \omega, t \rangle \in \mathfrak{se}(3)$, where $\omega \in \mathbb{R}^3$ and $t \in \mathbb{R}^3$ represent rotation and translation parameters, respectively, the corresponding transformation T can be obtained using the exponential map, which can be computed using the Rodrigues formula. The transformation matrix can be written as $T = \begin{bmatrix} R_{3 \times 3} & v_{3 \times 1} \end{bmatrix} \in \mathbf{SE}(3)$ where $v = Vt$ and R and V are defined as

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

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

where $[\omega]$ denotes the skew-symmetric matrix constructed from the vector ω using the hat operator (Eq. 7.4 in [78]), 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

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

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 consisting of object parts is required to learn local surface models. However, such datasets are not readily available. Instead, existing category-level shape datasets can be utilized. The key observation is that objects from the same category typically contain parts that share similar local geometric structures when the objects are placed in their canonical poses.

To extract such surfaces, we use a reference frame provided through a grasp demonstration on a representative instance from the dataset. Note that such reference frames could also be obtained using other waypoints in demonstrations, for example bottlenecks in trajectories [79]. The reference frame is used to extract local surface regions from objects in the dataset. While these surfaces are expected to share similar geometric structure, they may not initially be well aligned.

The extracted surfaces are then used to train neural field models that represent the corresponding local surface descriptors. These learned surface models can subsequently be used to identify geometrically similar surfaces on novel objects and to transfer grasp poses associated with those surfaces. Because the descriptors capture local geometric structure rather than object categories, the resulting grasp poses can generalize across objects that share similar surface geometry, as demonstrated in Paper A. The same framework can also be extended to other manipulation tasks beyond grasping by associating additional actions with the learned surface descriptors.

3.4 Extending Implicit Representations Beyond Grasping

The implicit surface representations introduced in this chapter are not limited to grasp synthesis. Because robot motions are closely tied to the geometry of the surrounding environment, implicit surface representations can also provide a structured foundation for motion generation. In particular, representing scenes through continuous geometric functions enables robot actions and trajectories to be conditioned directly on underlying scene structure rather than

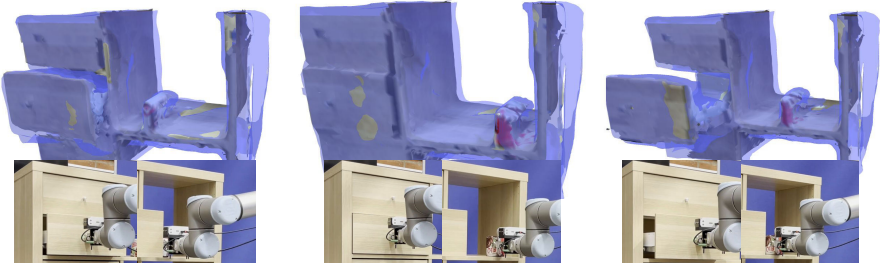


Figure 3.2: Example of motion generation grounded in implicit scene representations. Input meshes (top) and the corresponding surfaces reconstructed by the implicit model (shown in blue) are shown for different scene configurations. Snapshots of the resulting robot motion at key grasp poses are shown below the reconstructions. The reconstructed surfaces capture variations in scene geometry, such as drawer openings and box positions, enabling successful grasps of the drawer handle and the box.

solely on raw sensory observations.

Many recent approaches learn robot manipulation policies directly from 3D observations of the scene [62], [63]. These observations are typically represented either through rasterized structures such as voxel grids or image projections, or through unordered point clouds obtained from depth sensors. Rasterized representations allow the use of convolutional architectures but introduce a trade-off between spatial resolution and computational cost due to discretization. Point-cloud-based methods avoid discretization but require policies to reason over complex unordered sets of points, which can make the learning problem challenging and often necessitate large training datasets.

To reduce this complexity, several recent approaches introduce structured 3D representations such as learned feature fields [64], [65], [66], [67]. These methods improve data efficiency by decomposing manipulation into intermediate key poses predicted from the scene representation. However, this formulation typically constrains the resulting behaviors to waypoint-based policies, limiting the flexibility of the generated motions.

In contrast, implicit neural representations (INRs) model scene geometry as continuous functions rather than discretized grids. In this thesis, we employ decoder-only INRs that generate scene geometry conditioned on latent variables. Such representations remove the need for explicit rasterization and allow the scene to be queried at arbitrary spatial resolution. In practice, train-

ing and inference operate on sampled coordinates rather than fixed grids, enabling flexible and potentially sparse representations of geometry. A related perspective has been explored in recent work that treats INRs themselves as datapoints, enabling downstream tasks to be performed directly on these function representations [53]. However, these approaches typically rely on unconstrained latent embeddings to parameterize INRs. In contrast, some of the methods presented in this thesis restrict latent representations to interpolations within the convex hull of embeddings observed during training. This enables smooth interpolation between geometries observed in demonstrations while supporting data-efficient generalization to similar scene configurations.

Our formulation allows scene geometry and robot motion to be represented through shared parametric models. Variations in scene geometry are captured through embeddings learned from demonstrations, while motion generation is conditioned on these representations. Because these embeddings form a structured latent space of observed scenes, motion generation can generalize across variations in geometry through interpolation within this learned embedding space. An example of this formulation is illustrated in Figure 3.2, where implicit surface models reconstruct scene geometry while the corresponding robot motion is generated from the associated latent representation. This perspective extends the geometric grounding introduced earlier in this chapter beyond grasp synthesis toward trajectory generation conditioned on scene structure. Chapter 4 develops this idea further through joint scene-motion representations for data-efficient manipulation learning.

Summary: This chapter introduced a neural field framework for modeling local object surfaces to enable grasp transfer. By learning shared representations of grasp-relevant local geometry, the proposed approach improves data efficiency by transferring grasp knowledge across object categories. Further implementation details and experimental evaluations are provided in Paper A. However, successful manipulation requires reasoning beyond grasp synthesis, including representing the scene and generating task-appropriate motions. The next chapter therefore extends these ideas from modeling local object surfaces to joint scene and motion modeling, introducing structured latent representations for data-efficient manipulation learning.

CHAPTER 4

Structured Scene and Motion Representations

While the previous chapter demonstrates that local surface representations enable the transfer of grasp knowledge across similar geometric parts, it assumes that the object can be treated in isolation. In practice, manipulation is rarely task-agnostic or context-free. Executing manipulation skills such as lifting, rotating, or throwing requires motion trajectories that adapt to the surrounding environment and task-specific constraints. Consequently, the desired trajectory cannot be inferred from local object geometry alone. This motivates representations that capture variations in the surrounding scene and their relationship to robot motion.

Motion modeling is commonly achieved either through task parameters or through representations learned directly from observations. Task parameter-based approaches require sufficiently describing the scene using a set of pre-defined variables, which must either be provided by the user or estimated through additional perception pipelines, requiring further engineering effort. In contrast, observation-driven approaches learn mappings from sensory inputs to motion, often through encoder-based models. While these approaches remove the need for explicit parameterization, they typically require significantly more data to generalize across variations in the scene, as the underlying

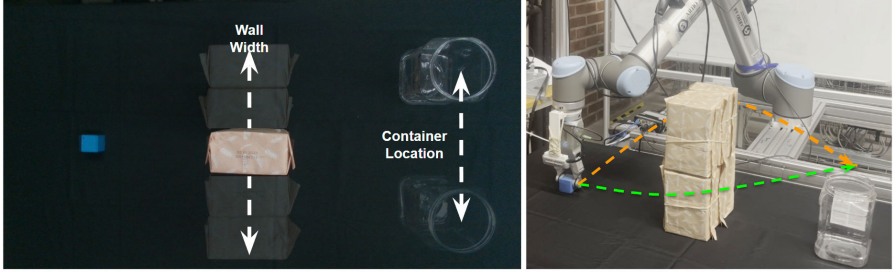


Figure 4.1: Wall avoidance task. The task parameters are on the left, and motion trajectories are shown on the right.

latent parameterization must be inferred from observations.

For example, consider the wall avoidance task shown in Figure 4.1. In this task, the robot must navigate around obstacles and place an object into a container, where the obstacle width and container location vary across different scenes. Such a task can be described using a set of task parameters, e.g., obstacle width and container position. However, these parameters must either be provided manually or estimated through additional perception modules. Alternatively, the task can be modeled directly from observations, but doing so requires collecting data that densely covers variations in the scene.

This highlights the need for representations that can capture variations in both scene geometry and motion without requiring explicit parameterization or large-scale data collection. To address this, we adopt a generative formulation in which scene observations and motion trajectories are modeled as functions of latent variables. These latent variables are not inferred through an encoder, but are instead associated with individual demonstrations and optimized jointly with the model parameters using an auto-decoder framework. In this setting, the model is primarily required to reconstruct observed samples, rather than generalize across the full variability of the data, which makes the learning problem more tractable in data-limited robotic settings. In contrast, learning an encoder requires mapping observations to latent representations in a way that generalizes across such variability. In robotic settings, this variability can arise even from factors such as sensor noise, while the available data is often limited and may not capture all sources of variation. As a result, such models can be challenging to train reliably.

Generative modeling alone does not ensure that scene observations and motion trajectories are represented consistently. In particular, a latent embedding may produce a scene and a motion that do not correspond to each other. To address this, we impose structure on the latent space so that a single latent embedding yields a consistent scene–motion pair. Furthermore, neural field-based representations naturally exhibit smoothness with respect to latent variables, allowing interpolations in the latent space to produce gradual and meaningful variations in both geometry and motion. By combining structured latent spaces with smooth generative models, we obtain representations that support both consistency and data-efficient generalization across variations in the environment.

While this formulation provides a unified representation of scenes and motions, representing all scene variations using a single latent embedding can become unnecessarily complex as the number of objects and their independent variations increase. Robotic environments naturally consist of multiple objects whose geometries and poses vary independently across demonstrations. Capturing all such variations within a single latent representation can entangle these factors, increasing the complexity of the learning problem. To address this, we later introduce object-centric compositional representations, in which individual objects are modeled separately while their representations are combined to describe complete scenes and their associated manipulation behaviors.

In the following sections, we first briefly introduce movement primitives. We then model scenes and motions as smooth functions of latent variables, introduce structured latent spaces for consistency, and finally extend this formulation to compositional representations for complex tasks.

4.1 Movement Primitives

Movement primitives (MPs) [25], [26], [27], [28], [29], [80], [81], [82] are widely used representations for modeling robot motion in Learning from Demonstration (LfD [20], [21], [83]). In these approaches, trajectories are encoded as smooth functions of time, allowing complex motions to be represented in a compact and continuous form. By explicitly incorporating temporal variables, MPs enable the generation of trajectories from a small number of demonstrations, making them particularly suitable for data-efficient learning. Movement

primitives can represent both point-to-point and periodic motions, for example through appropriate choices of temporal variables or basis functions [80].

At a conceptual level, MPs represent motion as parameterized functions of time that capture the structure of demonstrated behaviors. The term “primitives” highlights their role as modular building blocks that can be combined to form more complex behaviors. Extensions further allow these parameters to be conditioned on task variables, enabling adaptation to different situations.

Conditioning MPs on task variables requires identifying features that describe the scene, such as object positions, orientations, or goal configurations. In many formulations [26], [27], [29], this conditioning is achieved through target states, via-points, or coordinate frames derived from the scene, making the representation dependent on such geometric descriptors. While this enables adaptation to new situations, specifying these variables can be challenging. Alternatively, some approaches [28], [84] learn this mapping directly from high-dimensional observations using encoder-based models, but typically require larger datasets to generalize reliably. In both cases, this introduces a separation between perception and motion generation, where the resulting behavior depends on the accuracy and consistency of the extracted features.

The concept of movement primitives is inherently compositional, where complex behaviors can be constructed by combining simpler motion segments. In practice, however, this compositionality is often defined manually, for example, through concatenation or via-point specification.

In this thesis, we take a different perspective by modeling motion directly from scene observations within a generative framework. Rather than relying on predefined features or learned encoders to extract task variables, we represent scenes and motions using a shared latent parameterization. This allows variations in the scene and their corresponding robot motions to be represented jointly within a common latent space, preserving the correspondence between perception and action. In addition, instead of explicitly defining how motion segments should be combined, compositional structure emerges implicitly through a mixture-of-experts formulation with object-centric latent representations.

4.2 Learning Scenes and Motions as Smooth Functions

We consider LfD tasks in which both scene observations and motions can be described by underlying task parameters p . Accordingly, scenes and motions can be represented with neural field-based generator functions $f_p(X_{\text{scene}})$ and $g_p(X_{\text{motion}})$, respectively, where the generators are conditioned on these parameters. The variable $X_{\text{scene}} \in \mathbb{R}^m$ denotes the coordinates used to represent the scene, such as pixel locations in an image. For motion modeling, we consider two cases. In the first, trajectories are modeled directly as functions of time, where $X_{\text{motion}} \in \mathbb{R}^1$ corresponds to temporal coordinates. In the second, motions are modeled implicitly through the minimization of cost functions, where $X_{\text{motion}} \in \mathbb{R}^n$ corresponds to motion variables such as joint angles or end-effector positions. Further details on scene and motion modeling with neural fields are provided in Chapter 2.

In practice, such task parameters are not always available or easy to specify. Instead, we adopt a data-driven formulation in which these parameters are replaced by learned latent variables that capture variations across demonstrations. This yields generator functions $f_z(X_{\text{scene}})$ and $g_z(X_{\text{motion}})$ conditioned on shared latent representations. A key assumption in this formulation is that variations in scenes and motions are smooth with respect to these latent variables. By leveraging the smoothness properties of neural fields, this allows for consistent interpolation across variations in both geometry and motion.

Smoothness and Coarse-to-Fine Modeling in Neural Fields

Neural fields represent functions over continuous coordinates by directly taking spatial or temporal coordinates as input. Due to their spectral bias, they tend to model smooth variations with respect to these coordinates, capturing gradual changes in scenes and motions but struggling to represent fine details. In practice, this limitation is often addressed through positional encodings, which enable the representation of higher-frequency variations in the input space.

However, positional encodings introduce a trade-off. While they allow detailed reconstruction, higher-frequency variations are more difficult to learn and typically require more data to generalize effectively across variations in the underlying task parameters.

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 balance this trade-off, we separate the neural field into deformation and template components that use low- and high-frequency positional encodings, respectively. In this formulation, the template neural field models a canonical scene $T(x)$, while the deformation field models variations of this canonical scene as $D_z(x) = \Delta x$, yielding

$$f_z(x) = T(x + D_z(x)).$$

Only the deformation field is conditioned on the latent variable z , while the template field remains shared across demonstrations. Modeling the deformation field with low-frequency positional encodings enables smooth variations across latent variables and supports generalization with fewer observations, while the template field captures fine geometric details.

In practice, the frequency of the template field is increased progressively during training. This allows the model to first capture coarse structure before introducing higher-frequency details, improving training stability and generalization.

This is illustrated in Figure 4.2. The first row shows ground truth images, where images with red borders are not seen during training. Rows b) and c) show reconstructions without deformation fields using low- and high-frequency positional encodings, respectively. The former fails to capture fine details, while the latter does not generalize smoothly across scene variations. By combining deformation and template fields, as shown in row d), the model captures both smooth variations and fine details.

Lipschitz Regularization for Smooth Latent Representations

A key assumption in our setting is that variations in the underlying task configurations lead to smooth changes in the resulting outputs. More broadly, many robotic tasks can be reasonably approximated as smooth functions of task parameters, since small changes in object configurations or goals typically lead



Figure 4.2: Effect of positional encoding and the deformation-based formulation on image reconstruction. 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.

to gradual changes in the resulting motions. This property is important for generalization: if the underlying relationship were highly non-smooth, learning would require significantly denser data coverage. As a result, assuming smooth dependence on task parameters provides a useful inductive bias for data-efficient learning.

In our formulation, these variations are captured by learned latent variables. For the latent representation to be meaningful, distances in latent space should reflect differences in the corresponding task configurations. In particular, we seek a representation where distances are preserved up to a constant scaling factor,

$$\|z_1 - z_2\| \approx c_1 \|p_1 - p_2\|,$$

where p denotes the underlying (possibly unknown) task parameters. This implies that variations in latent space correspond consistently to variations in the underlying configurations.

In addition, the generator should vary smoothly with respect to latent variables:

$$\|f_{z_1} - f_{z_2}\| \leq c_2 \|z_1 - z_2\|.$$

This Lipschitz condition ensures that changes in latent space lead to controlled and predictable changes in the generated outputs. While neural network models often exhibit some degree of smoothness, the resulting behavior can still be less stable, particularly in data-limited settings. This is especially relevant for scene generation, where high-frequency details can lead to irregular variations.

The deformation-based formulation introduced earlier already promotes smooth behavior by representing variations as coordinate shifts of a canonical template. When parameterized with low-frequency representations, these deformations encourage gradual changes in the generated outputs with respect to latent variables. However, this implicit smoothness is not always sufficient to ensure stable behavior.

To improve stability, we incorporate Lipschitz regularization [85] to constrain the sensitivity of the scene generator with respect to latent variables. Encouraging Lipschitz smoothness reduces distortions in the mapping and promotes more consistent variations. As a result, the learned latent space tends to exhibit more uniform scaling behavior, where distances more reliably reflect differences in the underlying configurations. This can be interpreted as moving toward an approximately scaled isometric representation, even though such a structure is not enforced explicitly.

This behavior is illustrated in Figure 4.3, where we compare the learned latent representation with the corresponding task-parameter space for the cube stacking task. In this setting, the task consists of stacking three cubes, and the figure shows both the task setup and the associated latent geometry for one of the objects. Embeddings corresponding to training demonstrations are projected to two dimensions using principal component analysis, and an affine transformation is then estimated from these projected features to the normalized task parameters. Note that these task parameters are not provided to the model during training. The resulting close agreement between the two sets of points, with only small residual deviations, indicates that the learned latent space captures the geometric structure of the task space in a consistent

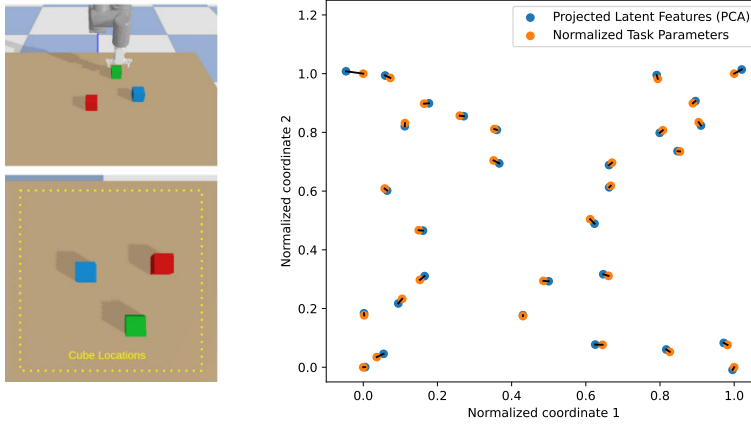


Figure 4.3: Cube stacking task and corresponding learned latent space representation. The left shows the task setup with three cubes, while the right visualizes the relationship between normalized task parameters and projected latent embeddings for one of the objects. The close alignment after affine transformation indicates that the learned latent space captures the geometric structure of the task space with only small residual deviations.

and well-organized manner. Such a structured representation is particularly beneficial in data-limited settings, enabling the model to learn a successful policy from only 30 demonstrations.

4.3 Shared Embeddings for Joint Modeling of Scenes and Motions

In the proposed framework, scene observations and motion trajectories are modeled through a shared latent representation. For many manipulation tasks, the underlying variation is intrinsically low-dimensional and can often be described by a small number of parameters, such as object positions or goal configurations. In contrast, the learned latent embeddings typically lie in a substantially higher-dimensional space. As a result, scene reconstruction does not necessarily identify a unique latent embedding, and multiple latent

values may correspond to similar observations. While this ambiguity is not problematic when modeling scenes or motions independently, since similar outputs may still be generated from different latent embeddings, it becomes important when the same latent variable is expected to produce a consistent scene-motion pair.

This consistency is essential for scene-conditioned motion generation. At inference time, a latent embedding is estimated from the observed scene and then used to generate the corresponding motion. For this process to be reliable, latent inference must recover a representation that is not only consistent with the scene but also remains compatible with the associated motion. In this thesis, we employ two mechanisms to ensure consistency in this process: constraining latent inference to the convex hull of the training embeddings and a latent relabeling process that aims to reduce the number of latent embeddings to obtain a more compact convex hull.

Convex Hull Constraints for Latent Inference

To improve consistency during inference, we restrict latent inference to the convex hull of embeddings observed during training. This is motivated by the smoothness assumptions underlying the proposed framework. Since each training embedding corresponds to a valid scene-motion pair, and the learned generators vary smoothly with respect to latent variables, latent interpolations between demonstrated embeddings are expected to preserve the corresponding scene-motion structure. In this sense, the convex hull of training embeddings provides a natural support for latent inference, as it captures interpolations among valid latent embeddings learned during training, while avoiding unconstrained extrapolation. This is also consistent with the smooth interpolation behavior observed in the learned latent space. This intuition is illustrated conceptually in Figure 4.4.

Note that latent samples outside the convex hull may still generate scenes and motions that appear plausible or similar to those seen during training. This is generally not problematic when scenes and motions are considered independently. However, in the joint modeling setting considered here, the key requirement is not only plausibility, but consistency: the generated scene and motion should correspond to the same underlying task configuration. This correspondence is more reliably maintained when latent inference is restricted to the convex hull of training embeddings.

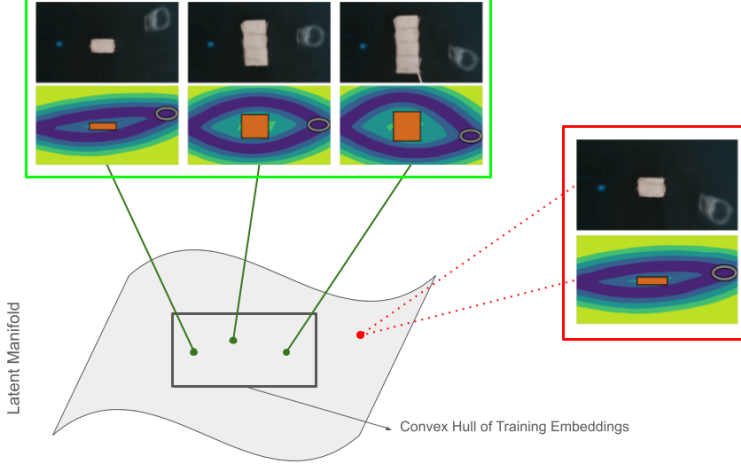


Figure 4.4: Conceptual illustration of the shared latent space used for joint scene-motion modeling. Interpolations within the convex hull of the training embeddings are expected to preserve consistent scene-motion correspondences. Outside the convex hull, latent representations may no longer generate corresponding scene-motion pairs, as illustrated by the red box, leading to less reliable inference.

Formally, let $\{z_i\}_{i=0}^{n-1}$ denote the latent embeddings associated with the n demonstrations in the training set. Instead of optimizing a latent variable directly during inference, we represent it as a convex combination of training embeddings:

$$z = \sum_{i=0}^{n-1} \alpha_i z_i, \quad (4.1)$$

subject to

$$\alpha_i \geq 0, \quad \sum_{i=0}^{n-1} \alpha_i = 1. \quad (4.2)$$

Latent inference is then performed by optimizing the convex coefficients α to minimize scene reconstruction error:

$$\arg \min_{\alpha} L_{\text{scene}}(f_z, X), \quad (4.3)$$

where z is defined through the convex combination above. This formulation constrains latent inference to remain within the latent support defined by the demonstrated data, improving the consistency of the inferred scene-motion pair.

Latent Relabeling for Compact Convex Support

While constraining latent inference to the convex hull of training embeddings improves consistency, not every demonstration needs to be included explicitly in the convex search. When the underlying task variation is smooth and low-dimensional, many demonstrations correspond to intermediate configurations and can instead be represented through interpolation between a smaller number of boundary cases.

Motivated by this observation, we introduce a latent relabeling procedure that replaces the original set of demonstration embeddings used in Eq. 4.1 with a more compact set of anchor embeddings. The objective is to identify a reduced latent support whose convex hull still spans the demonstrated scene and motion variations. In this way, demonstrations whose embeddings lie in the interior of this convex support need not be treated as independent variables during latent inference.

Formally, let $\{\tilde{z}_j\}_{j=0}^{m-1}$ denote a reduced set of latent anchors, with $m < n$, where n is the number of original demonstrations. Instead of treating each demonstration as an independent latent variable, we relabel the training set so that each original embedding z_i is represented as a convex combination of these anchor embeddings:

$$z_i = \sum_{j=0}^{m-1} \beta_{ij} \tilde{z}_j, \quad (4.4)$$

subject to

$$\beta_{ij} \geq 0, \quad \sum_{j=0}^{m-1} \beta_{ij} = 1. \quad (4.5)$$

The relabeling is defined through the learned generators by enforcing consistency in the reconstructed outputs. Specifically, the convex coefficients are chosen such that the scene generated by the relabeled embedding closely matches that of the original embedding. This replaces the original embedding set with a smaller convex support before training the motion generation

framework.

A key advantage of this formulation is that it substantially reduces the dimensionality of the latent inference problem. If latent search is performed directly over n demonstration embeddings, the convex coefficients define an $(n - 1)$ -dimensional search space. After relabeling, this becomes an $(m - 1)$ -dimensional problem. For example, representing a task with 30 demonstrations yields a 29-dimensional convex search space, whereas reducing this to 4 anchor embeddings yields a 3-dimensional search problem. Although this reduced dimensionality may still exceed the intrinsic degrees of freedom of the task, it is significantly more compact and better aligned with the underlying task variation. This also simplifies latent inference and yields a more interpretable latent structure, where anchor embeddings correspond more directly to extremal or representative task configurations.

4.4 Compositional Scene and Motion Generation

Many manipulation tasks exhibit an inherently compositional structure. At the motion level, task execution typically unfolds through a sequence of object-centered interactions, such as approaching, grasping, transporting, and placing. At the scene level, the environment itself is naturally composed of multiple objects. These two forms of structure are closely related: different phases of a manipulation task depend on different subsets of objects and their configurations. This suggests that scene understanding and motion generation need not be treated as independent, monolithic problems, but can instead be modeled using structured, reusable components. Such a formulation is particularly advantageous in data-limited settings, as it enables reuse of object-level representations and motion patterns across different task instances, improving generalization and interpretability.

In this thesis, we introduce a mixture-of-experts (MoE)-based framework for compositional scene and motion generation over spatial and temporal coordinates, respectively. At the scene level, observations are decomposed into object-centric components, each represented by a latent-conditioned neural field. At the motion level, trajectories are generated through a temporal mixture-of-experts model that combines object-conditioned motion contributions over time. Figure 4.5 illustrates this decomposition, showing how object-centric scene representations relate to different phases of a manipulation task.

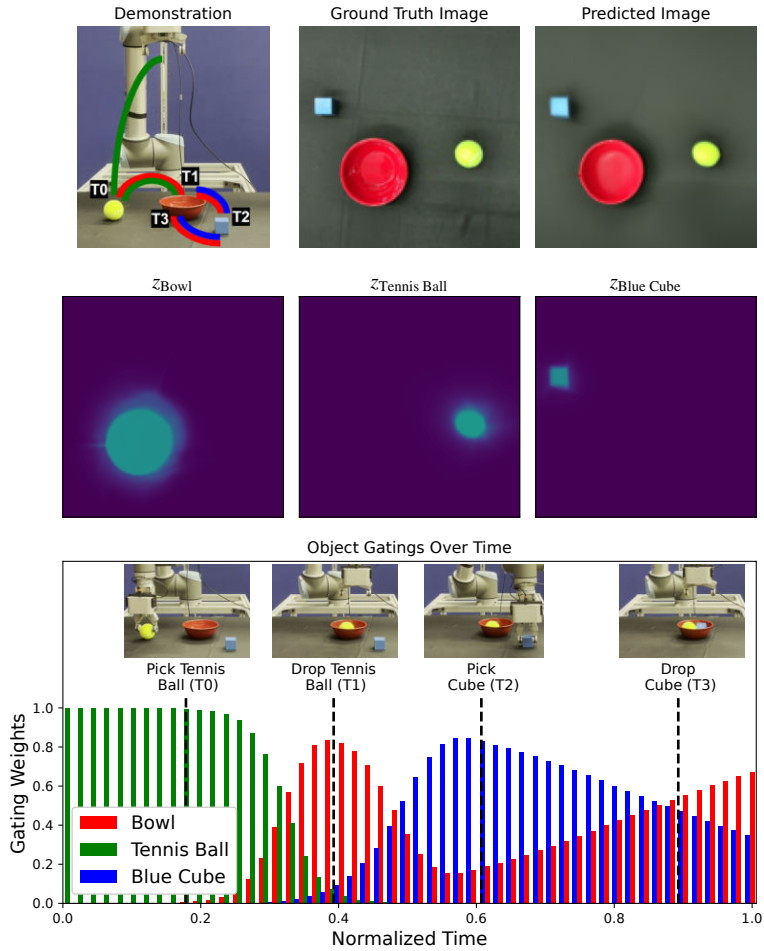


Figure 4.5: Illustration of compositional scene and motion generation. The scene is represented through latent-conditioned object-centric neural fields. These latent representations also condition motion generation, where object-specific contributions are combined through a time-varying mechanism, allowing different objects to influence different phases of the task.

We model each scene using an object-centric spatial mixture-of-experts formulation:

$$\hat{I}(x \mid \{z_i\}_{i=1}^N) = \sum_{i=1}^N M_i(x_i) f_i(x_i) + M_{\text{bg}}(x) f_{\text{bg}}(x), \quad (4.6)$$

where $x_i = x - \Delta_{z_i}(x)$ denotes coordinates deformed according to the latent embedding z_i . Here, f_i represents the appearance of object i , Δ_{z_i} models its deformation, and M_i denotes a spatial mixing mask that determines the contribution of each object across the image. The masks are computed via a spatial softmax over object and background logits,

$$M_i(x_i) = \frac{\exp(l_i(x_i))}{\exp(l_{\text{bg}}) + \sum_{j=1}^N \exp(l_j(x_j))}, \quad (4.7)$$

which enforces a normalized spatial mixture over foreground objects and background. The background is modeled separately through f_{bg} and M_{bg} . Object-centric structure is encouraged during training through mask-guided sampling, which biases each object model toward its corresponding spatial region while reducing interference from other objects. Importantly, these masks need not be pixel-accurate; it suffices that they roughly localize the target object while avoiding other objects, thereby providing a weak yet effective spatial prior for object grounding.

At the motion level, we model trajectory generation through a temporal mixture-of-experts formulation over object latent vectors $\{z_i\}_{i=1}^N$. Different phases of a manipulation task generally depend on different subsets of objects, and this is captured through a time-dependent pooled conditioning signal:

$$c(t) = \sum_{i=1}^N w_i(t) c_i(\hat{z}_i), \quad w_i(t) = \frac{\exp(s_i(t))}{\sum_{j=1}^N \exp(s_j(t))} \quad (4.8)$$

where $s_i(t)$ are learnable relevance scores, $c_i(\hat{z}_i)$ denotes object-specific conditioning variables and $w_i(t)$ defines the contribution of each object latent at time t . In Paper D, the conditioning vectors are realized through feature-wise linear modulation (FiLM) [52], although the formulation is compatible with other latent conditioning mechanisms. The trajectory generator then predicts

a distribution over states $q(t) \in \mathbb{R}^d$ as

$$q(t) \sim \mathcal{N}(\mu(t), \text{diag}(\sigma^2(t))) . \quad (4.9)$$

with

$$[\mu(t), \sigma(t)] = f(t \mid c(t)), \quad (4.10)$$

where $\sigma(t)$ is constrained to be positive. The model is trained by minimizing the negative log-likelihood of demonstrated trajectories under the predicted time-varying Gaussian distribution. To encourage robust compositional behavior, latent dropout is applied during the early stages of training. In addition, the probabilistic trajectory formulation reduces spurious dependencies between object representations by allowing uncertainty arising from less informative latent variables to be absorbed into the predicted variance. Further details are provided in Paper D.

Summary: This chapter introduces a unified framework for joint scene and motion modeling based on structured latent representations. By leveraging smooth latent spaces, convex latent constraints, and object-centric compositionality, the proposed approach enables consistent and data-efficient learning of both scene geometry and manipulation behavior. Further implementation details and experimental evaluations of these ideas can be found in Paper B and D. However, the methods presented in this chapter do not explicitly account for online reactivity to disturbances. Building on the use of neural fields as continuous function representations, the next chapter introduces a reactive motion generation framework based on neural potential functions.

CHAPTER 5

Reactive Motion Generation

In the previous chapters, we address how to generate robot motion trajectories that generalize to scene variations. While direct trajectory generation produces accurate motions from demonstrations, the generated trajectories are typically fixed once inferred and therefore provide limited reactivity to perturbations or changes in the environment. In many robotic settings, however, the robot must adapt continuously to perturbations, uncertainty, or changing task conditions. This motivates representing motion not as a trajectory alone, but as a continuously evaluable structure that guides robot actions online.

A common approach is to represent motion using dynamical system (DS)-based policies, which map the current system state directly to control actions [86]. Because the control action is computed from the instantaneous system state, these policies naturally adapt to perturbations and changes in the environment, making them particularly suitable for real-world robotic applications. Rather than following a time-dependent trajectory, where reactivity is inherently limited, the system evolves according to its current state and can therefore respond dynamically during execution. However, this formulation introduces a fundamental ambiguity when a task requires different velocity commands at the same state depending on the stage of execution.

Since the state alone does not carry sufficient information to resolve this ambiguity, the resulting policy becomes ill-defined at such states.

Resolving this ambiguity requires augmenting the state representation with additional information. One possibility is to extend the state with velocity, allowing motion segments that pass through the same position in different directions to be distinguished. However, in practice, such formulations can be sensitive to velocity disturbances and may become ambiguous when similar motion segments precede intersections. Alternatively, a phase variable augments the state with a measure of task progression, directly resolving the ambiguity by associating each state with its stage of execution. Phase variables have been widely used in movement primitives for this purpose [25]. However, traditional phase variables rely on open-loop temporal schedules that advance independently of the system state. This limits reactivity, since when perturbations displace the robot, the phase continues to progress regardless. Instead, a state-dependent phase variable preserves the disambiguating role of phase while remaining coupled to the robot’s actual execution, allowing ambiguity caused by state revisits to be resolved without sacrificing the reactive nature of DS-based policies.

Beyond the choice of state representation, DS-based policies can also be understood through their connection to Lyapunov theory [87]. In this setting, stability is characterized with respect to a scalar function $V(q)$ over the state space, often referred to as a Lyapunov function. By constructing or learning dynamics that decrease this function over time, the system is driven toward desired configurations while maintaining stability. The goal of dynamical system-based policies is therefore to induce motions that satisfy an appropriate Lyapunov property for the task. This perspective highlights an important property: the Lyapunov function defines a geometric landscape over the state space that characterizes the desired system behavior. Within this landscape, desired system behavior corresponds to trajectories that continuously decrease the value of the Lyapunov function over time. This viewpoint naturally leads to a constructive interpretation of reactive control, in which the gradient of the Lyapunov function induces the governing vector field. In this sense, Lyapunov functions can be interpreted as potential functions, where motion is driven by descending the corresponding energy landscape. Potential functions can be composed of multiple complementary energy functions that encode different aspects of the task and together characterize the desired sys-

tem behavior. Their gradients jointly determine the resulting motion, while individual energy functions can additionally provide meaningful task variables, such as measures of task progression or safety. This viewpoint also underlies artificial potential field methods in robotics, where attractive and repulsive potentials are used to generate reactive behaviors such as goal reaching and obstacle avoidance [40].

This naturally motivates the following question: can such potential functions be learned directly from demonstrations? This chapter introduces Phase-varying Neural Potential Functions (PNPF). Rather than manually designing the energy landscape, PNPF learns it from a small number of demonstrations. The learned energy landscape consists of two complementary components: a nominal energy that encodes task progression along a reference trajectory and a safety energy that characterizes the region of state space supported by demonstrations. Their gradients jointly induce reactive motions while preserving the geometric structure of the demonstrated behavior. Furthermore, the nominal energy provides a state-dependent measure of task progression, from which we derive the phase variable without relying on an external clock. This preserves the reactive properties of the dynamical system while resolving the ambiguity introduced by state revisits.

Implicit representations provide a natural modeling framework for such potential functions. In robotics, continuous distance-like functions have long been used to encode geometric structure, for example in obstacle representations, collision checking, and motion optimization. More recently, neural implicit models have extended this idea by enabling expressive continuous representations of geometry, occupancy, and distance directly from data. This makes them particularly well suited for modeling potential landscapes: they define scalar functions continuously over the state space, remain differentiable with respect to their inputs, and therefore provide gradients that can be directly used for motion generation. In this way, neural fields offer a principled connection between modern implicit function learning and classical potential-based control. In this chapter, we present a method for constructing and learning such neural potential functions from demonstrations, and for generating reactive motion through their gradients.

5.1 Dynamical Systems for Motion Generation

Dynamical systems (DS)-based LfD methods represent robot motion as ordinary differential equations (ODEs) of the form $\dot{x} = f(x)$ where f is learned from human demonstrations. The goal is to learn a stable autonomous DS that enables reactive motion generation, where stable means trajectories converge asymptotically to a goal state from arbitrary initial conditions, and autonomous means f has no explicit time dependence. Since velocity commands are computed directly from the current state, the robot can adapt online to perturbations without replanning.

An important aspect of reactive motion generation is stability, and many foundational approaches focus on explicitly enforcing it. The Stable Estimator of Dynamical Systems (SEDS) [88] represents the underlying dynamics as a mixture of locally linear systems and guarantees global asymptotic stability through Lyapunov-based constraints. In particular, SEDS assumes a quadratic Lyapunov function centered at the goal and constrains each local linear system such that it decreases this shared energy function. When these local dynamics are combined through positive, state-dependent weights, the resulting vector field also decreases the same Lyapunov function everywhere, thereby ensuring global stability.

Later, Linear Parameter-Varying Dynamical Systems (LPV-DS) [89] extend this framework by relaxing the coupling between the mixture representation and the local dynamics, and by introducing a parameterized quadratic Lyapunov function (P-QLF) for stability. In contrast to SEDS, this approach separates the estimation of the mixture model from the learning of the dynamical system. Specifically, it first learns a mixture model to partition the state space into local regions, and then estimates the parameters of the local linear dynamics under P-QLF stability constraints, resulting in a more flexible yet still globally stable dynamical system. Despite this increased flexibility, both SEDS and LPV-DS can be interpreted as constructing dynamical systems that descend along a quadratic potential centered at the attractor, with LPV-DS effectively reshaping this potential through a parameterized metric.

A related line of work formulates motion generation as the gradient of a learned potential function constructed from demonstrations [90]. In this approach, a potential function is designed to encode the desired behavior by combining an attractive component that pulls the system toward a demonstrated trajectory with a tangential component that drives motion along it.

This results in a structured prior that captures both convergence to the demonstration and progression along it. However, since the potential is constructed from a single trajectory, the resulting dynamics tend to be stiff, strongly constraining trajectories to remain close to the demonstrated path and limiting flexibility in the surrounding state space.

To address this limitation, Rana *et al.* [91] extend this formulation by introducing a learned Riemannian metric that warps the geometry of the state space, effectively modulating the influence of the potential and enabling greater flexibility in reproducing complex behaviors. While this reduces the stiffness of trajectory-centric dynamics, the learned metric is applied globally, including in regions not supported by demonstrations. In particular, the metric can attenuate directions that steer the system back toward the trajectory, weakening the corrective behavior induced by the potential design. As a result, the original attraction toward the trajectory may no longer be consistently preserved, leading to a trade-off between flexibility and preservation of the intended behavior. This limitation is illustrated in Figure 5.1.

A different class of approaches enforces stability by constructing dynamical systems through diffeomorphic transformations [92]. In these methods, complex motions are obtained by mapping a simple, stable dynamical system defined in a latent space through a smooth, invertible transformation, such that the stability properties are preserved under the change of coordinates. As a result, stability is encoded directly through the structure of the model, since the learned system is topologically equivalent to the base stable system.

However, this formulation imposes strong architectural constraints on the learned model. In particular, the transformation must be globally invertible and smooth, which typically requires specialized parameterizations such as normalizing flows or structured diffeomorphic mappings. To relax these constraints, recent approaches such as CONDOR [93] instead learn neural network-based dynamical systems without enforcing such structural restrictions, while encouraging stable behavior through additional training objectives. In particular, CONDOR employs contrastive learning to promote similarity between the learned dynamics and a simple stable dynamical system, allowing greater flexibility in the learned representation.

Despite their differences in formulation, many existing approaches exhibit a common behavioral pattern in the generated trajectories. As illustrated in Figure 5.1 [91], trajectories often evolve approximately parallel to the demon-

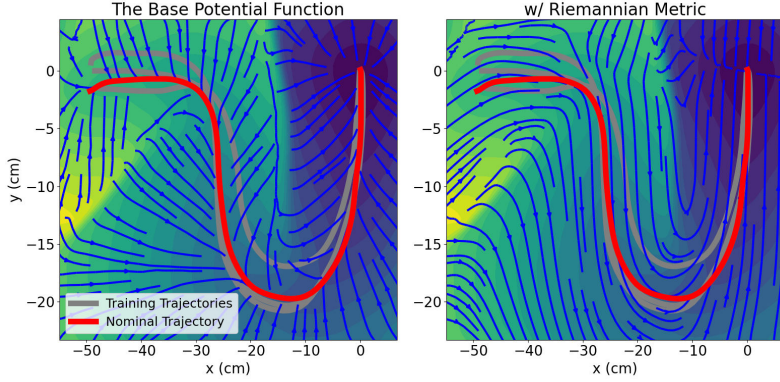


Figure 5.1: Comparison of the base potential function and the Riemannian metric formulation of [91]. The left figure shows the vector field generated by the base potential function, while the right figure shows the corresponding vector field after applying the learned Riemannian metric. The generated vector field exhibits trajectories that remain largely parallel to the nominal trajectory over most of the motion, with convergence occurring only near the end via a perpendicular correction. This behavior indicates that the attraction toward the trajectory is weakened in large regions of the state space, consistent with the global warping induced by the learned metric.

strations before converging toward the attractor near the goal. Similar behavior can also be observed in LPV-DS and neural approaches such as CONDOR as illustrated in Fig 5.2. This behavior arises from an inherent stability design based on single-attractor dynamical systems. In particular, smooth or convex modulations of the vector field tend to preserve the underlying convergence structure, making early or strong convergence toward the demonstrated trajectories difficult to achieve unless such behavior is explicitly encoded in the demonstrations.

From a dynamical systems perspective, this behavior can be related to the notion of contraction [94], which emphasizes the convergence of trajectories toward each other rather than toward a single equilibrium. In contrast to single-attractor formulations, contraction-based systems aim to reduce the distance between trajectories throughout the state space, promoting convergence toward a set of desired motions rather than only a goal. While we

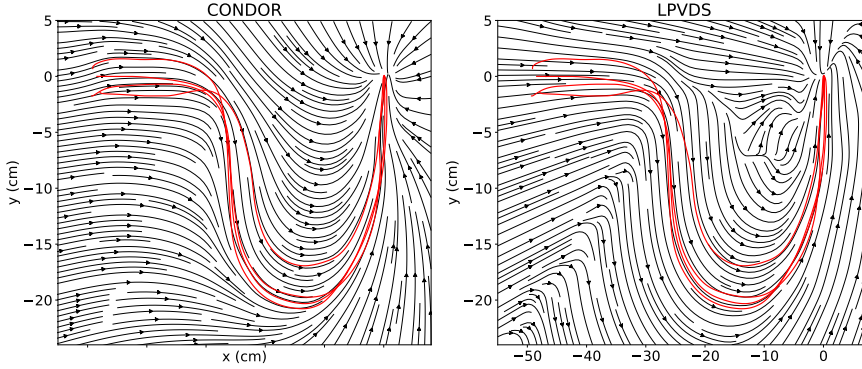


Figure 5.2: Vector fields generated by CONDOR and LPV-DS. In both approaches, trajectories tend to evolve approximately parallel to the demonstrations before converging toward the attractor near the goal.

do not explicitly adopt a contraction-theoretic framework in this thesis, this perspective highlights a desirable property that is not always captured by existing approaches. Motivated by this observation, we consider a formulation that separates different objectives within the dynamical system, where task progression and attraction toward demonstrated regions are represented as distinct components; this idea forms the basis of our approach and will be detailed in the remainder of this chapter.

Finally, a common limitation of these reactive motion generation methods arises from their reliance on first-order state-based representations. Since these models assign a unique velocity to each state, they cannot represent intersecting trajectories in which the same state may require different onward motions depending on the progression of the task. To address this, second-order formulations augment the state with velocity to disambiguate trajectories; however, this introduces sensitivity and may remain ambiguous when similar local motions precede intersections. Alternatively, temporal variables can be introduced to resolve such ambiguities, but this makes the system non-autonomous and significantly reduces reactivity. These limitations highlight the need for representations that can capture such behaviors while preserving the reactive properties of dynamical systems. In this chapter, we explore such a formulation by introducing a state-dependent phase variable that remains

coupled to the system state, enabling more expressive yet reactive motion generation.

5.2 Data-Driven Energy Landscape Design

A useful perspective for motion generation is to view behavior as evolution on an energy landscape defined over the system state. In this view, trajectories are generated by descending a scalar potential function through gradient flow or iterative optimization. Similar principles appear throughout control and decision-making, where cost or value functions encode the desirability of states and actions. From this perspective, the geometry of the function determines the resulting motion: minima correspond to goals or stable states, while local slopes define directions of motion.

In classical control and motion planning, such landscapes are typically designed manually using geometric intuition, analytic models, or task-specific heuristics. More recent approaches have extended this principle through energy-based [95] or distance-function formulations [41], [42], [96] that encode task objectives in a structured way. In contrast, we seek to construct these landscapes directly from demonstrations of expert behavior. Rather than hand-specifying the objective, learning from demonstration allows the underlying geometry of the task to be inferred from data, yielding motion generation policies that reflect demonstrated strategies and preferences.

A natural representation for such landscapes is provided by implicit functions, which define continuous scalar fields over the state space and can be evaluated at arbitrary query points. As discussed in Chapter 2, neural fields provide a flexible framework for learning such functions directly from data. Their differentiability enables analytic gradients for optimization and control, while their coordinate-based structure supports compact representations that generalize across states and task variations. In this chapter, we build on that foundation and consider neural implicit energy functions learned from demonstrations, whose gradients induce reactive motions while preserving the geometric advantages of continuous field representations.

Decomposing Demonstrated Behavior

For the modeled dynamical systems, we consider two complementary behaviors. First, the system should encode nominal task progression: a preference for states that advance execution toward successful completion of the task. This component determines how progress is achieved through a scalar field whose values decrease as the task evolves, thereby inducing motion toward the goal. In this sense, the landscape plays a role analogous to a value or cost-to-go function, assigning lower energy to states that lie further along successful execution.

Second, the system should characterize the region of state space in which motion remains admissible, capturing the variability and robustness exhibited across demonstrations. Real demonstrations rarely collapse to a single trajectory; instead, they typically span a surrounding admissible region containing valid variations arising from execution noise, compliance, redundancy, and stylistic differences across demonstrations. A useful motion landscape should therefore distinguish deviations that remain within this admissible region from those that leave the support of the data.

We realize these two behaviors through a modular design. This allows the corresponding energy landscapes to be modeled separately, so that progression and admissibility can each be specified in a structured manner while remaining compatible when combined into a single motion-generation objective. This separation also helps mitigate the parallel-motion tendency discussed in the previous section.

Encoding Nominal Behaviour

We first consider how to model nominal task behavior through an energy landscape. For this purpose, assume access to an optimal reference trajectory that represents ideal task execution. In optimal control, the value function associated with the task would increase monotonically as this optimal trajectory is followed, with its gradient aligned with the direction of the optimal trajectory. This naturally motivates an equivalent potential-field formulation, in which the same ordering is encoded through decreasing energy values. A convenient construction is to define the scalar energy through geodesic distance along the reference path, so that states associated with earlier phases of execution are assigned higher energy, while states closer to task completion receive lower

energy.

This ordering can then be extended to nearby off-trajectory states by projecting them onto the closest point of the reference path and assigning the corresponding energy value. Geometrically, this is equivalent to extending the optimal trajectory into a family of locally parallel level sets that inherit the same progress coordinate. As a result, neighboring states share a consistent notion of task advancement, while the gradients of the induced potential field remain approximately aligned with the optimal trajectory.

In practice, however, access to an optimal trajectory is rarely available. Instead, we estimate a nominal trajectory from the available demonstrations and use it as a surrogate reference for constructing the nominal energy function.

Modeling Support Regions of Demonstrations

In addition to nominal progression, demonstrations also reveal a region of state space within which task execution remains valid. This region is shaped by task constraints, system dynamics, and natural variability in execution. Estimating such support is useful for motion generation, as it provides a data-driven notion of where demonstrated behavior is feasible and where nearby perturbations may remain admissible. However, directly recovering these regions is challenging. Exact concave hull constructions become increasingly intractable in higher dimensions, while practical non-convex boundary estimators are often sensitive to sparsity, noise, and hyperparameter choices.

If a sufficiently large number of demonstrations were available, this admissible region could be densely covered by observed trajectories, making its geometry directly visible from the data. In this ideal setting, the support of demonstrated behavior could be inferred directly from the aggregate occupancy of the trajectories themselves. A corresponding distance function could then be estimated by sampling query states throughout the state space and labeling them according to whether they lie inside or outside the observed support using a proximity threshold. States whose distance to the densely sampled admissible region is sufficiently small may be treated as interior points, while more distant states are treated as exterior samples. Distances to the nearest demonstrated states or nearest exterior samples would then provide a practical approximation of the signed distance function (SDF) describing the boundary of this admissible region.

In practice, however, demonstrations are typically limited in number. As

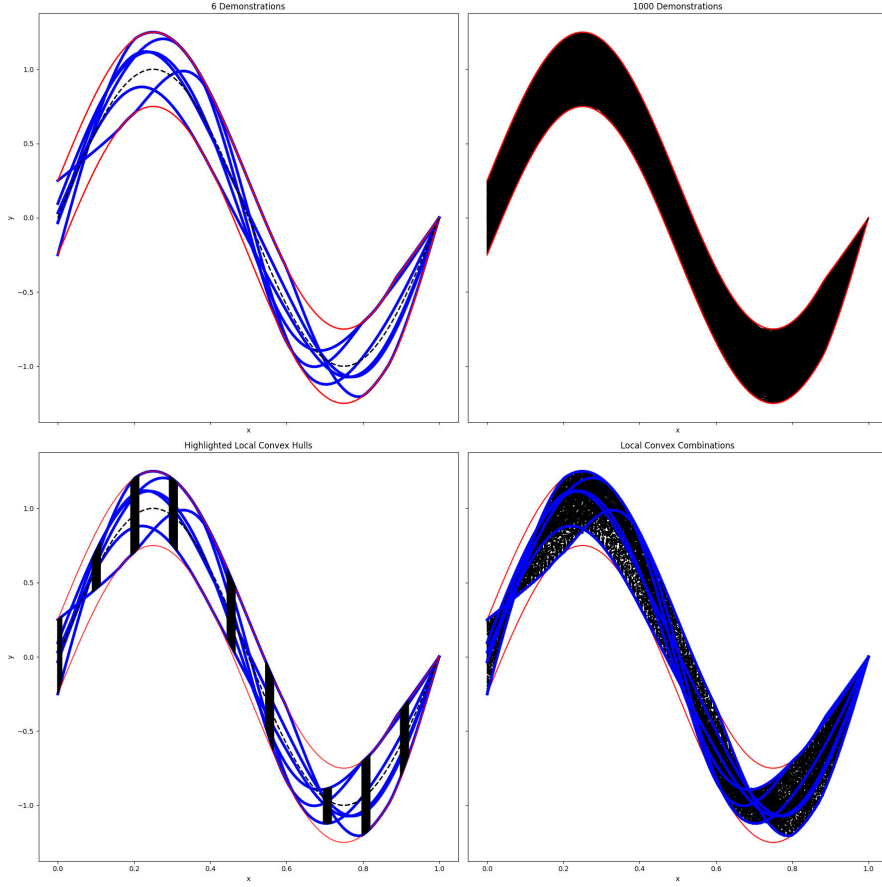


Figure 5.3: Demonstration support estimation on a sinusoidal test system. With sufficiently many demonstrations, the support is directly visible through trajectory occupancy. However, sparse demonstrations provide incomplete coverage of the admissible region. Instead, we approximate the support region using local convex sets formed from neighboring states along the demonstrated trajectories, whose union over time yields a denser estimate of the latent admissible region.

a result, direct occupancy-based estimation from the raw trajectories is often insufficient to recover the underlying support region. To address this limitation, we seek to enrich the observed support through additional samples that remain consistent with the demonstrated motions. One useful interpretation is to view admissible variation locally: around each stage of execution, states within the trajectory form a local convex set. Accordingly, sampling convex combinations of these states provides a mechanism for densely covering regions that would otherwise remain sparse.¹ For smooth and non-intersecting motions, the union of these enriched local neighborhoods over time provides a practical approximation of a globally non-convex support region.

Figure 5.3 illustrates this behavior using a sinusoidal test system. With many demonstrations, the admissible region is directly visible through dense trajectory occupancy. With only a few demonstrations, the region appears sparse and incomplete. By augmenting sparse temporal neighborhoods with additional interpolated samples and aggregating them across time, a substantially richer approximation of the latent support region is obtained while remaining constrained by the variability present in the demonstrations.

Once these support samples are generated, fitting an SDF to the resulting point set yields a smooth and differentiable approximation of the demonstration support region that can be queried continuously at arbitrary states. The learned SDF therefore provides both a geometric model of demonstrated behavior and a practical objective for optimization-based motion generation, where gradients of the field guide trajectories toward regions supported by demonstrations.

5.3 Neural Potential Functions for Motion Generation

The energy functions introduced in the previous section define continuous scalar fields over the state space, which form an energy landscape. A key advantage of such representations is that, when the functions are sufficiently smooth, their gradients are well-defined almost everywhere and provide meaningful geometric information about the local structure of the landscape. In

¹An equivalent interpretation is to generate many interpolated trajectories and aggregate samples within local temporal windows. We adopt this mechanism in practice for estimating local support, as described in the following section.

particular, the negative gradient of a scalar function defines the direction of steepest descent. Accordingly, by constructing continuous energy functions that encode task objectives and constraints, their gradients induce a vector field, which defines a dynamical system consistent with these objectives. As an illustrative example, the negative gradient of an SDF associated with an admissible region generates a vector field that drives the system toward this region along shortest paths.

Building on this perspective, we construct an energy landscape that encodes task progression while enforcing that the robot remains within an admissible region inferred from demonstrations. As motivated in Section 5.2, the nominal energy captures progression along the reference trajectory, shaping the landscape so that energy decreases in the direction of the desired motion. The safety energy formalizes the notion of demonstration support, guiding the system back toward the admissible region² when deviations occur. We now define these energy functions formally.

Given a discretized nominal trajectory $\{x_j^* = x^*(t_j)\}_{j=0}^N$, where N denotes the final index of the trajectory, nominal energy along this reference trajectory is defined as

$$\phi_{\text{nominal}}(x_k^*) = \sum_{j=k}^{N-1} \|x_{j+1}^* - x_j^*\|. \quad (5.1)$$

This quantity represents the remaining arc length to the goal, and therefore decreases monotonically as the system progresses along the trajectory. For states not lying exactly on the trajectory, the energy is approximated by projecting the state onto the closest point on the nominal path, resulting in

$$\phi_{\text{nominal}}(x) = \phi_{\text{nominal}}(x^*(\arg \min_k \|x_k^* - x\|)). \quad (5.2)$$

This construction extends the notion of task progression to the entire state space, encouraging motion that remains locally aligned with the nominal trajectory while ensuring $\phi_{\text{nominal}}(x) \geq 0$ for all x .

Given an SDF of the admissible region, the safety energy is given by

$$\phi_{\text{safety}}(x) = \text{relu}(\lambda_{\text{safety}}(\text{SDF}(x))) \quad (5.3)$$

where λ smooths the function near 0, ensuring that the gradients of ϕ_{safety}

²This region is also referred to as the safe set.

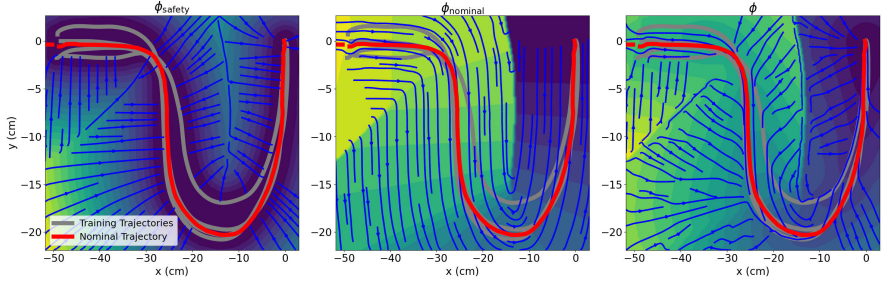


Figure 5.4: An example potential function in form of $\phi(x)$. Here, ϕ_{nominal} guides the robot to follow the nominal behavior while ϕ_{safety} guides the robot back to the safe set.

decay gradually as $\text{SDF}(x)$ approaches 0. By construction, $\phi_{\text{safety}}(x) \geq 0$ and vanishes for states within the admissible region.

The overall potential function is defined as the combination of these two terms,

$$\phi(x) = \phi_{\text{nominal}}(x) + \phi_{\text{safety}}(x), \quad (5.4)$$

which induces a vector field through its negative gradient. The resulting dynamical system is given by

$$\dot{x} = -\nabla_x \phi(x). \quad (5.5)$$

This formulation leads to two complementary behaviors. Within the admissible region, where the safety term is inactive, the system follows the direction induced by the nominal energy, resulting in motion aligned with the reference trajectory. Outside the admissible region, the safety term becomes active and drives the system back toward demonstrated states, while the nominal component continues to encourage progression toward the goal. This behavior is shown in Figure 5.4.

In the following subsection, we describe how both the nominal trajectory and the SDF of the admissible region are estimated from demonstration data.

Trajectory Synthesis for Nominal Trajectory and Admissible Region Estimation

The construction of the potential function requires both a nominal trajectory and an estimate of the admissible region. As discussed in Section 5.2, the nominal energy is defined with respect to an optimal reference trajectory, which is not available in practice, and the admissible region requires dense coverage of the demonstration support, which sparse demonstrations cannot provide directly. We address both requirements through a shared trajectory generation model.

This model captures the variability present in the demonstrations and generates additional trajectories consistent with the task. From the generated set, a representative trajectory is selected using Dynamic Time Warping (DTW) distance with respect to the demonstrations and used as the nominal trajectory. In parallel, the union of samples from all generated trajectories provides a dense approximation of the admissible region. Since trajectories are generated at aligned timesteps, the samples at each stage of execution correspond to convex combinations of neighboring demonstrated states, realizing the local convex support construction described in Section 5.2. Compared to explicitly constructing convex combinations at each timestep, the trajectory model provides a smoother and more coherent approximation of this union, while reducing sensitivity to temporal misalignment and sparsity in the demonstrations. This enriched point set enables reliable estimation of the SDF used to define the safety energy.

Phase-driven Potential Functions

Our current formulation defines a static potential function $\phi(x)$ that assigns a single scalar value to each state. While this induces a well-defined vector field, it introduces a fundamental limitation: the same state cannot correspond to different energy values at different stages of a task. As a result, trajectories that revisit the same region of the state space, such as intersecting or looping motions, cannot be represented unambiguously. This corresponds to the intersection problem introduced in Section 5.1.

A phase variable can resolve this limitation by augmenting the state representation, enabling the system to assign different velocities to the same state depending on the current stage of execution. Phase variables have been widely

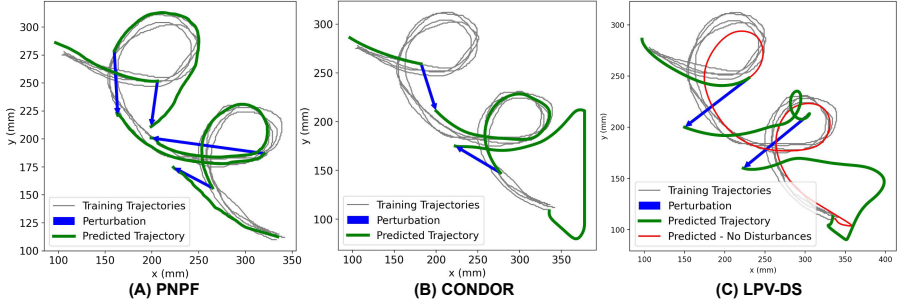


Figure 5.5: Perturbations on CONDOR, LPV-DS, and our method. Our method successfully recovers from perturbations, while other methods skip or repeat motion segments as they rely on velocity to infer task progression.

used in movement primitives for this purpose [25]. However, traditional phase variables rely on open-loop temporal schedules that advance independently of the system state. As a result, when perturbations displace the robot, the phase continues to progress regardless, leading to inconsistent behavior and loss of task continuity. This open-loop nature fundamentally limits reactivity, since the system cannot adjust its notion of task progression in response to deviations.

Rather than defining phase as an explicit function of time, we derive it directly from the nominal energy. Since ϕ_{nominal} decreases monotonically along the reference trajectory and can be evaluated continuously for arbitrary states, it serves as a natural intrinsic measure of task progression, yielding a phase variable that is driven by the state rather than an external temporal signal. This state-dependent phase evolves consistently with the system’s motion. Because the phase is derived locally from the current state, its evolution remains continuous and cannot exhibit abrupt jumps. As a result, when perturbations displace the system in state space, the phase estimate adjusts smoothly without skipping intermediate stages of execution.

In contrast, phase definitions based solely on projection onto a reference trajectory can introduce discontinuities. In particular, when trajectories contain loops or intersections, a perturbed state may be projected onto a distant segment of the trajectory, causing the system to skip intermediate stages of execution. Open-loop phase variables exhibit a similar limitation, as they evolve

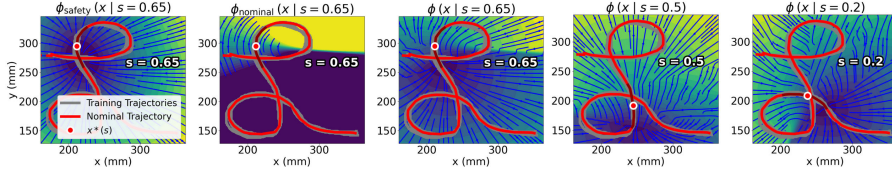


Figure 5.6: Phase-conditioned potential function $\phi(x | s)$ composed of safety and nominal energy functions, $\phi_{\text{safety}}(x | s)$ and $\phi_{\text{nominal}}(x | s)$, together with visualizations of $\phi(x | s)$ for different values of s .

independently of the state and may advance despite deviations. Similar issues also arise in dynamical systems that rely on higher-order state variables, where perturbations can lead to non-local transitions in task progression. These effects result in inconsistent behavior and loss of task continuity, as illustrated in Figure 5.5, where the system exhibits abrupt transitions across different segments of the motion following perturbations.

Incorporating the phase variable, we define a phase-conditioned potential function:

$$\phi(x | s) = \phi_{\text{nominal}}(x | s) + k_{\text{safety}} \Lambda_{\text{safety}}(s) \phi_{\text{safety}}(x | s), \quad (5.6)$$

where s denotes the phase, k_{safety} is the weight of the safety energy, and $\Lambda_{\text{safety}}(s)$ is a phase-dependent modulation that increases the influence of the safety term as the phase decreases. While not strictly required, this modulation biases the potential toward stronger safety corrections in later stages of the task.

The resulting dynamical system is given by

$$\dot{x} = -\alpha \nabla_x \phi(x | s), \quad (5.7)$$

$$s \leftarrow \phi_{\text{nominal}}(x' | s), \quad (5.8)$$

where x' denotes the state after applying $\dot{x}$. This formulation can be interpreted as defining a family of local potential functions, each associated with a particular stage of execution. In practice, this is realized by constructing the energies using local trajectory segments centered around the current phase. The nominal energy is derived from the corresponding segment of the reference trajectory, while the safety energy is constructed from neighboring segments

of the generated trajectories.

By restricting the energy construction to a local window of execution, both task progression and the admissible region are defined consistently at each stage of the task. This yields a phase-dependent energy landscape that evolves with the system state, enabling the representation of complex motions involving intersections and revisits. Figure 5.6 illustrates an example composition of the energy and potential functions at different phase values, demonstrating how the resulting dynamical system produces different velocity directions at similar states depending on the phase.

Modeling Periodic Motions and 6D Tasks

Periodic motions To model periodic motions, we introduce a circular encoding for both the trajectory representation and the phase-dependent potential. For the trajectory generation model, given a task period $\mathcal{T}$, we use a periodic input of the form

$$[\sin(2\pi \frac{t}{\mathcal{T}}), \cos(2\pi \frac{t}{\mathcal{T}})], \quad (5.9)$$

which ensures that generated trajectories are periodic and continuous across cycles.

For the control formulation, we represent the phase variable s using a corresponding circular embedding:

$$s^\circ = \left[\sin\left(2\pi \frac{s}{s_T}\right), \cos\left(2\pi \frac{s}{s_T}\right) \right], \quad (5.10)$$

where s_T denotes the phase corresponding to one full cycle. This allows the phase-conditioned energy functions to model cyclic behavior without discontinuities. Figure 5.7 illustrates this property, in which equivalent phase values induce identical vector fields, reflecting the periodicity of the representation.

The potential function is then defined as

$$\phi(x | s) = \begin{cases} \phi_{\text{nominal}}(x | s^\circ) + k_{\text{safety}} \phi_{\text{safety}}(x | s^\circ), & s \geq 0, \\ k_{\text{safety}} \phi_{\text{safety}}(x | s^\circ), & s < 0, \end{cases} \quad (5.11)$$

To maintain consistency with the periodic representation, the phase update

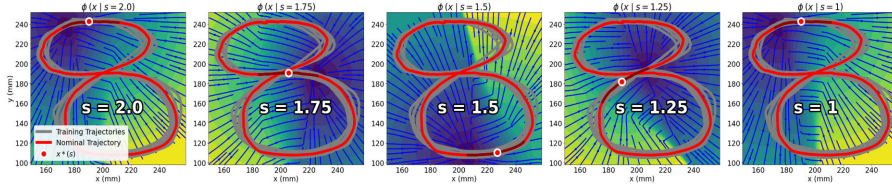


Figure 5.7: Periodic motions represented through a periodic phase parameterization. Equivalent phase values induce identical vector fields, as illustrated by the first and last panels.

is modified as

$$s \leftarrow s - (\phi_{\text{nominal}}(x \mid s^\circ) - \phi_{\text{nominal}}(x' \mid s^\circ)), \quad (5.12)$$

which preserves the interpretation of s as a progression variable while enabling cyclic evolution.

6D motions To extend the framework to motions involving both position and orientation, we represent orientations using axis-angle coordinates in the tangent space of a reference quaternion.

Given a quaternion trajectory $\{q_t\}_{t=1}^T$, we define the corresponding axis-angle representation as

$$r_t = \text{Log}(q_t \bar{q}_T), \quad (5.13)$$

where $\bar{q}_T$ is the conjugate of the reference quaternion q_T , and $\text{Log}(\cdot)$ denotes the logarithmic map from quaternions to axis-angle vectors [97].

This transformation yields a continuous representation of orientation trajectories, allowing them to be modeled jointly with position using the same framework. During execution, the gradient of the potential produces angular velocities directly. Alternatively, predicted axis-angle trajectories can be mapped back to quaternions via

$$\hat{q}_t = \text{Exp}(\hat{r}_t) q_T, \quad (5.14)$$

where $\text{Exp}(\cdot)$ denotes the exponential map.

This formulation enables the proposed method to handle full 6D motion without modifying the underlying potential-based control structure.

5.4 Stability Analysis of PNPF

The proposed formulation constructs a dynamical system by defining motion as the negative gradient of a learned potential function. While this provides an intuitive and flexible mechanism for motion generation, it is important to characterize the stability properties of the resulting closed-loop system.

In particular, we seek to understand under what conditions the induced dynamics converge to the desired target set, and how the decomposition into nominal and safety energies affects this behavior. We first analyze stability for a fixed phase (i.e., with ϕ independent of s), and then extend the analysis to the phase-dependent system. Since ϕ is learned, the resulting guarantees are regional rather than global, and hold only within the data-supported region of the state space.

We consider the potential-field controller $\dot{x} = -\alpha \nabla_x \phi(x)$ with $\alpha > 0$ and use the Lyapunov candidate $V(x) = \phi(x)$.

Lemma 1 (Stability of potential-field controller): *Let $S \subset \mathbb{R}^n$ be a target set. Suppose ϕ is differentiable and satisfies: (i) $\phi(x) \geq 0$ and $\phi(x) = 0$ iff $x \in S$; (ii) $\nabla_x \phi(x) \neq 0$ for all $x \notin S$. Then, along $\dot{x} = -\alpha \nabla_x \phi(x)$, $\dot{V}(x) = -\alpha \|\nabla_x \phi(x)\|^2 < 0 \quad \forall x \notin S$, and S is asymptotically stable (in the sense of set stability).*

Consider a potential constructed as the composition of multiple energy terms. If each energy term individually satisfies the conditions of Lemma 1 with respect to its target set, and if their composition does not introduce additional stationary points (e.g., through gradient cancellation such as $\nabla \phi_{\text{safety}} = -\nabla \phi_{\text{nominal}}$), then the composed potential preserves descent properties with respect to the intersection of the target sets.

In our case, we consider $\phi(x) = \phi_{\text{nominal}}(x) + \phi_{\text{safety}}(x)^3$, where each term is parameterized using neural fields with ReLU activations (the choice of architecture is discussed in the next section). This yields continuous, piecewise-smooth energy functions with gradients well-defined almost everywhere, except on a measure-zero set where activation boundaries are crossed⁴. The

³We omit weighting values for ϕ_{safety} for simplicity; scaling this weighting affects convergence speed and can temporarily dominate the phase term if chosen very large. However, its effect vanishes within the safe set, where the safety gradient is zero, and the stability argument remains unchanged.

⁴At these non-differentiable points, automatic differentiation returns a valid subgradient, ensuring that the controller remains numerically well-defined. Piecewise smoothness is sufficient in our setting due to the phase-windowed construction of the PNPF.

resulting closed-loop vector field is continuous and locally Lipschitz in the data-supported region, ensuring well-posed system dynamics.

The safety energy is a (smoothed) distance-to-set function to S_{safe} , hence $\phi_{\text{safety}}(x) \geq 0$ and vanishes iff $x \in S_{\text{safe}}$, with nonzero gradient away from the set. The nominal energy encodes task progress (e.g., remaining arc length) and vanishes on the terminal set S_{nominal} ; in practice it is computed via closest-point projection onto a discretized reference trajectory, which may introduce weakly varying regions or spurious critical points, while the neural-field parameterization regularizes this landscape and yields well-behaved gradients in the region of interest.

Consequently, $\phi(x) \geq 0$, $\phi(x) = 0 \iff x \in S_{\text{nominal}} \cap S_{\text{safe}} =: S$, so the first condition of Lemma 1 holds by construction. However, the absence of spurious stationary points, the second condition of Lemma 1, is not preserved under summation, as gradient cancellation may occur.

Assumption (A) (Local consistency) We assume that safety corrections do not oppose nominal behavior, i.e., $\nabla_x \phi_{\text{nominal}}(x \mid s)^\top \nabla_x \phi_{\text{safety}}(x \mid s) \geq -\varepsilon \|\nabla_x \phi_{\text{nominal}}(x \mid s)\|^2$, for a small $\varepsilon > 0$. This condition lower-bounds the interaction between the nominal and safety gradients, ensuring that safety corrections do not negate the nominal descent direction.

At the data level, this corresponds to the assumption that, within local regions of the state space, demonstrated trajectories share a dominant direction of motion, and safety-driven corrections are mostly orthogonal or only weakly opposing this direction. This effectively restricts the method to unimodal tasks, where local motion directions are consistent.

Under this assumption, the nominal and safety gradients remain sufficiently aligned to preserve descent, preventing destructive gradient cancellation. If this condition is violated, for example in multimodal tasks with conflicting local directions, gradient cancellation may occur and the stability guarantee no longer holds. Consequently, Assumption (A) supports the second condition of Lemma 1 in practice.

Phase variable We now extend the above Lyapunov analysis to the phase-dependent potential $\phi(x \mid s)$ by using the candidate $V(x, s) = \phi(x \mid s)$, whose derivative along the closed-loop dynamics is $\dot{V} = -\alpha \|\nabla_x \phi(x \mid s)\|^2 + \partial_s \phi(x \mid s) \dot{s}$. We use the phase update $s \leftarrow \phi_{\text{nominal}}(x \mid s)$ and consider trajec-

tories initialized such that $0 \leq s(0) \leq s_0$, corresponding to the demonstrated phase range. Since the phase is updated via a discrete rule, we analyze its evolution using a local continuous-time approximation. Specifically, we assume that phase updates occur over sufficiently small increments, i.e., $|\Delta s| < s_w$, such that the system remains within a local phase window. Within this window, the energy landscape can be treated as locally invariant with respect to s over an infinitesimal time interval. Accordingly, along trajectories we obtain $\dot{s} = \nabla_x \phi_{\text{nominal}}(x | s)^\top \dot{x} = -\alpha \nabla_x \phi_{\text{nominal}}(x | s)^\top \nabla_x \phi(x | s)$. Using Assumption (A), we have $\nabla_x \phi_{\text{nominal}}(x | s)^\top \nabla_x \phi(x | s) \geq (1 - \varepsilon) \|\nabla_x \phi_{\text{nominal}}(x | s)\|^2$, which is positive, since the nominal energy has well-behaved, non-vanishing gradients. Consequently, for $s > 0$, we obtain $\dot{s} < 0$, implying that $s(t)$ decreases monotonically and $s(t) \rightarrow 0$. Moreover, since the safety energy drives the state toward the safety region without opposing nominal progress (Assumption A) and becomes inactive inside it ($\phi_{\text{safety}}(x | s) = 0$), we have $\partial_s \phi(x | s) = \partial_s \phi_{\text{nominal}}(x | s)$ along typical executions. As $\partial_s \phi_{\text{nominal}}(x | s)$ has a fixed positive sign and $\dot{s} \leq 0$, it follows that $\partial_s \phi(x | s) \dot{s} \leq 0$.

Overall, under Assumption (A) the phase variable s decreases monotonically and satisfies $s(t) \rightarrow 0$, and the closed-loop dynamics follow a descent direction of the phase-dependent potential $V(x, s) = \phi(x | s)$. Along trajectories, $\dot{V} \leq 0$, and by LaSalle's invariance principle, trajectories initialized in the data-supported region converge to the largest invariant subset of S , establishing regional asymptotic convergence.

5.5 Smoothness of Energy Functions

The effectiveness of the proposed framework relies not only on the correctness of the energy formulation, but also on the smoothness of the resulting energy landscape. Since motion is generated through the negative gradient of the potential function, the quality of the induced vector field directly depends on the regularity of the energy function and its gradients. In particular, smooth and well-behaved gradients are essential for stable and predictable real-time control, as irregularities in the energy landscape can introduce abrupt changes in velocity or spurious stationary points.

To model such energy functions, we employ neural fields parameterized as multi-layer perceptrons (MLPs). These models define continuous scalar

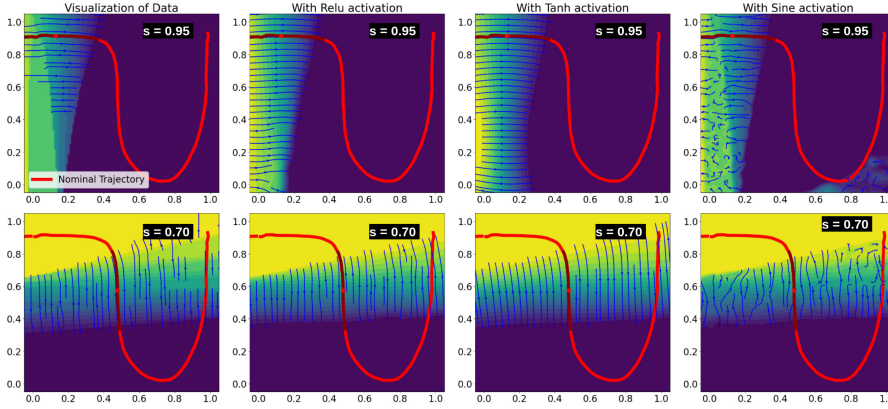


Figure 5.8: Nominal energy functions learned with different activation functions. ReLU and Tanh activations produce smoother energy functions, while Sine activation introduces saddle points.

functions over the state space and remain differentiable with respect to their inputs, allowing gradients to be computed analytically for control. In our formulation, the phase variable s is introduced through a hypernetwork that conditions the parameters of the neural field. This design separates spatial and temporal components: the energy is modeled as a smooth function of the state x , while phase-dependent variations are introduced through parameter modulation. As a result, gradients with respect to the state remain continuous and well-behaved, ensuring stable control behavior.

To enable sufficient expressiveness across different stages of execution, we apply a positional encoding to the phase variable before conditioning. Specifically, we use a Fourier-style embedding of s , which allows the model to represent higher-frequency variations in phase. This is important for capturing behaviors such as sharp turns that occur at different stages of the motion. Importantly, this additional expressiveness is introduced only along the phase dimension, while the spatial dependence of the energy function remains smooth.

The choice of activation function further influences the geometry of the learned energy landscape. As illustrated in Figure 5.8, periodic activations such as sine functions tend to introduce oscillatory behavior, leading to unintended saddle points and local extrema that disrupt stable motion generation. In contrast, both ReLU and Tanh activations produce smoother energy

functions. However, in practice, we observe that Tanh activations can still introduce saddle structures that lead to undesired local minima, hindering consistent convergence. This behavior is consistent with our empirical observations, where ReLU-based models exhibit more reliable convergence.

ReLU activations provide a favorable balance, yielding piecewise-smooth energy functions with well-defined gradients almost everywhere while avoiding oscillatory artifacts. In addition, their piecewise-linear structure tends to regularize small-scale variations in the data, effectively smoothing out local irregularities that may arise from noisy or sparse demonstrations. Although ReLU is not globally smooth, this does not pose a practical limitation: gradients are well-defined almost everywhere, and automatic differentiation provides valid subgradients at non-differentiable points, ensuring numerically stable evaluation.

The phase-driven formulation further mitigates the need for global smoothness by restricting the energy construction to local regions of the trajectory. Since each phase corresponds to a localized segment of execution, the energy functions only need to model local variations rather than the entire trajectory globally. This reduces the complexity of the learned function and the risk of spurious critical points arising from globally inconsistent energy landscapes. Combined with the piecewise-smooth structure of ReLU networks, this locality ensures that the resulting vector field remains suitable for stable real-time control.

Summary: This chapter introduces the key concepts underlying reactive motion generation using learned energy landscapes. These include representing robot behavior through complementary energy functions, deriving a state-dependent phase variable to resolve ambiguities arising from state revisits, and generating reactive motion through the gradients of neural potential functions. The formulation is further extended to periodic and 6D motions, and its stability and smoothness properties are discussed. Further details, implementation aspects, and experimental evaluations are provided in Paper C.

CHAPTER 6

Summary of included papers

This chapter provides a summary of the included papers.

6.1 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.

©2023 IEEE 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.

AET contributed to the conceptualization of the ideas, implementation, experimental design and execution, analysis of results, and manuscript writing. MPD contributed to manuscript review and provided feedback on clarity and presentation. YB contributed to the discussion and refinement of the ideas, project supervision, and manuscript review and editing.

6.2 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.
©2023 IEEE.

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.

AET contributed to the conceptualization of the ideas, implementation, experimental design and execution, analysis of results, and manuscript writing. MPD contributed to manuscript review and provided feedback on clarity and presentation. YB contributed to the discussion and refinement of the ideas, project supervision, and manuscript review and editing.

6.3 Paper C

Ahmet Ercan Tekden, Dimitrios Kanoulas, Aude Billard, Yasemin Bekiroglu

Reactive Motion Generation via Phase-varying Neural Potential Functions

IEEE Robotics and Automation Letters, vol. 11, no. 7, July 2026.

©2026 IEEE DOI: 10.1109/LRA.2026.3692035.

This paper introduces a reactive motion generation framework that combines the strengths of dynamical systems and phase-based methods for learning from demonstration. The approach models motion as the gradient of a learned potential function conditioned on a state-dependent phase variable estimated directly from the system state, enabling the representation of tasks with repeated or intersecting trajectories while preserving reactivity to perturbations. The potential function is composed of a nominal energy capturing task progression and a safety energy encoding the region supported by demonstrations, resulting in stable motion generation. The energy functions are parameterized using neural fields, providing smooth and flexible representations that generalize across point-to-point, periodic, and 6D manipulation tasks. The method is evaluated in simulation and real-world experiments, demonstrating improved performance over baseline approaches, particularly in tasks involving intersections and external disturbances.

AET contributed to the conceptualization of the ideas, implementation, experimental design and execution, analysis of results, and manuscript writing. DK and AB contributed to manuscript review and provided feedback on clarity and presentation. YB contributed to the discussion and refinement of the ideas, project supervision, and manuscript review and editing.

6.4 Paper D

Ahmet Ercan Tekden, Yasemin Bekiroglu

Compositional Motion Generation from Demonstration with Object-Centric Neural Fields

IEEE Robotics and Automation Letters, vol. 11, no. 9, September 2026.

©2026 IEEE DOI: 10.1109/LRA.2026.3713713 .

This paper proposes a compositional learning-from-demonstration framework for trajectory generation from visual input. Scenes are modeled using object-centric neural fields that capture smooth geometric and positional variations through latent-conditioned deformations, while motion generation is formulated using a temporal mixture-of-experts that combines object-conditioned movement primitives over time. The resulting framework jointly models scene structure and robot motion through shared latent representations, enabling data-efficient trajectory generation and systematic generalization across scene configurations. Experiments in simulation and on real robots demonstrate robust manipulation from few demonstrations, category-level generalization using language-guided segmentation and operation on 3D scene representations.

AET contributed to the conceptualization of the ideas, implementation, experimental design and execution, analysis of results, and manuscript writing. YB contributed to the discussion and refinement of the ideas, project supervision, and manuscript review and editing.

CHAPTER 7

Concluding Remarks and Future Work

This thesis investigated how representation learning can improve data efficiency in robotic grasping and manipulation. In particular, it explored neural fields as a unified framework for modeling geometry, scenes, and robot motion from limited demonstrations. Across the presented works, we showed that structured representations can substantially reduce data requirements while enabling reactivity and generalization across objects, scenes, and task variations.

A central insight of this thesis is that many key components of robotic manipulation, including geometry, task trajectories, and motion generation, can be represented as continuous functions over their underlying coordinates. Rather than treating these components as separate problems, this perspective makes it possible to model them within a common mathematical framework.

Neural fields provide a natural way to learn such coordinate-based functions. By parameterizing continuous functions with neural networks, they can represent complex geometric structures and motion patterns while remaining continuous and differentiable. This enables the same representation to be applied across different aspects of robotic manipulation.

The resulting representations are compact, support smooth interpolation

between observations, and facilitate structured generalization and reactive control. These properties are particularly valuable in data-constrained robotic settings, where they allow manipulation skills to generalize across objects, scenes, and task variations from relatively few demonstrations. Together with the expressive approximation capabilities of neural networks, these representations provide a flexible framework for conditioning on scene, object, task, and phase variables while maintaining a common continuous formulation.

The main contributions of this thesis can be summarized as follows:

- **Local surface representations for grasp transfer:** We introduced local surface-based descriptors modeled with neural fields, enabling grasp poses to be transferred across object categories by exploiting shared geometric structure. This approach improves spatial precision and enables data-efficient generalization beyond category-specific training.
- **Joint modeling of scenes and motions:** We proposed neural field representations that model scenes and motions as smooth functions of shared latent variables. This formulation enables learning movement primitives from a small number of demonstrations and supports generalization across scene variations through interpolation in latent space.
- **Reactive motion generation via neural potential functions:** We developed a neural potential function framework for reactive motion generation, in which complementary energy functions are represented by neural fields and their gradients jointly induce robot motion. A state-dependent phase formulation enables the representation of complex motion patterns while preserving reactivity to disturbances.
- **Compositional representations for manipulation:** We extended our movement primitive framework to object-centric and compositional settings, allowing scenes and motions to be decomposed into reusable components. This provides an additional source of data efficiency by leveraging shared structure across tasks.

Overall, these contributions demonstrate that incorporating structure in representations, through geometry, smoothness, and compositionality, improves data efficiency and generalization in robotic manipulation.

Future Work Several promising directions remain for future work. An important extension is the modeling of multi-finger and dexterous manipulation policies, where coordinated contact interactions introduce additional complexity in both representation and control. This includes learning functional grasp configurations and coordinating finger motion throughout task execution.

Another direction is the development of representations that operate across multiple task spaces and primitives, enabling the automatic coordination of multiple motion components within a unified framework. Such representations could allow complex behaviors to be composed from simpler learned primitives, for example, in bimanual manipulation tasks where both hands must be synchronized.

Finally, while this thesis explored several components independently, integrating them into a unified framework remains an important direction. Combining geometry, motion, and control representations within a single system could enable more coherent and scalable manipulation strategies.

References

- [1] R. Newbury et al., “Deep learning approaches to grasp synthesis: A review,” *IEEE Transactions on Robotics*, 2023.
- [2] D. Nguyen-Tuong and J. Peters, “Model learning for robot control: A survey,” *Cognitive processing*, vol. 12, pp. 319–340, 2011.
- [3] 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.
- [4] S. Schaal, “Is imitation learning the route to humanoid robots?” *Trends in cognitive sciences*, vol. 3, no. 6, pp. 233–242, 1999.
- [5] 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.
- [6] B. Apolloni, A. Ghosh, F. Alpaslan, and S. Patnaik, *Machine learning and robot perception*. Springer Science & Business Media, 2005, vol. 7.
- [7] 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.
- [8] J. H. Connell and S. Mahadevan, *Robot learning*. Springer Science & Business Media, 2012, vol. 233.
- [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] Z. Fu, T. Z. Zhao, and C. Finn, “Mobile aloha: Learning bimanual mobile manipulation with low-cost whole-body teleoperation,” in *Conference on Robot Learning (CoRL)*, 2024.
- [12] C. Chi et al., “Universal manipulation interface: In-the-wild robot teaching without in-the-wild robots,” in *Proceedings of Robotics: Science and Systems (RSS)*, 2024.
- [13] G. Dulac-Arnold et al., “Challenges of real-world reinforcement learning: Definitions, benchmarks and analysis,” *Machine Learning*, vol. 110, no. 9, pp. 2419–2468, 2021.
- [14] L. P. Kaelbling, “The foundation of efficient robot learning,” *Science*, vol. 369, no. 6506, pp. 915–916, 2020.
- [15] E. S. Spelke, K. Breinlinger, J. Macomber, and K. Jacobson, “Origins of knowledge.,” *Psychological review*, vol. 99, no. 4, p. 605, 1992.
- [16] 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.
- [17] O. Williams and A. Fitzgibbon, “Gaussian process implicit surfaces,” in *Gaussian Processes in Practice*, 2006.
- [18] Y. Xie, T. Takikawa, S. Saito, et al., “Neural fields in visual computing and beyond,” *Computer Graphics Forum*, 2022, ISSN: 1467-8659.
- [19] D. Foster, *Generative deep learning*. " O'Reilly Media, Inc.", 2022.
- [20] 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.
- [21] S. Schaal, “Learning from demonstration,” *Advances in neural information processing systems*, vol. 9, 1996.
- [22] 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.

-
- [23] 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.
 - [24] 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.
 - [25] 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.
 - [26] A. Paraschos, C. Daniel, J. R. Peters, and G. Neumann, “Probabilistic movement primitives,” *Advances in neural information processing systems*, vol. 26, 2013.
 - [27] S. Calinon, “A tutorial on task-parameterized movement learning and retrieval,” *Intelligent service robotics*, vol. 9, pp. 1–29, 2016.
 - [28] M. Y. Seker, M. Imre, J. H. Piater, and E. Ugur, “Conditional neural movement primitives,” in *Robotics: Science and Systems*, vol. 10, 2019.
 - [29] 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.
 - [30] 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.
 - [31] 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.
 - [32] 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.

- [33] 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.
- [34] 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.
- [35] G. Yang, S. Belongie, B. Hariharan, and V. Koltun, “Geometry processing with neural fields,” in *Thirty-Fifth Conference on Neural Information Processing Systems*, 2021.
- [36] 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.
- [37] 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.
- [38] 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.
- [39] 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.
- [40] 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.
- [41] T. Weng, D. Held, F. Meier, and M. Mukadam, “Neural grasp distance fields for robot manipulation,” *IEEE International Conference on Robotics and Automation (ICRA)*, 2023.
- [42] Y. Chen, X. Gao, K. Yao, L. Niederhauser, Y. Bekiroglu, and A. Bilard, “Implicit articulated robot morphology modeling with configuration space neural signed distance functions,” in *2025 IEEE International Conference on Robotics and Automation (ICRA)*, IEEE, 2025, pp. 4558–4564.

-
- [43] 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.
 - [44] L. Ruthotto and E. Haber, “An introduction to deep generative modeling,” *GAMM-Mitteilungen*, vol. 44, no. 2, e202100008, 2021.
 - [45] 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.
 - [46] 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.
 - [47] 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.
 - [48] 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.
 - [49] 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.
 - [50] D. Ha, A. M. Dai, and Q. V. Le, “Hypernetworks,” in *International Conference on Learning Representations*, 2017.
 - [51] T. Galanti and L. Wolf, “On the modularity of hypernetworks,” *Advances in Neural Information Processing Systems*, vol. 33, pp. 10 409–10 419, 2020.
 - [52] E. Perez, F. Strub, H. De Vries, V. Dumoulin, and A. Courville, “Film: Visual reasoning with a general conditioning layer,” in *Proceedings of the AAAI conference on artificial intelligence*, vol. 32, 2018.
 - [53] E. Dupont, H. Kim, S. Eslami, D. Rezende, and D. Rosenbaum, “From data to functa: Your data point is a function and you can treat it like one,” *arXiv preprint arXiv:2201.12204*, 2022.

- [54] 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.
- [55] N. Rahaman 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.
- [56] 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.
- [57] 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.
- [58] A. ten Pas, M. Gualtieri, K. Saenko, and R. Platt, “Grasp pose detection in point clouds,” *The International Journal of Robotics Research*, vol. 36, no. 13-14, pp. 1455–1473, 2017.
- [59] A. Mousavian, C. Eppner, and D. Fox, “6-dof graspnet: Variational grasp generation for object manipulation,” in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2019, pp. 2901–2910.
- [60] H.-S. Fang, C. Wang, M. Gou, and C. Lu, “Graspnet-1billion: A large-scale benchmark for general object grasping,” in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2020, pp. 11 444–11 453.
- [61] M. Sundermeyer, A. Mousavian, R. Triebel, and D. Fox, “Contact-graspnet: Efficient 6-dof grasp generation in cluttered scenes,” in *2021 IEEE International Conference on Robotics and Automation (ICRA)*, IEEE, 2021, pp. 13 438–13 444.
- [62] Y. Ze, N. Hansen, Y. Chen, M. Jain, and X. Wang, “Visual reinforcement learning with self-supervised 3d representations,” *IEEE Robotics and Automation Letters*, vol. 8, no. 5, pp. 2890–2897, 2023.
- [63] Y. Ze, G. Zhang, K. Zhang, C. Hu, M. Wang, and H. Xu, “3d diffusion policy: Generalizable visuomotor policy learning via simple 3d representations,” in *Proceedings of Robotics: Science and Systems (RSS)*, 2024.

-
- [64] M. Shridhar, L. Manuelli, and D. Fox, “Perceiver-actor: A multi-task transformer for robotic manipulation,” in *6th Annual Conference on Robot Learning*, 2022.
 - [65] T. Gervet, Z. Xian, N. Gkanatsios, and K. Fragkiadaki, “Act3d: 3d feature field transformers for multi-task robotic manipulation,” in *7th Annual Conference on Robot Learning*, 2023.
 - [66] A. Goyal, J. Xu, Y. Guo, V. Blukis, Y.-W. Chao, and D. Fox, “RVT: Robotic view transformer for 3d object manipulation,” in *7th Annual Conference on Robot Learning*, 2023.
 - [67] T.-W. Ke, N. Gkanatsios, and K. Fragkiadaki, “3d diffuser actor: Policy diffusion with 3d scene representations,” in *8th Annual Conference on Robot Learning*, 2024.
 - [68] 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.
 - [69] 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.
 - [70] F. T. Pokorny, K. Hang, and D. Kragic, “Grasp moduli spaces,” in *Robotics: Science and Systems*, 2013.
 - [71] 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.
 - [72] 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.
 - [73] 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.
 - [74] 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.

- [75] 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.
- [76] 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.
- [77] 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.
- [78] J.-L. Blanco, “A tutorial on se (3) transformation parameterizations and on-manifold optimization,” *University of Malaga, Tech. Rep*, vol. 3, p. 6, 2010.
- [79] 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.
- [80] T. Kulak, J. Silvério, and S. Calinon, “Fourier movement primitives: An approach for learning rhythmic robot skills from demonstrations,” in *Robotics: Science and systems*, 2020.
- [81] G. Li, Z. Jin, M. Volpp, F. Otto, R. Lioutikov, and G. Neumann, “Prodmp: A unified perspective on dynamic and probabilistic movement primitives,” *IEEE Robot. Autom. Lett.*, vol. 8, no. 4, pp. 2325–2332, 2023.
- [82] Y. Yildirim and E. Ugur, “Conditional neural expert processes for learning movement primitives from demonstration,” *IEEE Robot. Autom. Lett.*, 2024.
- [83] S. Calinon, “Learning from demonstration (programming by demonstration),” *Encyclopedia of robotics*, pp. 1–8, 2018.
- [84] A. Gams, A. Ude, J. Morimoto, et al., “Deep encoder-decoder networks for mapping raw images to dynamic movement primitives,” in *2018 IEEE International Conference on Robotics and Automation (ICRA)*, IEEE, 2018, pp. 5863–5868.

-
- [85] H.-T. D. Liu, F. Williams, A. Jacobson, et al., “Learning smooth neural functions via lipschitz regularization,” in *ACM SIGGRAPH 2022 Conference Proceedings*, 2022, pp. 1–13.
 - [86] A. Billard, S. Mirrazavi, and N. Figueroa, *Learning for adaptive and reactive robot control: a dynamical systems approach*. Mit Press, 2022.
 - [87] J.-J. E. Slotine, W. Li, et al., *Applied nonlinear control*. Prentice hall Englewood Cliffs, NJ, 1991, vol. 199.
 - [88] S. M. Khansari-Zadeh and A. Billard, “Learning stable nonlinear dynamical systems with gaussian mixture models,” *IEEE Trans. Robot.*, vol. 27, no. 5, pp. 943–957, 2011.
 - [89] N. Figueroa and A. Billard, “A physically-consistent bayesian non-parametric mixture model for dynamical system learning,” in *Conf. Robot. Learn. (CoRL)*, 2018, pp. 927–946.
 - [90] S. M. Khansari-Zadeh and O. Khatib, “Learning potential functions from human demonstrations with encapsulated dynamic and compliant behaviors,” *Auton. Robots*, vol. 41, pp. 45–69, 2017.
 - [91] M. A. Rana et al., “Learning reactive motion policies in multiple task spaces from human demonstrations,” in *Conf. Robot. Learn. (CoRL)*, 2020, pp. 1457–1468.
 - [92] M. A. Rana, A. Li, D. Fox, B. Boots, F. Ramos, and N. Ratliff, “Euclideanizing flows: Diffeomorphic reduction for learning stable dynamical systems,” in *Learning for Dynamics and Control*, 2020, pp. 630–639.
 - [93] R. Pérez-Dattari and J. Kober, “Stable motion primitives via imitation and contrastive learning,” *IEEE Trans. Robot.*, 2023.
 - [94] H. Tsukamoto, S.-J. Chung, and J.-J. E. Slotine, “Contraction theory for nonlinear stability analysis and learning-based control: A tutorial overview,” *Annual Reviews in Control*, vol. 52, pp. 135–169, 2021.
 - [95] P. Florence, C. Lynch, A. Zeng, et al., “Implicit behavioral cloning,” in *Conference on Robot Learning*, PMLR, 2022, pp. 158–168.
 - [96] Y. Li and S. Calinon, “From movement primitives to distance fields to dynamical systems,” *IEEE Robotics and Automation Letters*, 2025.
 - [97] F. S. Grassia, “Practical parameterization of rotations using the exponential map,” *J. Graph. Tools*, vol. 3, no. 3, pp. 29–48, 1998.

Part II

Papers

PAPER **A**

**Grasp Transfer based on Self-Aligning Implicit Representations of
Local Surfaces**

Ahmet Ercan Tekden, Marc Peter Deisenroth, Yasemin Bekiroglu

IEEE Robotics and Automation Letters, vol. 8, no. 4, Oct. 2023.
©2023 IEEE DOI: 10.1109/LRA.2023.3306272

The layout has been revised.

Abstract

Objects we interact with and manipulate often share similar parts, such as handles, that allow us to transfer our actions flexibly due to their shared functionality. This work addresses the problem of transferring a grasp experience or a demonstration to a novel object that shares shape similarities with objects the robot has previously encountered. Existing approaches for solving this problem are typically restricted to a specific object category or a parametric shape. Our approach, however, can transfer grasps associated with implicit models of local surfaces shared across object categories. Specifically, we employ a single expert grasp demonstration to learn an implicit local surface representation model from a small dataset of object meshes. At inference time, this model is used to transfer grasps to novel objects by identifying the most geometrically similar surfaces to the one on which the expert grasp is demonstrated. Our model is trained entirely in simulation and is evaluated on simulated and real-world objects that are not seen during training. Evaluations indicate that grasp transfer to unseen object categories using this approach can be successfully performed both in simulation and real-world experiments. The simulation results also show that the proposed approach leads to better spatial precision and grasp accuracy compared to a baseline approach.

1 Introduction

Grasp synthesis has been studied extensively [1] as robotic grasping skills have a significant impact on the success of subsequent manipulation tasks. A common approach for generating grasps is to train a prediction network, which either takes top-down scene images [2], [3] or 3D point clouds [4], [5], [6], [7], [8] for predicting multiple grasp locations. However, such models are data-hungry, and have to use non-curated data since data annotation at scale is often not feasible in robotics. Another approach for generating grasp poses is based on finding category-level correspondences [9], [10], [11] and

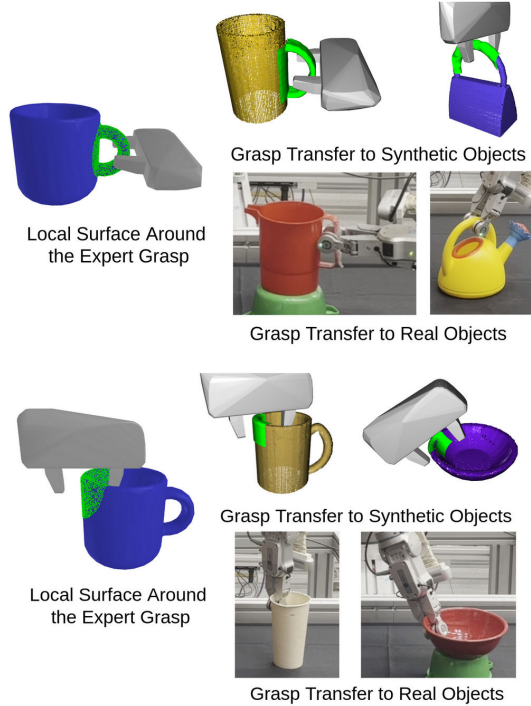


Figure 1: Example experiments of grasp transfer to various objects. Utilizing implicit representations of local surfaces, grasps are transferred to novel objects by identifying local surfaces that are geometrically similar to those surrounding the expert grasps.

using them for grasp prediction. Compared to grasp prediction networks, correspondence-based methods are more data-efficient and allow for transferring grasp demonstrations to other objects, thereby facilitating learning from demonstration [12], and imitation learning [13]. However, correspondence-based methods can predict correct grasp locations only on the trained object categories.

Compared to approaches for synthesizing grasp candidates from scratch [14], [15] or relying on object-level correspondences [9], [10], [11], making use of previous experiences/demonstrations from similar objects and transferring grasps onto new object instances can be more data efficient [16] and facilitate task-

oriented grasping [17]. In this study, we address the problem of reusing grasping knowledge when faced with a similar object without planning from scratch. Specifically, we focus on transferring grasps across object categories that share geometrically similar surfaces/parts, as illustrated in Figure 1. Using our method, we can match the local surfaces learned from, e.g. the handle or the rim of a mug object, to identify the corresponding grasp location on other objects such as bags and bowls that share similar parts.

In this work, we model local surfaces with implicit representations using neural fields [18] and perform shape inference by optimizing latent shape codes. Neural fields are position-based neural networks that take continuous spatial coordinates as input to parametrize a variety of functions, e.g. implicit surfaces [19], [20]. For latent shape code optimization, most neural field methods require all objects to be in canonical reference frames both at training and inference time [19]. Aligning objects to the exact canonical poses is challenging as it requires the prediction of object poses [21], [22], which often introduces a certain degree of error. We propose a novel approach for aligning object or surface poses to a common reference frame through pose embedding optimization. We rely on a small dataset of object models from a single category and a single grasp demonstration on one object within that category. By using the demonstrated grasp and the corresponding surface around it as a reference, we extract geometrically similar surfaces from the object models. These are then used to train an implicit surface representation. At inference time, the learned implicit representations are employed to identify surfaces on novel objects, based on the ones that they are trained on. This is achieved without exact pose knowledge and enables us to transfer grasps across different objects. The main contributions of this work can be summarized as follows:

- We present an end-to-end surface reconstruction approach that simultaneously learns to reconstruct, scale, and align shapes in a self-supervised manner. The proposed method facilitates learning implicit representations without complete surface match and alignment.
- We introduce a novel approach for learning local surface models using a sampling sphere placed on a provided reference frame. This sphere is used for sampling the training data and by adapting it to the current alignment pose, the local surface is modeled. This in turn allows for learning the local implicit surfaces on explicitly provided locations via

demonstrated grasps.

- Our approach successfully transfers grasp demonstrations from a single object to other objects, including ones from unseen categories. We utilize learned implicit representations to identify and localize local surfaces on test objects that are geometrically similar to those surrounding the demonstrated grasps, thereby enabling the transfer of grasp poses.
- In our results, we show that our method acquires better spatial precision and grasp accuracy compared to a state of the art baseline method. In addition, it acquires competitive performance on grasp transfer to novel objects in both simulation and real-world experiments.

2 Related Work

Early works on grasp prediction that rely on explicit shape information are usually built on primitive-based correspondences [23], [24], parametrization using smooth differentiable functions (e.g. Grasp Moduli Spaces [25], [26]), or hard-coded geometric features [27]. Primitive-based methods, in general, can struggle to generalize to objects that deviate significantly from the provided primitives and parametrization can deteriorate with partial point cloud data. Additionally, most other geometric feature-based approaches require explicit design and engineering effort. In contrast, our method learns local surface representations from surfaces extracted from an object dataset, enabling it to better generalize to shape deviations. Furthermore, by optimizing a latent shape code through shape reconstruction, our method can utilize partial point clouds of objects.

Various representations have been proposed for building models related to object parts to be used for grasp planning [16], [28]. [16] uses prototype parts to transfer grasps across novel object categories using a similarity measure to identify parts with similar shapes. However, these methods do not explicitly model local object parts, which can limit their ability to capture important shape details. In contrast, our approach explicitly models local surfaces, allowing it to capture more information about the underlying shapes.

Another approach for grasp transfer is to use category-level key points to transfer grasps to objects in the same category [9], [10], [11], [29], [30], [31]. However, such keypoint-based methods are category driven, and by design,

they cannot be utilized to transfer demonstrated grasps to novel object categories. Instead of modeling object-level correspondences, our network learns implicit representations of local surfaces. For representing each different surface type, we train individual shape reconstruction models. Our method leverages these models to identify grasp poses corresponding to each model’s local surface type. When the encountered surface resembles the local surfaces the model is trained on, it can accurately reconstruct its shape. However, when presented with a surface that deviates significantly from what the model is trained on, the model will fail to reconstruct its shape, leading to higher errors. Employing these local surface models enables our method to successfully transfer grasps to novel objects.

We model local surfaces by leveraging coordinate-based neural networks, which have proven effective in representing 3D shapes and scenes in prior research [19], [20], [32], [33], [34]. However, these networks do not address the task of modeling local surfaces. To achieve this, we utilize a single grasp demonstration to extract geometrically similar local surfaces from a dataset of an object category. However, due to geometric variations among the objects, the poses of extracted local surfaces are not properly aligned to the reference frame of the original grasp demonstration. The proposed network architecture addresses this issue by aligning the poses of these surfaces to the desired reference frame while simultaneously learning to reconstruct their shapes. Recently it has been shown that camera poses can be estimated through neural radiance field (NeRF) training [35], [36]. However, these methods are limited to images captured from a single scene that contain patches with matching features that guide the alignment process. In contrast, our approach aligns poses of local surfaces coming from different objects by leveraging their geometric similarities.

3 Method

The overall architecture of the proposed approach is shown in Figure 2. The proposed architecture learns a surface model while aligning the trained shapes to a shared common reference frame, as illustrated in step (2-a) in Figure 2. The shape alignment is localized to the surface around an expert grasp and is used to find the corresponding grasp locations on the rest of the training objects within the same category, as illustrated in step (2-b) in Figure 2. At

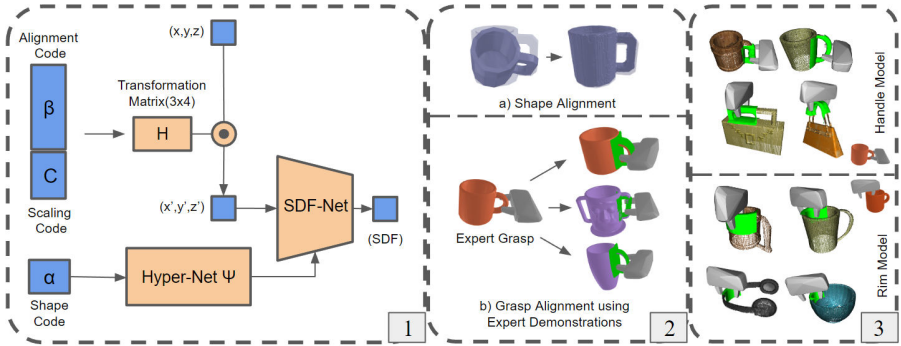


Figure 2: The proposed end-to-end network architecture, with self-aligning implicit representations (1). During Training (2), it simultaneously aligns objects with each other while learning to reconstruct their shape (2-a). When this is done on local surfaces around the expert grasp demonstration, it aligns grasps within an object category (2-b). At inference time (3), learned models are used to identify and localize grasp locations on novel objects by finding surfaces similar to those on which the networks are trained on.

inference time, the learned surface model is employed to transfer grasps to novel objects by identifying and localizing geometrically similar surfaces to the ones the model is trained on, as illustrated in step (3) in Figure 2. In this section, we will discuss the details of our approach.

3.1 Preliminaries

In our architecture, we build a signed-distance function (SDF) [19] based representation of local surfaces for shape reconstruction. SDF is an implicit function that represents the distance of its input point to the closest surface. SDF evaluates to negative values for the input points inside the object to be modeled, to positive values for the points outside the object and to 0 for the points on the object surface.

Our method consists of two MLP-based networks: SDF-Net, which predicts the SDF value for a queried position on the object, and Hyper-Net [37], which takes a shape code and predicts a part of the SDF-Net’s weights. In our method, each reconstructed shape is assigned a shape code, and these shape codes are optimized together with the network weights during the training.

In addition, we use position encodings, as they improve network performance on complex shapes [38]. Position encodings map 3D locations, X , to higher-dimensional inputs. We employ the positional encoding map:

$$\Gamma(X) = [X, \Gamma_0(X), \Gamma_1(X), \dots, \Gamma_{L-1}(X)] \in \mathbb{R}^{3+6L}$$

where $\Gamma_m(X) = [\cos(2^m \pi X), \sin(2^m \pi X)] \in \mathbb{R}^6$.

3.2 Pose Alignment of Shapes

The proposed architecture simultaneously finds 3D transformations that align the poses of local surfaces with each other while reconstructing their shapes. To train the system end-to-end, 3D transformations are modeled using $\mathfrak{se}(3)$ Lie algebra: $\mathbf{SE}(3)$ transformation has an associated Lie algebra $\mathfrak{se}(3)$. Using the exponential map $\mathfrak{se}(3) \rightarrow \mathbf{SE}(3)$, we map 6-dimensional embeddings to transformations that continuously lie on $\mathbf{SE}(3)$ manifolds [39]. Let $\beta = \langle \omega, t \rangle \in \mathfrak{se}(3)$ be a six-dimensional vector where ω , t are three-dimensional vectors corresponding to rotation and translation. Equivalently, alignment transformation is presented by the matrix $T_{\text{alignment}} = \begin{bmatrix} R_{3 \times 3} & v_{3 \times 1} \end{bmatrix} \in \mathbf{SE}(3)$ with $v = Vt$. Using the Rodrigues formula, R and V are written as

$$R = e^{[\omega]} = I_3 + \frac{\sin \theta}{\theta} [\omega] + \frac{1 - \cos \theta}{\theta^2} [\omega]^2, \quad (\text{A.1})$$

$$V = I_3 + \frac{1 - \cos \theta}{\theta^2} [\omega] + \frac{1 - \sin \theta}{\theta^3} [\omega]^2, \quad (\text{A.2})$$

where $[\omega]$ is the skew-symmetric matrix created from the vector ω , and $\theta = \|\omega\|$. To estimate the alignment transformation matrix $T_{\text{alignment}}$ from β , we use a Taylor-series expansion for linearization. This formulation is well defined, surjective, and allows for gradient-based optimization of β . At $\beta = 0$, $T_{\text{alignment}}$ corresponds to the identity transformation.

For scaling the objects we learn a scaling embedding $C \in \mathbb{R}^3$ that is then transformed to a 4×4 transformation matrix $T_{\text{scaling}} = \text{diag}([C, 1])$. Multiplying this matrix with $T_{\text{alignment}}$, we acquire an affine transformation matrix $H = T_{\text{alignment}} T_{\text{scaling}}$ that is used for pose and scale alignment. H is used to transform a given X location into X' via $X' = (H \begin{bmatrix} X & 1 \end{bmatrix})^T$. β and C together are referred to as the pose refinement code. Optimizing the pose refinement code together with the shape code results in reconstructed surfaces

that are well-aligned with each other. This is achieved without requiring any additional loss function that aids in the alignment of the poses of objects.

Coarse-to-fine-Approximation Position embeddings improve the performance of coordinate-based neural networks by allowing them to represent higher frequency signals (*i.e.*, more complicated shapes). However, their usage may lead to suboptimal performance on alignment as lower-frequency approximations of signals are better for alignment when there are larger pose differences. For this reason, by first approximating a shape using low-frequency bands of positional encoding and then progressively including higher-frequency bands, a coarse-to-fine approximation of the shape is performed. This process is achieved by applying a smoothing mask on the position encodings. More details can be found in [36].

3.3 Local Surface Modeling

We employ local surface representation obtained from one object category to identify geometrically similar surfaces on novel objects. For training these local surface representations, we utilize object models from a single category, along with a reference frame provided on a representative instance within that category. This reference frame is positioned on the target local surface area. The remaining instances within this category will naturally possess local surfaces that are geometrically similar to the targeted one around this reference frame. These surfaces are then utilized to train the implicit surface model.

We sample points exclusively within a sphere of radius r centered on the reference frame, which restricts the alignment to the designated area within the sphere, excluding any other surface areas. It also has the additional benefit of limiting the network’s capacity spent on surfaces outside of the targeted one. Since the alignment transformation is optimized at the same time with the network parameters, the sampling sphere is dynamically adapted to the current transformation matrix H . This ensures that only the targeted surface is learned during training.

3.4 Grasp Transfer ¹

We associate demonstrated grasps with local surface models, and use these models to identify geometrically similar surfaces on novel objects for grasp transfer. To achieve this, the reference frames for training these models are provided through grasp demonstrations.

We first record an expert grasp demonstration on an object, which we assign as the anchor object (a). Using this expert grasp demonstration, we find a transformation between the world frame (W) and the demonstration frame (D) which will be equal to the transformation between the world frame and the grasp frame (G) for the anchor object $T_{WD} = {}^aT_{WG}$. Using T_{WD} , we transform all object frames to the demonstration frame, which will put their grasp frames close to the vicinity of their current identity pose.

Our method then learns to reconstruct each local surface while simultaneously finding the transformations that align the shapes (i denotes the index of the shape) from the demonstration frame to a common reference frame denoted as the alignment frame (A) ${}^iH = {}^iT_{DA}$. When local surfaces are well-aligned, we assume that the transformation between the alignment frame and the grasp frame will be independent of the object, and can be denoted as T_{AG} . The transformation between the alignment and the grasp frame is found using the alignment transformation of the anchor object $T_{WD}{}^aT_{DA}T_{AG} = {}^aT_{WG} = T_{WD} \implies T_{AG} = {}^aT_{AD}$. For the remaining objects, we estimate the transformation between the world frame and the grasp frame as ${}^iT_{WG} = T_{WD}{}^iT_{DA}T_{AG}$.

After the learning step, for transferring the grasp to new objects, we use the learned implicit representations. During the inference step, the weights of SDF-net and Hypernet are fixed. We pick several candidate frames, with transformations ${}^iT_{WC_j}$ (C_j corresponds to the candidate frame with index j), at locations close to the target surface for the object i with a local surface that is geometrically similar to the originally trained one. We employ these candidate frames on copies of the object to put them in the vicinity of possible grasp locations. We then optimize a shape code along with a pose refinement code for each object copy to identify surfaces similar to the underlying surface type. After the optimization, if the reconstruction error is lower than a given threshold, the pose refinement code corresponding to the candidate j of the

¹We use the transformation notation ${}^oT_{SD}$, where S is the source frame, D is the destination frame, and o is the corresponding object.

object i is used to estimate the transformation matrix ${}^{i,j}H$ which denotes the transformation between the candidate frame and the alignment frame ${}^{i,j}H = {}^iT_{C_jA}$. The transformation between the candidate frame and the alignment frame is employed together with the transformation between the world frame and the candidate frame to identify grasp poses on new objects ${}^iT_{WG} = {}^iT_{WC_j} {}^iT_{C_jA} T_{AG}$.

3.5 Local Surface Prediction on Point clouds

At inference time, we work with point cloud observations and build their corresponding SDF representation as the first step in the local surface prediction stage. Point clouds have SDF values of 0, as the observed points are on the surface of the objects. While these points can be used directly for optimizing shape and pose refinement codes, it often gives sub-optimal results. Instead, we sample points uniformly in the workspace and find their unsigned distance value directly by checking the distance between them and the originally observed point cloud. Using these sampled points in addition to the observed point cloud in optimization yields better shape reconstruction and grasp pose estimation.

4 Experiments

We evaluate our method based on three simulation experiments: shape alignment, grasp alignment and grasp transfer. Furthermore, we validate our method using a real robot, showing that it can transfer grasps to novel real objects, such as the watering pot and the bowl shown in Figure 6, where we use the same models trained with synthetic objects, *e.g.*, mugs.

We use the ShapeNet-Core V2 dataset [40] which has all the shapes in their canonical poses. We employ the same data generation process as DeepSDF [19]; however, as we deal with smaller surface areas, we perturb surface points with a higher variance accordingly (0.0025×15). In all experiments, we use a 5-layer ReLU MLP for SDF-Net and multiple 2-layer ReLU MLPs for Hypernet. All MLPs have a hidden layer size of 256. We use Adam optimizer [41] with a learning rate of 10^{-3} . The loss function we used for training is $L_T = w_1 L_{SDF} + w_2 L_{shape} + w_3 L_{tr}$ in which L_{SDF} is the $L1$ -distance between the predicted and the ground truth SDF values; L_{shape} and L_{tr} are

the $L2$ norm of the shape code, α , and translation vector, t , respectively. At inference time, an additional loss term for scaling code, $L2$ norm of $C - 1$, with weight w_4 is added to constrain it ². Experiments are performed using a system equipped with an Intel i9-11950H processor (2.6 GHz, 8 cores), an Nvidia RTX A3000 GPU and 32 GB of RAM. The training of a single shape model typically takes approximately 25 minutes.

For grasp pose estimation, we train three implicit surface models that represent local surface types that are frequently encountered during two-finger grasping. Each model is trained with local surfaces extracted from 64 object meshes. We employ a single grasp pose demonstration as a reference for the rim and the handle of the anchor mug (orange mug shown in Figure 2) to train the rim and the handle grasp models. We train a cylinder grasp model based on primitive cylinders and cuboids with varying sizes ³. For these objects, the reference grasp pose is given on the sides of cuboids and cylinders, and grasp poses are pre-aligned. To allow the model to represent cuboids and cylinders of different sizes, the pose refinement codes are fixed during the training of this model.

We perform the simulation experiments using a Panda robot. In real-world experiments, our experimental setup consists of a UR10 robotic arm equipped with an RG-2FT gripper. In both cases, we use a Real-Sense D435 camera to acquire the point cloud of the scene. The camera is fixed on the wrist of the robotic arm. In the experiments, all objects are placed in stable standing poses on a table top.

4.1 Shape Alignment

In this experiment, we show that our network can align the poses of objects from a specific category throughout the training. We use 64 shapes from four different object categories, which are mugs, chairs, planes, and cars, to create four datasets. For each shape in the datasets, the poses of the objects are varied by applying random perturbations. We train our network with one object in its canonical pose as the anchor, and the rest of the objects with randomly perturbed positions and orientations. In this experiment, scaling embeddings are fixed as they cannot be used in the evaluation.

²In this work, $w_1 = 3 \times 10^3$, $w_2 = 10^4$, $w_3 = 10^2$, $w_4 = 2.5 \times 10^2$

³Cylinders with diameters $\varnothing = [10, 80]$ cm and heights $h = [25, 100]$ cm, cuboids with side lengths $x = [10, 80]$ cm and heights $h = [25, 100]$ cm.



Figure 3: Example shape alignment results. The meshes on their canonical poses are shaded. Reconstructed meshes are on the right, given perturbed meshes are on the left. Post-training, shapes are aligned very closely to their canonical poses.

Visualizations of example reconstructions are shown in Figure 3. Our method aligns objects very close to their canonical poses. We have evaluated our method’s alignment performance by comparing two different Chamfer distances for all the object categories. The first Chamfer distance is between the perturbed mesh and the ground-truth mesh, and the second is between the reconstructed mesh and the ground-truth mesh. For this, we sample 10,000 points from all meshes (except for the anchor one) and estimate the two-way Chamfer distance. We repeat this experiment five times. The results are shown in Table 1. This table shows that the Chamfer distances for reconstructed meshes are significantly lower than the Chamfer distances for perturbed poses. Another observation is that our method exhibits a higher reconstruction loss and standard deviation for chairs and planes. This disparity is due to significant variations in terms of geometry within these categories, e.g. the chair category not only includes chairs but also sofas. Regardless, even for these categories, our method acquires lower Chamfer distances. Overall, these results show that our approach can align object meshes while learning their implicit representations.

Table 1: Shape Alignment Results

Object Name	Reconstructed Mesh	Pose Perturbed Mesh
Mugs	1.5 ± 0.5	17.5 ± 1.0
Chairs	5.8 ± 3.9	20.7 ± 1.0
Planes	7.8 ± 3.7	25.2 ± 3.9
Cars	0.8 ± 0.3	20.1 ± 2.4

* The error values in the table are divided by 10^{-3} . $\pm$ denotes the standard deviation.

Table 2: Spatial Precision and Grasp Accuracy (%)

ϵ	Ours		NDF [11]		Naive-Transfer	
0°	0.16 ± 0.1	100	0.29 ± 0.2	100	0.18 ± 0.1	100
10°	0.15 ± 0.1	100	0.42 ± 0.2	88	0.30 ± 0.1	100
20°	0.16 ± 0.1	100	0.30 ± 0.1	100	0.52 ± 0.1	100
30°	0.16 ± 0.1	100	0.53 ± 0.8	88	0.76 ± 0.1	0
40°	0.47 ± 0.7	81	0.43 ± 0.2	94	1.00 ± 0.1	0

* $\pm$ denotes standard deviation.

4.2 Spatial Precision of Grasp Alignments

To evaluate the grasp alignment performance of our method, we compare it to a recent baseline, neural descriptor fields (NDF) [11], based on two metrics. The first metric is spatial precision of pre-grasp positions, *i.e.*, the distance between grasps generated by our method and the demonstrated grasp. This metric is calculated based on the Euclidean distances between the predicted and the expert annotated grasp poses. The second metric is grasp success accuracy in simulation via lifting tests by checking whether the object slips or it is dropped. This experiment is conducted on the handle region of the mug objects as this part is unique for mugs ⁴.

We annotate expert grasp locations on 16 mug objects to compare the proposed network with NDF. NDF transfers the grasp to objects of the same

⁴We also tested NDF with the multi-category model. However, the model did not exhibit consistent success in predicting grasp locations correctly. Hence, we limited our baseline comparison to mug objects. Furthermore, we perform this comparison solely on points sampled from the CAD model of an object as the NDF did not work very well with the point cloud acquired from the simulation. This is most likely due to the fact that such point cloud is partial, *i.e.*, some parts of the mug are not visible to the camera.



Figure 4: Comparison of resulting grasp poses. The anchor mug with expert grasp pose is shown with the gray mesh on the left. Grasp poses, represented as fingertip frames, predicted by NDF and our method are shown on the pink and the brown meshes.

category with energy minimization on the latent space of queried positions. We utilize the pose of the same expert grasp demonstration on the anchor object used in training our method as a query to condition the NDF model. For both models, initial grasp candidate locations are generated around the expert grasp pose. We repeat this experiment with four different levels of perturbations applied to object yaw angle: 10° , 20° , 30° , 40° in which we rotate the test objects by the corresponding degree before optimizing the grasp locations. As an additional baseline, we include the naive transfer results where the originally demonstrated grasp on the anchor object is directly transferred without any adjustments. Results of these experiments can be found in Table 2. Results are normalized to have a maximum error of 1. Illustrations of two example grasps predicted by our method and NDF can be found in Figure 4. Table 2 and Figure 4 show that our method predicts grasps more similar to the expert ones until 40° , and acquires better spatial precision than NDF. At 40° , it still predicts good grasps for most of the objects but fails for some of them potentially due to getting stuck in a local minima. Compared to our method, NDF has a smaller error at 40° as it does not need as good initialization as our method does for grasp candidate sampling, however, it overall performs worse than our method. Furthermore, our method leads to an average optimization time of 4.62 seconds for 8 grasp pose candidates while NDF requires 62.2 seconds for the same number of poses. Grasp pose timings for our network grow linearly with the number of grasp candidates.

4.3 Grasp Transfer

In this experiment, we show that the learned implicit representations can be used in identifying grasps on unseen mugs and novel objects from different

Table 3: Grasp Success Rate (%) / Post Grasp Motion (cm)

Object - Part (# of objects)		Ours		NOCS-0°	
Mugs - Handle (16)		100	0.2	100	0.58
Mugs - Rim (16)		100	0.41	100	0.57
Bags - Handle (23)		87	0.2	100	0.73
Bowls - Rim (65)		98	0.66	100	1.06
Earphones - Rim (22)		91	0.69	100	0.62
Bottles - Cylinder (35)		94	0.59	80	0.67

NOCS-20°		NOCS-40°		NOCS-60°		NOCS-80°	
100	0.7	12	1.66	0	<i>NA</i>	0	<i>NA</i>
100	0.66	100	1.32	100	1.94	100	2.3
91	1.02	83	1.57	30	2.67	4	0.25
98	1.09	98	1.14	100	1.14	97	1.07
100	1.63	95	2.84	68	3.53	45	3.48
77	0.76	71	0.65	80	0.99	80	0.62

categories: bags, bowls, earphones, bottles, and cans. For this, each object is placed in a stable pose on a table in PyBullet [42] simulation. A camera mounted on the wrist captures point cloud observations of the object from multiple predetermined viewpoints. These point clouds are then merged and filtered to retain only the points of the object of interest.

For baseline comparison, we use the normalized object coordinate system (NOCS) [21] for grasp pose transfer. In NOCS, all points in the point cloud are normalized uniformly such that each of their x , y , z values are between 0 and 1 with minimum and maximum being 0 and 1. For the baselines, depending on the object type, the grasp pose from the handle, rim, and cylinder categories are defined in NOCS coordinates. For this, expert grasp poses on a single representative instance from each object type are used. NOCS-based grasp pose estimation requires accurate object pose detection. Therefore, we devise five baselines where we assume NOCS methods have access to canonical object poses with yaw angle errors of 0°, 20°, 40°, 60°, and 80°. Our method also has access to the provided grasp pose for each object type for grasp candidate sampling, along with the corresponding surface type, however, it does not have access to the object poses. Specifically, utilizing only the provided object-level grasp pose locations, we sample 12 candidate grasp poses with 30° intervals around the object and optimize their parameters.

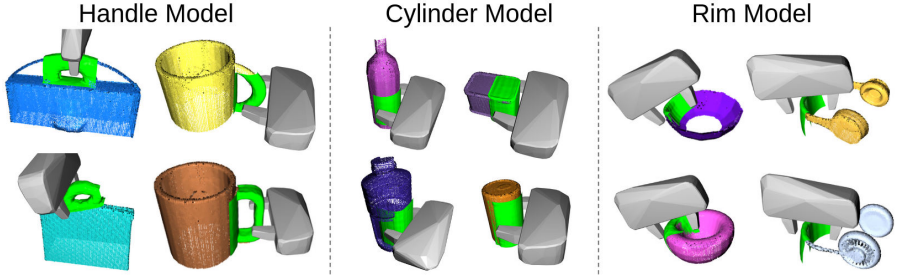


Figure 5: Results of transferring grasps to different objects. Reconstructed surfaces are shown with green meshes.

We run our method and the NOCS baseline on the point clouds of each object for grasp transfer. Visualizations of some of the grasp poses can be seen in Figure 5. The grasp transfer approach is assessed using two criteria. The first criterion is based on post grasp motion, *i.e.*, the Euclidean distance between the object’s position before and after the robot executes its grasping action, prior to lifting the object. The second criterion is based on whether or not the object can be successfully lifted up after the grasp execution.

The grasp accuracy and the post-grasp motion can be found in Table 3. In Table 3, we observe that our method leads to successful grasp transfer results on all experiment objects with similar success rates to NOCS-0°. When the amount of angle error for the NOCS baseline increases, our network outperforms the NOCS baseline, especially on non-symmetric objects. One interesting observation here is that our method always performs better than the NOCS baseline on bottle objects even though they are mostly symmetric. After analyzing grasps on bottle category, we observe that bottle objects in their canonical pose do not consistently have the geometrically shorter side in front. This leads to failure for NOCS-based baselines as they rely on consistency in geometry for objects in their canonical pose. As a result, the NOCS-based methods sometimes grasp bottles from their longer side. Our method however adapts to shape variations and consistently predicts grasps from the shorter side of the bottle. In addition, as our method generates grasp poses based on local surface knowledge, it acquires lower post-grasp motion than the NOCS baselines, even on objects in which pose perturbations do not hinder the grasp success.

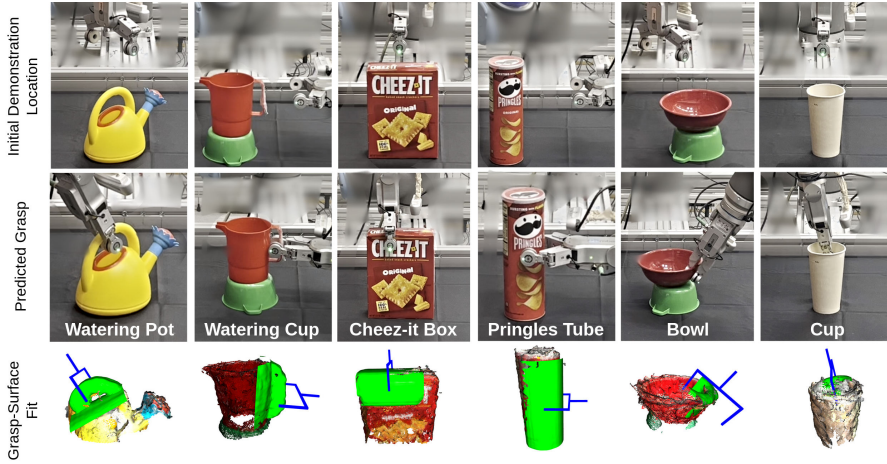


Figure 6: Grasp pose alignments on real-world point clouds. Our method leverages a single rough initial demonstration for each object and finds the correct grasp pose by searching for a grasp-surface fit.

4.4 Real Robot Experiments

In these experiments, we analyze our method’s performance using a real robot composed of a UR10 arm and an OnRobot RG2-FT gripper, and six objects as seen in Figure 6. We demonstrate that our approach can successfully transfer grasp poses using point cloud observations from real objects belonging to unseen categories. This is achieved by utilizing the same models trained with synthetic data. For each object, we initialize our method by providing a single demonstration, which is around the surface area to be grasped for grasp candidate sampling, and the surface type. The robot scans the scene from six predefined camera views and merges the acquired point clouds. The camera views are chosen independently of the object’s poses. Then the initial demonstration is utilized in grasp candidate sampling to find the correct surface-grasp alignment on the merged point cloud. Using this alignment, a grasp is executed. Following these steps, we perform grasp transfer experiments for each object five times using varying object poses to test robustness to pose changes. We use the same initial demonstration for all trials involving each corresponding object. For each grasp execution, scene capture takes 1 minute and grasp pose estimation takes, on average, around 10.5 seconds.

Our method’s grasp predictions on the objects are shown in Figure 6. The success rates for watering pot, watering cup, Cheez-it box, Pringles tube, bowl, and cup are as follows: 5/5, 4/5, 5/5, 5/5, 5/5, 5/5, respectively. Overall, the method demonstrates high success rates in transferring grasps to real novel objects including cases where the objects are partially visible, with only one observed failure. In the failure case, the final transferred grasp pose leads to successful lifting of the object, but it is counted as failure. The actual reason for failure is that the best handle surface fit is found at the rim of the object, not at the handle, which might be due to the combination of the stochasticity in the initial grasp candidate sampling and the point cloud being noisy.

5 Conclusion

This paper presents an implicit representation architecture that aligns local surfaces around an expert grasp demonstration while learning to reconstruct their shapes. This representation is used to transfer grasps to local surfaces that are similar in shape to the surface the expert grasp is demonstrated on. Through both simulation and real-world experiments, we show that we can transfer grasps to new objects with similar local geometry and acquire better spatial precision than a baseline approach.

In future work, we plan to investigate the effects of employing equivariant shape representations [43], [44], keypoint detection methods, and zero-shot category level pose estimators [45] in terms of improving the generation and evaluation of local surface candidates. In addition, we aim to perform faster grasp pose generation and scale our method to multi-object scenarios [46]. We also plan to use multi-fingered hands for a direct mapping between fingertip locations and the local surfaces.

References

- [1] R. Platt, “Grasp learning: Models, methods, and performance,” *Annual Review of Control, Robotics, and Autonomous Systems*, vol. 6, 2022.
- [2] S. Levine, P. Pastor, A. Krizhevsky, et al., “Learning hand-eye coordination for robotic grasping with deep learning and large-scale data collection,” *The International journal of robotics research*, vol. 37, no. 4-5, pp. 421–436, 2018.

-
- [3] 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.
 - [4] C. Eppner, A. Mousavian, and D. Fox, “Acronym: A large-scale grasp dataset based on simulation,” in *2021 IEEE International Conference on Robotics and Automation (ICRA)*, IEEE, 2021, pp. 6222–6227.
 - [5] A. Ten Pas, M. Gualtieri, K. Saenko, and R. Platt, “Grasp pose detection in point clouds,” *The International Journal of Robotics Research*, vol. 36, no. 13-14, pp. 1455–1473, 2017.
 - [6] M. Sundermeyer, A. Mousavian, R. Triebel, and D. Fox, “Contact-graspnet: Efficient 6-dof grasp generation in cluttered scenes,” in *2021 IEEE International Conference on Robotics and Automation (ICRA)*, IEEE, 2021, pp. 13 438–13 444.
 - [7] A. Mousavian, C. Eppner, and D. Fox, “6-dof graspnet: Variational grasp generation for object manipulation,” in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2019, pp. 2901–2910.
 - [8] Z. Jiang, Y. Zhu, M. Svetlik, et al., “Synergies between affordance and geometry: 6-dof grasp detection via implicit representations,” *Robotics: Science and Systems*, 2021.
 - [9] 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.
 - [10] 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.
 - [11] 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.
 - [12] 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.

- [13] S. Schaal, “Is imitation learning the route to humanoid robots?” *Trends in cognitive sciences*, vol. 3, no. 6, pp. 233–242, 1999.
- [14] C. de Farias, N. Marturi, R. Stolkin, and Y. Bekiroglu, “Simultaneous tactile exploration and grasp refinement for unknown objects,” *IEEE Robotics and Automation Letters*, vol. 6, no. 2, pp. 3349–3356, 2021.
- [15] Y. Qin, R. Chen, H. Zhu, et al., “S4g: Amodal single-view single-shot se(3) grasp detection in cluttered scenes,” in *Conference on Robot Learning*, PMLR, 2020, pp. 53–65.
- [16] 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.
- [17] Y. Bekiroglu, D. Song, L. Wang, and D. Kragic, “A probabilistic framework for task-oriented grasp stability assessment,” in *IEEE International Conference on Robotics and Automation (ICRA)*, 2013, pp. 3040–3047.
- [18] Y. Xie, T. Takikawa, S. Saito, et al., “Neural fields in visual computing and beyond,” *Computer Graphics Forum*, 2022, ISSN: 1467-8659.
- [19] 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.
- [20] 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.
- [21] H. Wang, S. Sridhar, J. Huang, et al., “Normalized object coordinate space for category-level 6d object pose and size estimation,” in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2019, pp. 2642–2651.
- [22] C. Wang, R. Martín-Martín, D. Xu, et al., “6-pack: Category-level 6d pose tracker with anchor-based keypoints,” in *2020 IEEE International Conference on Robotics and Automation (ICRA)*, IEEE, 2020, pp. 10 059–10 066.

-
- [23] 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.
 - [24] 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.
 - [25] F. T. Pokorný, K. Hang, and D. Kragic, “Grasp moduli spaces,” in *Robotics: Science and Systems*, 2013.
 - [26] F. T. Pokorný, 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.
 - [27] A. Myers, C. L. Teo, C. Fermüller, and Y. Aloimonos, “Affordance detection of tool parts from geometric features,” in *2015 IEEE International Conference on Robotics and Automation (ICRA)*, IEEE, 2015, pp. 1374–1381.
 - [28] O. Kroemer, E. Ugur, E. Oztop, and J. Peters, “A kernel-based approach to direct action perception,” in *2012 IEEE international Conference on Robotics and Automation*, IEEE, 2012, pp. 2605–2610.
 - [29] 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.
 - [30] 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.
 - [31] B. Wen, W. Lian, K. Bekris, and S. Schaal, “Catgrasp: Learning category-level task-relevant grasping in clutter from simulation,” in *2022 International Conference on Robotics and Automation (ICRA)*, IEEE, 2022, pp. 6401–6408.
 - [32] 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.

- [33] 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.
- [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] L. Yen-Chen, P. Florence, J. T. Barron, et al., “Inerf: Inverting neural radiance fields for pose estimation,” in *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, IEEE, 2021, pp. 1323–1330.
- [36] 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.
- [37] D. Ha, A. M. Dai, and Q. V. Le, “Hypernetworks,” in *International Conference on Learning Representations*, 2017.
- [38] 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.
- [39] J.-L. Blanco, “A tutorial on se (3) transformation parameterizations and on-manifold optimization,” *University of Malaga, Tech. Rep.*, vol. 3, p. 6, 2010.
- [40] A. X. Chang, T. Funkhouser, L. Guibas, et al., “ShapeNet: An Information-Rich 3D Model Repository,” Stanford University — Princeton University — Toyota Technological Institute at Chicago, Tech. Rep. arXiv: 1512.03012 [cs.GR], 2015.
- [41] D. P. Kingma and J. Ba, “Adam: A method for stochastic optimization,” in *International Conference on Learning Representations*, 2015.
- [42] E. Coumans and Y. Bai, *Pybullet, a python module for physics simulation for games, robotics and machine learning*, 2016.
- [43] X. Li et al., “Leveraging se (3) equivariance for self-supervised category-level object pose estimation from point clouds,” *Advances in neural information processing systems*, vol. 34, pp. 15 370–15 381, 2021.

- [44] R. Sajnani, A. Poulenard, J. Jain, R. Dua, L. J. Guibas, and S. Sridhar, “Condor: Self-supervised canonicalization of 3d pose for partial shapes,” in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2022, pp. 16 969–16 979.
- [45] W. Goodwin, S. Vaze, I. Havoutis, and I. Posner, “Zero-shot category-level object pose estimation,” in *European Conference on Computer Vision*, Springer, 2022, pp. 516–532.
- [46] A. Murali, A. Mousavian, C. Eppner, et al., “6-dof grasping for target-driven object manipulation in clutter,” in *IEEE International Conference on Robotics and Automation (ICRA)*, 2020, pp. 6232–6238.

PAPER **B**

**Neural Field Movement Primitives for Joint Modelling of Scenes
and Motions**

Ahmet Ercan Tekden, Marc Peter Deisenroth, Yasemin Bekiroglu

*IEEE/RSJ International Conference on Intelligent Robots and Systems
(IROS), 2023, Detroit, USA.*

©2023 IEEE

The layout has been revised.

Abstract

This paper presents a novel Learning from Demonstration (LfD) method that uses neural fields to learn new skills efficiently and accurately. It achieves this by utilizing a shared embedding to learn both scene and motion representations in a generative way. Our method smoothly maps each expert demonstration to a scene-motion embedding and learns to model them without requiring hand-crafted task parameters or large datasets. It achieves data efficiency by enforcing scene and motion generation to be smooth with respect to changes in the embedding space. At inference time, our method can retrieve scene-motion embeddings using test time optimization, and generate precise motion trajectories for novel scenes. The proposed method is versatile and can employ images, 3D shapes, and any other scene representations that can be modeled using neural fields. Additionally, it can generate both end-effector positions and joint angle-based trajectories. Our method is evaluated on tasks that require accurate motion trajectory generation, where the underlying task parametrization is based on object positions and geometric scene changes. Experimental results demonstrate that the proposed method outperforms the baseline approaches and generalizes to novel scenes. Furthermore, in real-world experiments, we show that our method can successfully model multi-valued trajectories, it is robust to the distractor objects introduced at inference time, and it can generate 6D motions.

1 Introduction

Learning from demonstration (LfD) methods enable robots to bootstrap the acquisition of new skills necessary to complete tasks by leveraging expert-provided demonstrations [1]. These methods are useful in tasks where modeling the motion with respect to scene variations is difficult. When the task is complex, and the associated scene can only be described using highly nonlinear task parameters, LfD can facilitate the acquisition of new skills. It offers

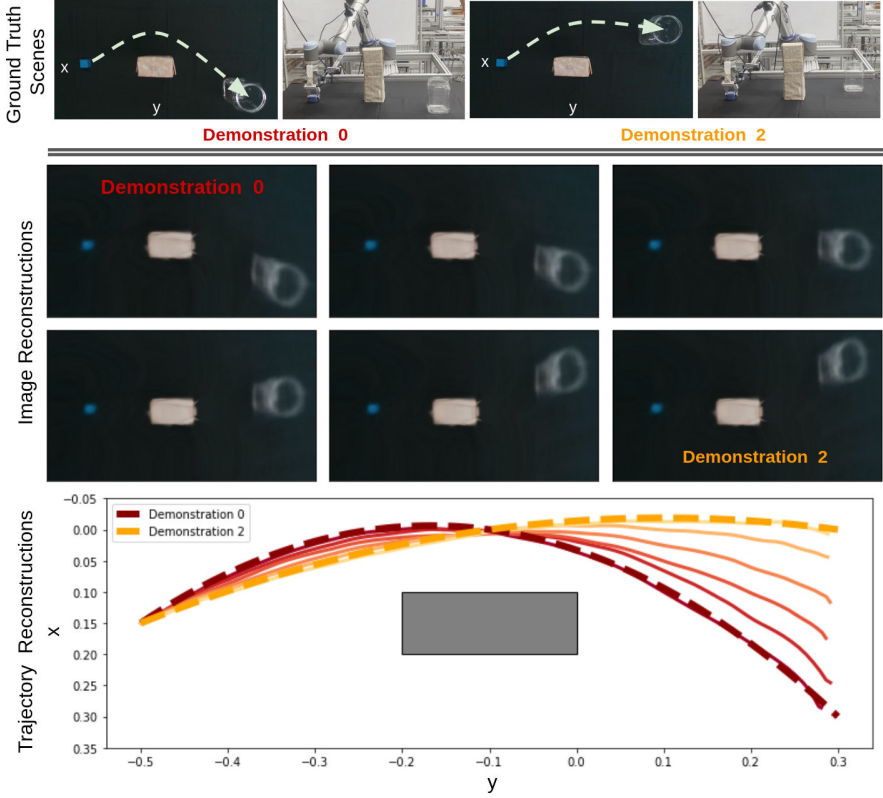


Figure 1: Examples illustrating the approach: Two expert demonstrations for two different plastic container locations are shown on top. Our network can smoothly and accurately interpolate between these demonstrations, generating the images and the trajectories corresponding to the unseen intermediate plastic container locations.

insights into how the skill is affected by changes in the task parameters and enables easier generalization to previously unseen scene conditions.

Movement primitives [2], [3], [4], [5], [6], [7] are often built upon LfD approaches that provide an efficient way to teach robots complex motor skills. In LfD, data efficiency corresponds to the number of expert demonstrations required to model the motion correctly. These methods [4], [5], [6], [7] generally employ hand-crafted task parameters that well describe the scene to learn skills that generalize to novel scenes in a data-efficient way. Common examples of these task parameters are waypoints, obstacle locations, and sizes. Often, these task parameters have to be provided by the user, or they need to be estimated, which requires further engineering effort by experts that have access to clear knowledge of the tasks, and know the limitations of the underlying methods. To reduce the dependence on experts, some movement primitive architectures use raw images [4], [8] to extract the factors that influence the skill. However, they usually require a thousand [4], or a few thousand [8] demonstrations to achieve this. We propose an alternative approach: modeling the scenes and the motions using neural fields. Recently, it has been shown that neural fields [9] can be utilized to represent 3D scenes [10], shapes [11], images [12] using small amount of data. Inspired by this, we propose an LfD method that jointly models the scenes and the motions using neural fields. Our method¹ learns to reconstruct the motion trajectories and the scenes given corresponding expert demonstrations of the task using neural fields. Later, it can generate accurate motion trajectories for novel scenes. An example generation of our method is shown in Figure 1. In this figure, our method interpolates between two expert demonstrations with different container locations and accurately generates the scene and the motion representations for unseen container locations. Unlike the previous works, our method achieves such trajectory generation without access to task parameter vectors at training or test time. Furthermore, it uses a similar number of expert demonstrations that the task parameter-based LfD methods use.

Neural fields [9] are powerful scene representation tools that can be employed to model images, surfaces, and 3D scenes. Using them allows for representing scenes in a format ideal for the task and provides a versatile way of modeling movement primitives. Neural fields can be equipped to handle modeling smooth changes in the scene by using deformation and template

¹Project page: <https://fzaero.github.io/NFMP/>

neural fields. Using deformation fields, our method maps smooth scene and motion changes to each other, enabling accurate motion trajectory generation for novel scenes. However, when there is a large change between samples, deformation fields fail to model changes in the scene correctly (*i.e.*, rapid scene changes [13]). This is because neural fields model complex signals through positional encodings [14]: Low-frequency positional encodings are good at modeling smooth changes, while high-frequency positional encodings are good at capturing fine details. Using low-frequency positional encodings results in overly smooth scene reconstruction. However, with high-frequency positional encodings, the model starts overfitting to high-frequency details in low-sample density settings and loses the generalization capacity. To prevent this from happening, we propose using low-frequency positional encodings for the deformation field while keeping the template field the same. This allows our network to capture the details necessary to represent the scene while still preserving the smoothness that low-frequency positional encodings provide.

Our main contributions can be summarized as follows:

- We propose a novel LfD method that simultaneously generates a scene and a motion representation by utilizing a shared embedding. Our method assigns an embedding vector to each expert demonstration and learns scene and motion neural fields smoothly parametrized with the embeddings. On novel scenes, by employing test time optimization to find an embedding that can correctly generate the scene, our approach can generalize to new scenarios and accurately predict the correct motion.
- Our method preserves the smooth changes in the scene by using low-frequency positional encodings for the deformation field. This allows our approach to be data-efficient while still capturing the necessary details to represent the scene. For all the experiments, the method generalizes to the novel scenes based on low number of demonstrations (3^n demonstration for n independent task parameters).
- We evaluate our approach both in simulation and real-world experiments. Simulation results show that our method outperforms the baseline movement primitive networks on the tasks where the underlying task parametrization is based on object positions and geometric scene changes. Our real-world experiments demonstrate that our method can

handle multi-valued trajectories through implicit modeling of the motion. In addition, our method is robust to distractor objects introduced to the scene at the inference time, and it can model 6D pick and place tasks using a surface-based representation.

2 Related Work

Learning from Demonstration (LfD) is a widely used approach in robot learning that leverages expert demonstrations to bootstrap the learning process [1], [15]. LfD [16] can be employed to encode continuous joint or state trajectories, *e.g.* movement primitives [2], [3], [4], [5], [7]; experience abstractions, *e.g.* behavioral cloning [17], [18], offline RL [19]; and task waypoints, *e.g.* affordance heatmaps [20], keypoints [21], [22], [23]. Our work is related to the first category, where the underlying robotic skills are encoded as low-level motion trajectories using neural fields. Methods in this category often use task parameters that describe the scene compactly to reduce the number of trajectories necessary to generalize to the variations in the scene [24]. However, this requires accurate extraction of task parameters from the scene both during the training and the test time. This can be challenging as it requires developing sophisticated techniques to estimate the task parameters for each different task. In practice, it is often handled by direct provision of the task parameters by a human operator. In contrast, our network directly works with any scene representation that can be generated through neural fields and is validated with RGB images and signed distance functions (SDF) [11]. Furthermore, by modeling the motions and the scenes through a shared embedding, our method learns movement primitives that smoothly generalize to changes in the task parameters. This allows our method to learn new skills with a similar number of demonstrations that previous task-parameter-based movement primitive methods use.

For representing scenes and motions, we use neural fields. Neural Fields [9] are coordinate-based neural networks: They take continuous spatial or temporal coordinates as input to parametrize functions of such coordinates. They allow for a representation of a wide variety of scene modalities, including images [12], implicit surfaces [11], [25], 3D scenes [10], point clouds [26], tactile readings [27], and even audio [12]. Such representations have been used for SLAM [28], navigation [29], control [30], and affordance estimation [31]. Un-

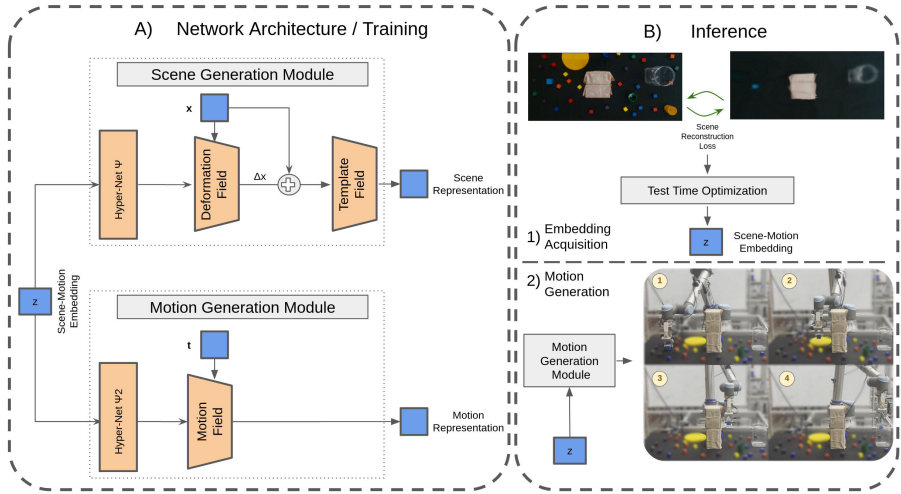


Figure 2: Our network maps scene-motion embeddings to scene representations, *e.g.* RGB images or signed distance functions (SDF), and motion representations, such as joint angles or end-effector position of the robot (A). At inference time, through test-time optimization, scene-motion embeddings can be obtained through scene reconstruction loss (B-1) and can be used to reproduce trajectories for new scenes (B-2).

like these works, our method utilizes neural fields to generate scenes and their corresponding motion trajectories.

Neural fields have also been used to map scenes to grasp locations [32] or grasp trajectories [33] using implicit functions. However, these networks employ large datasets that can only be generated via simulations. They learn the grasp trajectories from millions of samples. In addition, designing a new environment for each unique task and bridging the simulation-to-reality gap are essential requirements for such methods. Instead, our method requires only a handful of demonstrations to model each new task by mapping scene changes to the motion. It can directly learn from real-world data, and it can still generate accurate motion trajectories. Other neural field-based approaches aim to model the scene changes smoothly through deformation fields [13], [34]. However, these networks attain smoothness through denser data sampling. Collecting such densely sampled data in LfD settings is challenging as each scene variation requires an expert demonstration. Instead, we use low-

frequency positional encodings to model smooth changes in a data-efficient way.

3 Method

The overall framework is illustrated in Figure 2. Our network does not have traditional inputs and outputs. Instead, it assigns a scene-motion embedding for each expert demonstration. At training time, our method jointly learns to render the scene and reconstruct the motion for each provided demonstration, while optimizing their shared scene-motion embeddings (A). Our method can employ a wide range of scene and motion modalities that neural fields can model as long as their underlying representations are smooth. At inference time, for a given scene, the scene-motion embedding is first found through test-time optimization that minimizes the scene reconstruction loss (B-1). Later, this embedding is used to generate the corresponding motion trajectory (B-2). Section 5 provides example tasks modeled using our method.

3.1 Problem Formulation

We are interested in LfD tasks parametrized with task parameter vector p where the scene is represented with the function $f_p(x)$ with spatial coordinate $x \in \mathbb{R}^m$, and motion is represented with function $g_p(t)$ with normalized temporal coordinates $t \in [0, 1]$. For these LfD tasks, $f_p(x)$ and $g_p(t)$ are smooth functions of p ².

3.2 Neural Fields

Neural fields are coordinate-based neural networks that take spatial or/and temporal coordinates and parametrize physical properties of the scene, objects, or motions. Scene and motion functions, $f_p(x)$ and $g_p(t)$, can be modeled as neural fields parameterized with task parameter vector p . Neural fields can model a wide range of scene and motion representations:

Images Images are common visual representations in robotics acquired using a wrist or a scene camera. Using coordinate-based MLPs, a mapping, $\mathbb{R}^2 \rightarrow$

²This is generally assumed in most LfD tasks.

$\mathbb{R}^3$, between pixel locations and RGB values is learned to represent the scenes.

Surfaces 3D surfaces are represented with signed distance functions (SDFs) $\mathbb{R}^3 \rightarrow \mathbb{R}^1$, which map 3D locations onto their corresponding signed distance value [11]. For a given point x , SDF yields the distance between the underlying surface and x , which is defined as $x < 0$ if x is inside the surface, and $x > 0$ if x is outside of the surface. Through the marching cubes algorithm [35], the 3D mesh of the scene is recovered using the signed distance values obtained from the SDF.

Motion Trajectory The motion trajectory is the sequence of states of end effector poses or joint angles. Using normalized temporal coordinates, a mapping, $\mathbb{R}^1 \rightarrow \mathbb{R}^n$, between time t and n -dimensional joint/pose state q is employed to model the motion. In the case of multi-valued trajectories, motion is represented implicitly with a mapping $\mathbb{R}^{n+1} \rightarrow \mathbb{R}^1$, where (q, t) coordinates are mapped to state or cost values [18].

In our framework, we adapt SIREN [12] neural field architecture, and model the neural fields with multi-layer perceptron (MLP) parametrized using a Hypernet [36].

3.3 Positional Encodings

Using spatial or temporal coordinates directly in neural networks leads to blurry reconstructions [10]. To prevent this, positional encodings [14] are utilized. Specifically, for a coordinate $x \in \mathbb{R}^m$, we use the positional encoding map $\Gamma(x) = [x, \Gamma_0(x), \Gamma_1(x), \dots, \Gamma_{L-1}(x)] \in \mathbb{R}^{m+2mL}$, where $\Gamma_n(x) = [\cos(2^n \pi x), \sin(2^n \pi x)] \in \mathbb{R}^{2m}$ and L is the number of frequency bands. Higher frequency bands are useful for representing high-frequency signals, such as fine details. In contrast, lower frequency bands are better at capturing low-frequency signals, such as the general silhouette of the scene.

3.4 Scene-motion Embeddings

Obtaining task parameters that describe the scene well is often challenging, even for the initially provided demonstrations. Instead, we assign a scene-motion embedding $z \in \mathbb{R}^k$ to each demonstration with an unknown task parameter vector p . We then represent the scene and the motion as smooth

functions of embedding z . With this new formulation, tasks are equivalently represented with neural fields $F_z(x)$ and $G_z(t)$ parameterized by z embedding.

$$(F_z(x), G_z(t)) \equiv (f_p(x), g_p(t)) \quad (\text{B.1})$$

In LfD, the number of available demonstrations is often small. This makes it challenging for an encoder network to learn a smooth mapping between the input scenes and the latent space for z . As a result, even slight changes in the scene may cause large changes in the motion at inference time, potentially resulting in incorrect motion predictions. Instead, we directly optimize the z embeddings for each expert demonstration alongside the scene and motion neural fields. Test time optimization is utilized at inference time to predict the correct z embedding for new scenes.

3.5 Deformation and Template Fields

In our framework, we assume the scenes are smooth functions of the task parameters, and the changes between the scenes can be expressed using deformations. With this assumption, we model the scene using a template neural field that models the canonical scene $T(x)$ and its deformation with a deformation field $D_z(x) = \Delta x$:

$$F_z(x) = T(x + D_z(x)). \quad (\text{B.2})$$

As the template field corresponds to the canonical scene and is not parameterized with z , it is modeled using a standalone MLP.

The deformation field causes an under-constrained optimization problem where different deformations could result in the same scene reconstruction. Increasing the sampling density of the scene with respect to the task parameters could alleviate this issue to some extent. However, as we would like to minimize the number of provided expert demonstrations, we rely on the following two approaches to address this problem:

Coarse-to-fine Approximation In scene reconstruction with neural fields, using a small number of frequency bands admits low-resolution reconstruction, while a higher number of bands admit high-resolution reconstruction. In low-resolution reconstruction, it is easier to model larger changes in the scene; in high-resolution reconstruction, it is easier to model small scene changes. To

balance this tradeoff, we use a coarse-to-fine approximation where the network first uses a low number of frequency bands, and the higher frequency bands are gradually introduced. This is achieved by applying a smoothing mask to the positional encodings. We adopt the coarse-to-fine approximation method proposed in the BARF [37].

Deformation Fields with a Lower number of frequency bands In our work, we model the template field using a high number of frequency bands and the deformation field using a low number of frequency bands. This allows our network to capture the details necessary to represent the scene while preserving the smoothness provided by the low-frequency positional encodings.

3.6 Test Time Optimization

We estimate the z embedding through test time optimization. Given the scene image or the signed distance values of the modelled surface, we can find the embedding z that minimizes the scene reconstruction loss

$$\operatorname{argmin}_z L_{\text{scene}}(F_z, X) = \sum_{(x_j, s_j) \in X} \|F_z(x_j) - s_j\|_2^2 \quad (\text{B.3})$$

where s_j corresponds to the ground truth scene signal at position x_j for scene samples X . Ideally, the neural networks should establish a one-to-one mapping between the embeddings and the scene-motion functions. However, the network might learn embeddings that correspond to the same scene but with different motions. To mitigate this, we constrain the search space to the convex hull of embeddings utilized in the training process. If there are n demonstrations in the training set, the optimization problem becomes

$$\operatorname{argmin}_\alpha L_{\text{scene}}(F_z, X) \text{ where } z = \sum_{i=0}^{n-1} \alpha_i z_i \quad (\text{B.4})$$

where $\alpha = [\alpha_0, \dots, \alpha_{n-1}]$ and $\sum_{i=0}^{n-1} \alpha_i = 1$; $\alpha_i \geq 0$. 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.

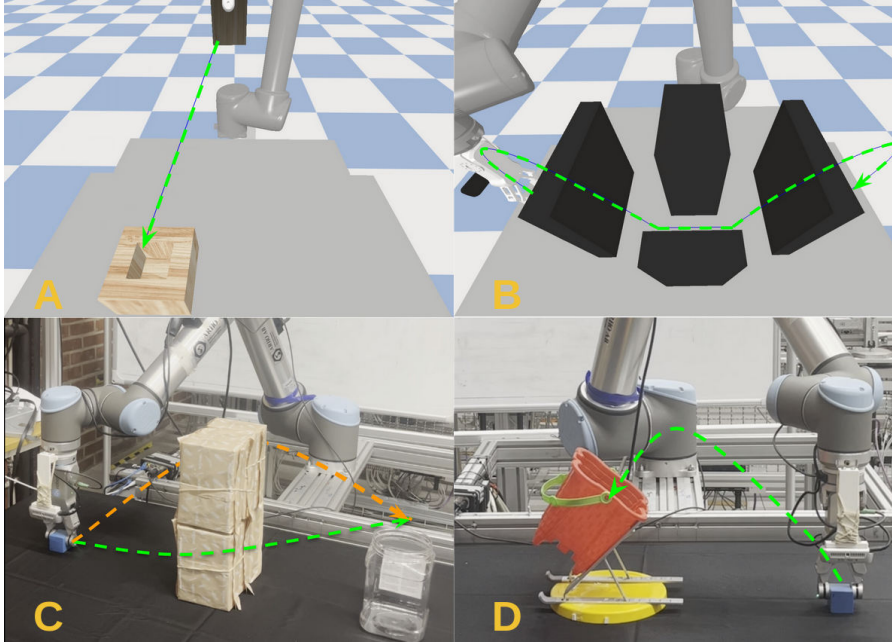


Figure 3: Tasks and the trajectory visualization of the expert demonstrations. Experiments for tasks A and B are conducted in simulation while others are conducted in real-world. For task C, each scene has two alternative motions shown with the color orange and light green.

4 Implementation Details

For all neural fields, a 3-layer Relu MLP with 128 hidden layer sizes and for Hypernets, a 2-layer Relu MLP with 64 hidden layer sizes are used. For template and motion fields, positional encodings with 8 frequency bands and for the deformation field, positional encoding with 2 frequency bands are employed. All scene-motion embeddings used in our experiments have a vector size of 128. All experiments are conducted with a UR10 robot equipped with an RG2 gripper and D435i wrist camera. Pybullet [38] simulator is used for simulation experiments.

We initialize the z embedding for each demonstration as zero vectors. We use the learning rate 10^{-3} for network parameters, and 10^{-4} for embeddings

during training. We use L2 loss between the predicted and the ground truth signal values for motion and scene reconstruction, and the square of the Euclidean norm of embeddings as regularization loss. To prevent learning unnecessary deformation mappings between background pixels, we utilize the square of the Euclidean norm of deformations. In summary, the loss function during training is formulated as:

$$w_1 L_{\text{motion}} + w_2 L_{\text{scene}} + w_3 L_{\text{deform_reg}} + w_4 L_{\text{embedding_reg}}$$

where w are weights for each loss term. Values for w_1 and w_2 are chosen such that they balance each other during training. For w_3 and w_4 , we choose the highest values that do not hinder the performance of the scene and motion reconstruction. All network parameters are fixed at the test time, and we optimize the α vector with a learning rate 0.05 for each test demonstration only from the scene reconstruction loss L_{scene} . Adam [39] optimizer is used both for training and test time optimization.

5 Experiments

To demonstrate the capabilities of our network, we design two simulation and two real-world robotic experiments. Visualizations of the experiment setups along with the demonstrated trajectories are shown in Figure 3 and their details can be found in Table 1. Furthermore, Figure 4 shows the task parameterization. These experiments demonstrate that our framework works on RGB and SDF-based scene representations and position- and joint-based motion representations.

For a baseline comparison, we use two versions of a recent neural network-based motion primitive architecture, CNMP [4]. These baselines are task parameter CNMP (TP-CNMP), which has direct access to task parameters, and C-CNMP, which processes the image of the scene via a CNN encoder to predict the motion trajectory. The network proposed in this paper outperforms the baseline architecture on generalization to shape and position in simulation experiments. In addition, as supported by the real robot experiments, the network can model multi-valued trajectories using implicit motion representation and enables 6D pick-and-place through 3D shape-based generalization.

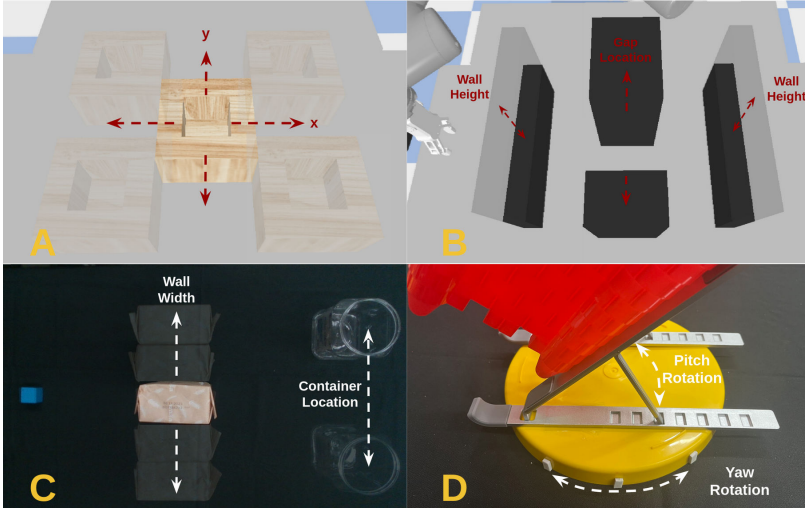


Figure 4: Illustration of how tasks are parameterized. Each arrow shows the extent and the boundary of each task parameter. Expert demonstrations for each task are collected from the boundary and the center locations of the task parameters. The generalization is tested on the interpolation region of these task parameters.

Table 1: Task Details

Task Name	Scene Modality	Motion Modality
Peg in hole	RGB Image	Joint Angles
Wall Avoidance 1	RGB Image	Joint Angles
Wall Avoidance 2	RGB Image	Multi-valued Position
6D Pick and Place	Object Surface	Joint Angles

5.1 Shape and Position Based Generalization

To test our model’s generalization capacity, we design two simulation experiments: based on position, and shape generalization. For both experiments, a scene camera is placed on the opposite side of the robot so that it can well capture the possible changes in the scene.

For the position-based generalization, we set up a peg-in-hole experiment. This experiment contains a box with a peg slot put on a table at a given



Figure 5: Comparison between reconstructed images and ground-truth images. Images with a red border are not seen during the training. However, the scene neural field can smoothly interpolate between the images seen in training and can reconstruct the unseen ones.

(x, y) location. The task for the robot is to insert the peg into the slot on the box while generalizing to its location. A trajectory for the task is shown in Figure 3A. The task parameters for this task are the x and y location of the box. For the shape-based generalization, we set up a wall avoidance experiment with three walls. How this task is parameterized, and the trajectory that robot follows with its end effector is shown in Figure 4B, and Figure 3B. This task can be expressed most compactly with three task parameters: the first wall’s height, the gap location on the second wall, and the third wall’s height.

For both experiments, joint angle-based motion representations, and RGB image-based scene representations are used. For generating the training data-

Table 2: Generalization Results (Average Joint Angle Error in *deg*)

Test Case	Our Method	TP-CNMP	C-CNMP
3x3	0.94	1.01	1.24
4x4	2.40	2.55	3.03
5x5	2.28	2.60	3.04
3x3x3	1.36	3.87	2.28
4x4x4	2.13	3.75	3.90
5x5x5	2.27	3.76	4.13

sets, expert demonstrations and the corresponding scene images are collected from the boundary and the center locations of the task parameters. These datasets are named 3x3 and 3x3x3 for peg-in-hole and wall-avoidance experiments, respectively, and contain 9 and 27 expert demonstrations. For evaluating our network, test datasets using mesh grids of the task parameters with grid sizes 4 and 5 are collected. These datasets contain data from the interpolation region of training demonstrations, and they aim to show how much our network can generalize to novel scenes. These datasets are named based on their grid configurations: 4x4, and 5x5 with 16 and 25 trajectories, respectively, for the peg-in-hole experiment, and 4x4x4 and 5x5x5 with 64 and 125 trajectories, respectively, for the wall-avoidance experiment.

Our comparison with the baseline architecture can be found in Table 2. Our network outperforms both baselines in all cases. Task-parameter-based CNMP performs better than CNN-based one. However, even when the CNMP uses the task parameters directly, our method outperforms it, especially for the wall experiment, which involves more complex motion generation. This is because positional encoding not only improves the scene generation but also enhances the motion generation (*i.e.*, without positional encoding, neural network generates overly-smooth trajectories). To illustrate our network’s smooth interpolation capability, we further show our model’s scene generation results from 5x5 and 5x5x5 datasets in Figure 5. In the first and third rows, data points seen in training are shown with a green border, while unseen ones are shown with a red border. In the second and fourth rows, it can be seen that our network can successfully model shape and position changes between samples correctly.

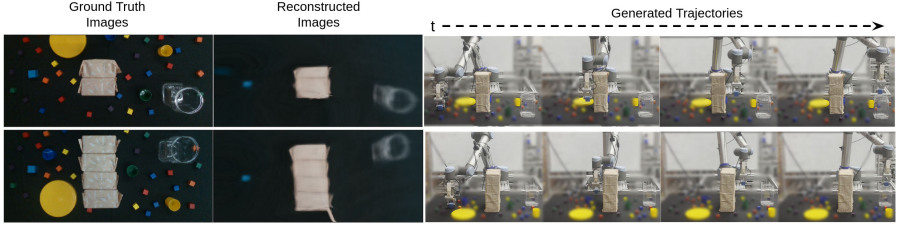


Figure 6: Illustration on the robustness of our method against distractor objects. Our method successfully ignores the distractor objects in its image reconstructions and can generate accurate motion trajectories for unseen environments.

5.2 Multi-valued Trajectory Generation

This experiment shows that our method can model multi-valued trajectories by using implicit motion representations. The experiment includes several objects and obstacles: a blue cube on the left side of the table, a wall made of boxes in the middle, and a plastic container on the other side of the table. The task for the robot is to pick up the blue cube and drop it into the plastic container while avoiding the wall. However, this task can be achieved in two different ways, as illustrated in Figure 3C. The task is parameterized with the number of boxes in the wall, and the y location of the plastic container. The task is modeled using RGB images and position trajectories. Training data is collected using walls with one, three, and five boxes and a plastic container with three different locations, where $y = \{-0.15, 0, 0.15\}$. For each scene, one trajectory that goes from the left and one trajectory that goes from the right are recorded. In total, 18 demonstrations are collected.

Since there are two alternative trajectories for each scene, we model the motion implicitly. This is achieved as follows: for a given 3D location in the scene, if it is on one of the ground truth trajectories, the cost value corresponds to 0; otherwise, its cost value is equal to its distance to the ground truth trajectories (for the distance calculation, the time coordinate is also included). However, implicit trajectory representations lack the same level of smoothness as explicit representations. Therefore, we similarly adopt deformation fields for representing the implicit motion.

Since implicit motion only provides the cost function, the optimal trajectory

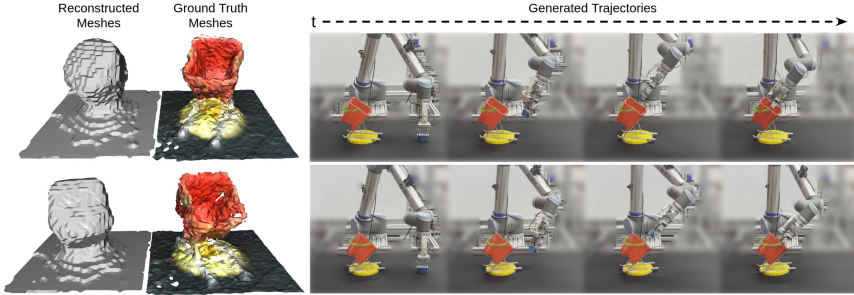


Figure 7: Illustration on how our method can utilize surface-based representations. While our method fails to model the opening of the box in its shape reconstruction due to the quality of the modeled mesh, it still manages to model the rotation of the box correctly. This allowed our method to generalize to novel scenes, and generate accurate motion trajectories for them.

is acquired by optimizing the following term:

$$\operatorname{argmin}_q \int_{t=0}^1 G_z(q_t, t).$$

While previous movement primitive methods [3], [4] require conditioning to handle multi-valued trajectories, our method can directly generate optimal trajectories without such conditioning. Nevertheless, if required, soft conditioning of the trajectory can be achieved by initializing q close to the desired trajectory mode.

We evaluate our method with 12 unseen environments, with walls made of two and four boxes, and novel plastic container locations. To further test our method’s robustness, distractor objects are added to the scene. These distractor objects and how our method ignores them in its scene reconstructions and successfully generates motion trajectories are shown in Figure 6. For these 12 unseen environments, our method accurately generates correct trajectories and acquires 100 % accuracy. Furthermore, Figure 1 illustrates the smooth interpolation of scenes and trajectories between demonstrations 0 and 2. Overall, our method can accurately model this task.

5.3 Shape-based motion generation

In this experiment, we show that our model can exploit surface-based 3D shape representations, which do not have a structured rasterized format, and enable performing a 6D pick-and-place task.

For this experiment, the task is to pick up the blue cube and place it in the box as shown in Figure 3D. This box is placed with yaw and pitch rotations shown in Figure 4D. As this box is angled, the robot needs to adapt its trajectory according to the rotation of the box. Moreover, the opening of the box is narrow, and the task requires around 2 cm precision to prevent the robot from colliding with the sides of the box.

The task is modeled using signed distance values and joint angle trajectories. For acquiring the ground truth scene representation, a set of points with their corresponding signed distance values are sampled from the mesh of the box estimated by recording the scene from 8 pre-defined camera poses. For this process, open3D [40] and pointcloud-utils [41] libraries are employed. The training dataset is collected on scenes with yaw rotations angles of -30° , 0° , and 30° and pitch rotations corresponding to setting the stand on the first, third, and fifth gaps.

Similarly to our previous experiment, we evaluate our method’s performance on 12 unseen environments with novel yaw and pitch rotations. Our method generates accurate trajectories for each scene and manages to acquire 100 % accuracy. However, since the estimated mesh is not watertight, meshes reconstructed using our method cannot model the opening of the box. Yet, since it can accurately model the rotation of the box, it still generates correct trajectories. Examples of estimated and reconstructed meshes, and their corresponding motion trajectories are shown in Figure 7.

6 Conclusion

In this work, we present a novel LfD method that can simultaneously model the scene and the motion for a task. Our experiments show that our method can generalize to novel scenes by accurately generating their motion trajectories and reconstructing their underlying scene representations. In addition, the proposed approach outperforms the baseline methods, can use multiple different scene modalities, such as RGB images and SDF, can model multi-valued trajectories, and is robust to disturbances. In future work, we plan to increase

our method’s data efficiency further and enable effect-based conditioning for trajectory generation. Furthermore, we plan to validate our method in a wider variety of task settings, including dexterous manipulation.

References

- [1] 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.
- [2] 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.
- [3] A. Paraschos, C. Daniel, J. R. Peters, and G. Neumann, “Probabilistic movement primitives,” *Advances in neural information processing systems*, vol. 26, 2013.
- [4] M. Y. Seker, M. Imre, J. H. Piater, and E. Ugur, “Conditional neural movement primitives,” in *Robotics: Science and Systems*, vol. 10, 2019.
- [5] 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.
- [6] M. Akbulut, E. Oztop, M. Y. Seker, et al., “Acnmp: Skill transfer and task extrapolation through learning from demonstration and reinforcement learning via representation sharing,” in *Conference on Robot Learning*, PMLR, 2021, pp. 1896–1907.
- [7] S. Calinon, “A tutorial on task-parameterized movement learning and retrieval,” *Intelligent service robotics*, vol. 9, pp. 1–29, 2016.
- [8] A. Gams, A. Ude, J. Morimoto, et al., “Deep encoder-decoder networks for mapping raw images to dynamic movement primitives,” in *2018 IEEE International Conference on Robotics and Automation (ICRA)*, IEEE, 2018, pp. 5863–5868.
- [9] Y. Xie, T. Takikawa, S. Saito, et al., “Neural fields in visual computing and beyond,” *Computer Graphics Forum*, 2022, ISSN: 1467-8659.

- [10] 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.
- [11] 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.
- [12] 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.
- [13] 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.
- [14] 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.
- [15] S. Schaal, “Learning from demonstration,” *Advances in neural information processing systems*, vol. 9, 1996.
- [16] 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.
- [17] 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.
- [18] P. Florence, C. Lynch, A. Zeng, et al., “Implicit behavioral cloning,” in *Conference on Robot Learning*, PMLR, 2022, pp. 158–168.
- [19] S. Levine, A. Kumar, G. Tucker, and J. Fu, “Offline reinforcement learning: Tutorial, review, and perspectives on open problems,” *arXiv preprint arXiv:2005.01643*, 2020.
- [20] 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.

-
- [21] 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.
 - [22] 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.
 - [23] 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.
 - [24] H. Ravichandar, A. S. Polydoros, S. Chernova, and A. Billard, “Recent advances in robot learning from demonstration,” *Annual review of control, robotics, and autonomous systems*, vol. 3, pp. 297–330, 2020.
 - [25] 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.
 - [26] Q. Xu, Z. Xu, J. Philip, et al., “Point-nerf: Point-based neural radiance fields,” in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2022, pp. 5438–5448.
 - [27] Y. Wi, P. Florence, A. Zeng, and N. Fazeli, “VirDo: Visio-tactile implicit representations of deformable objects,” in *2022 International Conference on Robotics and Automation (ICRA)*, IEEE, 2022, pp. 3583–3590.
 - [28] E. Sucar, S. Liu, J. Ortiz, and A. J. Davison, “Imap: Implicit mapping and positioning in real-time,” in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2021, pp. 6229–6238.
 - [29] M. Adamkiewicz, T. Chen, A. Caccavale, et al., “Vision-only robot navigation in a neural radiance world,” *IEEE Robotics and Automation Letters*, vol. 7, no. 2, pp. 4606–4613, 2022.
 - [30] Y. Li, S. Li, V. Sitzmann, et al., “3d neural scene representations for visuomotor control,” in *Conference on Robot Learning*, PMLR, 2022, pp. 112–123.

- [31] Y.-C. Lin, P. Florence, A. Zeng, et al., “Mira: Mental imagery for robotic affordances,” in *Conference on Robot Learning*, PMLR, 2023, pp. 1916–1927.
- [32] Z. Jiang, Y. Zhu, M. Svetlik, et al., “Synergies between affordance and geometry: 6-dof grasp detection via implicit representations,” *Robotics: Science and Systems*, 2021.
- [33] T. Weng, D. Held, F. Meier, and M. Mukadam, “Neural grasp distance fields for robot manipulation,” *IEEE International Conference on Robotics and Automation (ICRA)*, 2023.
- [34] 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.
- [35] 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.
- [36] D. Ha, A. M. Dai, and Q. V. Le, “Hypernetworks,” in *International Conference on Learning Representations*, 2017.
- [37] 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.
- [38] E. Coumans and Y. Bai, *Pybullet, a python module for physics simulation for games, robotics and machine learning*, 2016.
- [39] D. P. Kingma and J. Ba, “Adam: A method for stochastic optimization,” in *International Conference on Learning Representations*, 2015.
- [40] Q.-Y. Zhou, J. Park, and V. Koltun, “Open3D: A modern library for 3D data processing,” *arXiv:1801.09847*, 2018.
- [41] F. Williams, *Point cloud utils*, 2022.

**Reactive Motion Generation via Phase-varying Neural Potential
Functions**

Ahmet Ercan Tekden, Dimitrios Kanoulas, Aude Billard, Yasemin
Bekiroglu

IEEE Robotics and Automation Letters, vol. 11, no. 7, July 2026.
©2026 IEEE DOI: 10.1109/LRA.2026.3692035

The layout has been revised.

Abstract

Dynamical systems (DS) methods for Learning-from-Demonstration (LfD) provide stable, continuous policies from few demonstrations. First-order dynamical systems (DS) are effective for many point-to-point and periodic tasks, as long as a unique velocity is defined for each state. For tasks with intersections (e.g., drawing an “8”), extensions such as second-order dynamics or phase variables are often used. However, by incorporating velocity, second-order models become sensitive to disturbances near intersections, as velocity is used to disambiguate motion direction. Moreover, this disambiguation may fail when nearly identical position-velocity pairs correspond to different onward motions. In contrast, phase-based methods rely on open-loop time or phase variables, which limit their ability to recover after perturbations. We introduce Phase-varying Neural Potential Functions (PNPF), an LfD framework that conditions a potential function on a phase variable which is estimated directly from state progression, rather than on open-loop temporal inputs. This phase variable allows the system to handle state revisits, while the learned potential function generates local vector fields for reactive and stable control. PNPF generalizes effectively across point-to-point, periodic, and full 6D motion tasks, outperforms existing baselines on trajectories with intersections, and demonstrates robust performance in real-time robotic manipulation under external disturbances.

1 Introduction

Learning-from-Demonstration (LfD) infers task policies directly from demonstration data, removing the need for manual design. A central challenge in LfD is producing motions that follow demonstrations while remaining robust to perturbations. Among LfD approaches, dynamical systems (DS)-based methods represent motions using ordinary differential equations (ODEs) [1],

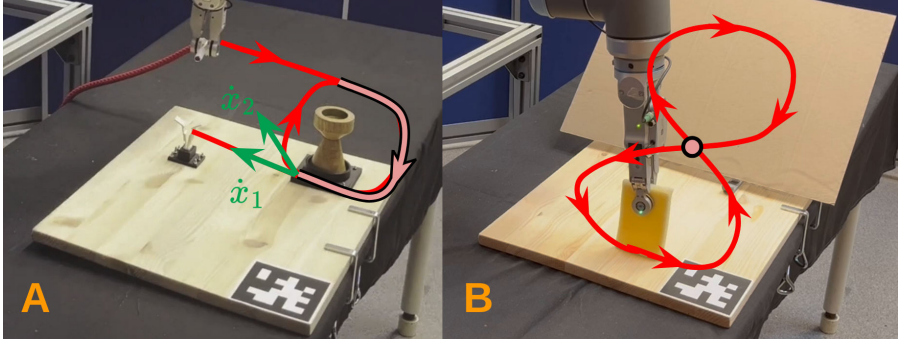


Figure 1: Reactive motion generation for tasks (6D trajectories) where the robot revisits similar states: (A) point-to-point knotting, with identical motion segments branching into different actions $\dot{x}_1, \dot{x}_2$ from the same position-velocity state (ambiguous for second-order models), and (B) 8-shaped periodic wiping.

[2], enabling stable motion generation from few demonstrations. Time- or phase-dependent DS approaches use an explicit temporal variable [1], allowing modeling of tasks where the same state is revisited at different times (Figure 1). However, reliance on a temporal variable limits their ability to recover smoothly after perturbations. In contrast, time-independent DS approaches generate motions from the current state alone [2], [3], [4], [5], [6], [7], offering greater reactivity. For tasks with intersections, they can be extended with velocity information to model the required motions. However, this introduces two key limitations: (i) sensitivity to disturbances near intersections due to reliance on velocity to disambiguate motion direction, and (ii) ambiguity when similar motion segments yield nearly identical position-velocity pairs that require different onward actions (e.g., $\dot{x}_1$ and $\dot{x}_2$ in knotting task, Figure 1).

DS approaches face a trade-off between modeling capability and reactivity to perturbations. We address this with Phase-varying Neural Potential Functions (PNPF) ¹, artificial potential functions conditioned on a novel state-dependent phase variable. *The phase is estimated directly from state progression*, ensuring that the DS remains time-invariant. PNPF combines two energy functions: (i) a nominal energy, which encodes the desired task

¹Project page: <https://fzaero.github.io/PNPF/>

behavior and decreases monotonically with task progress; and (ii) a safety energy, which discourages leaving the boundary of demonstrated data, with this boundary estimated in a data-driven manner from the visited states.

The nominal energy provides a scalar measure of task progress, enabling estimation of the phase variable directly from the state without relying on time or velocity. This addresses the two main weaknesses of second-order dynamical system models: their sensitivity to velocity disturbances and the ambiguity introduced by repeated states.

The safety energy guides the system back toward the region validated by demonstrations whenever it moves outside, while still allowing compliant behavior within. This region encodes the workspace and task constraints implicitly followed by the demonstrator, representing the part of the state space where reliable task execution has been observed. Staying within this region is desirable because it ensures that the robot operates in areas well-supported by data, minimizing the risk of unsafe or infeasible behavior. At the same time, the system can adapt to variations within the demonstrated region. For example, if the robot is subjected to an external disturbance that moves it to a nearby state within the demonstrated workspace, or if it encounters obstacles, it can adjust its trajectory while remaining in the reliable region. This approach balances the need for safety and reliability while allowing flexibility within the validated region.

This modular design of PNPF separates phase estimation from demonstration-based safety constraints and enables the integration of additional energy terms, such as those for obstacle avoidance, as demonstrated in the experiments. The method handles point-to-point and periodic motions, generalizes to tasks in configuration or task space, and supports full 6D motion while preserving reactivity.

Our main contributions are: (i) a reactive motion generation framework based on neural potential functions that models point-to-point and periodic motions from few demonstrations, while incorporating a data-driven safety set, to keep the robot within states observed in demonstrations; (ii) a state-dependent phase variable that tracks task progress while preserving reactivity to external disturbances; and (iii) compatibility with configuration- and task-space motion generation, including full 6D tasks. We evaluate our approach on 2D handwriting and 6D real-robot tasks, benchmarking it against state-of-the-art reactive motion generation methods. PNPF enables robust real-robot

execution and outperforms baselines in overall performance, particularly in scenarios with intersecting trajectories where other methods fail.

2 Related work

LfD is a widely adopted paradigm in robot learning for acquiring new skills [8], [9]. Prominent approaches include state-only DS models [2], [3], [4], [5], [6], [7], [10], [11], [12], [13], and time- or phase-dependent motion generation methods, such as movement primitives [1], [14], [15], [16], [17], [18]. State-only DS models represent motion as vector fields, enabling online reactivity to perturbations [19]. To handle tasks with intersecting trajectories, second-order representations incorporating velocity are preferred. However, these models can be sensitive to disturbances near intersections and may become ambiguous when intersections are preceded by similar motion segments (Sec. I). In contrast, movement primitives manage intersections using time or phase variables, but this comes at the cost of reduced reactivity, since progression is governed by an open-loop schedule rather than the current state. Our approach combines the strengths of both families by using a phase variable to parameterize a potential function, preserving DS-style reactivity and matching the modeling capacity of movement primitives for point-to-point and periodic tasks.

Extensions of DS models include discrete sequencing and locally active regions. In the former, complex skills are divided into subtasks, each modeled as a stable DS, with sequencing handled by a Hidden Markov Model (HMM) [20]. The HMM introduces a discrete notion of progress: subtask switching is governed by transition probabilities that increase near the attractor of the current mode. This allows revisiting the same state across different stages of the task, but subtask switching can be delayed under perturbations, as transitions depend on attractor proximity. In the latter, locally active regions impose stiffness-like attraction around a reference trajectory while preserving global stability [21]. These regions are predefined rather than derived from demonstrations and tie recovery to a single path. In contrast, our method provides continuous progress via a phase variable estimated directly from state progression, and learns attraction to demonstration regions obtained from data, enabling robust sequencing and safe recovery. Regions analogous to our safety boundary can also be specified through control barrier functions (CBF) [22]. For instance, [23] constructs data-driven CBF safety constraints using a small

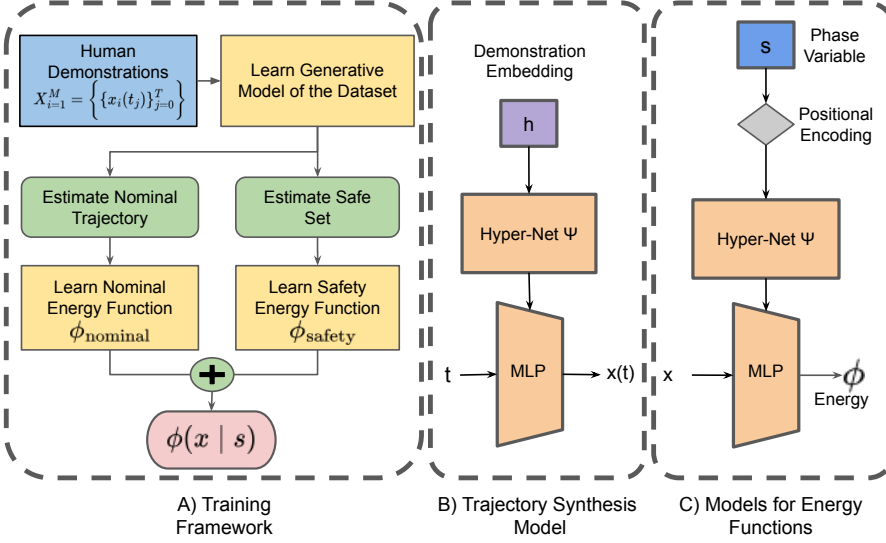


Figure 2: The proposed model architecture: (A) shows how PNPf $\phi(x | s)$ are constructed. (B) and (C) shows the neural network architectures utilized for trajectory generation and energy function estimation.

set of linear hyperplanes defined by heuristic rules. This formulation, however, limits the safety region to coarse, convex, piecewise-linear shapes. Although convex curved sets can be approximated with additional hyperplanes, the approach does not readily scale to more complex geometries. In contrast, our safety energy models the boundary as a single smooth function, enabling a much wider variety of smooth shapes without relying on numerous local linear approximations.

Artificial potential functions [24] generate reactive motion by shaping landscapes that attract the robot to targets and repel it from obstacles. Recent work has learned such functions from demonstrations [10], [11], but often suffers from stiff dynamics [10] or underconstrained optimization [11], leading to unpredictable behavior in unseen states. Relying solely on state information also limits the modeling of tasks with repeated state visits. Our approach instead builds potential functions from synthesized trajectories, jointly encoding task behavior and safety regions, and conditions them on a state-dependent

phase variable, enabling predictable behavior in unseen states and handling repeated state visits.

Smooth potential functions are crucial for generating continuous vector fields that support stable and reactive control. We achieve this using neural fields [25], coordinate-based networks that compactly represent continuous functions and have seen success in visual [26], [27], and robotic [17], [28], [29], [30], [31] applications. Previous works have used neural fields for motion modeling [17], but these methods were not designed for online use and lack the smoothness required for reactive control. We address this with a positional encoding tailored for smoothness, ensuring that the gradients of the energy function, i.e., the vector field, vary smoothly with the robot state, enabling stable and reactive real-time control.

In summary, we present an LfD framework that combines the reactivity of state-based DS with the expressiveness of movement primitives, outperforming prior approaches on complex motions and supporting real-time control.

3 Motion generation with Potential functions

In the proposed model, trajectories follow the gradient of a scalar-valued potential function, which defines an energy landscape over the state space that encodes goal-directed behavior. By following the gradient of the landscape, the desired motion arises naturally as trajectories that converge to low-energy regions representing task goals. Formally, we define a potential function $\phi(x \mid s)$ where $x \in \mathbb{R}^d$ is the state in the configuration or task space, and s is a phase variable, denoting task progress. The policy is then expressed as $\dot{x} = -\nabla_x \phi(x \mid s)$. This formulation enables time-independent control while allowing tasks to revisit the same state at different timesteps via phase conditioning. Figure 2 illustrates the potential function construction process and the proposed neural network architectures.

To construct the potential function $\phi(x \mid s)$, we propose a design composed of two components: nominal and safety energy functions. The nominal energy function encodes task-specific behavior by guiding the system along an unknown nominal trajectory $x^*(t)$. It supports state-dependent estimation of the phase variable s to keep the system synchronized with task progression (Sec. 3.3). The safety energy enforces active corrections by attracting the system toward a demonstration-based safety region, defined as a data-

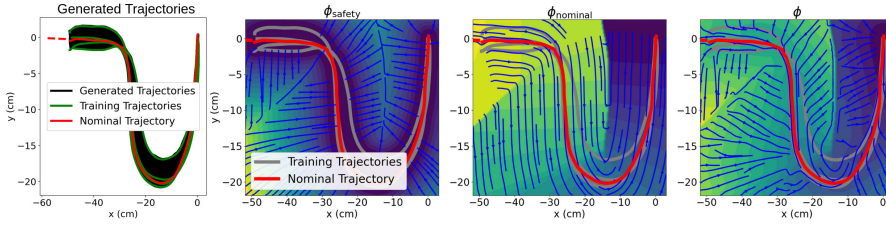


Figure 3: Generated trajectories, and the corresponding potential function in the form of $\phi(x)$. The generated trajectories densely cover the safe set, which contains the training demonstrations (green). The nominal trajectory (solid red) has the lowest DTW distance to demonstrations; the dashed segment shows its extrapolation. ϕ_{nominal} guides the robot to follow the nominal behavior while ϕ_{safety} guides the robot back to the safe set.

driven support region around the demonstration states and formally denoted as the safety set C . Given M human demonstrations as state trajectories $X_{i=1}^M = \{x_i(t_0), x_i(t_1), \dots, x_i(t_T)\}$, we utilize a trajectory generation model to infer both the unknown nominal trajectory $x^*(t)$ and the safe set C . This structure yields motions that are both safe and consistent with demonstrated behavior.

3.1 Trajectory synthesis through generative modeling

We learn a generative model that produces trajectories similar to the demonstrations while densely filling the gaps between them. The training trajectories are time-aligned and parameterized over $t \in [0, T]$, and examples are shown in Figure 3. These generated trajectories are used to estimate the nominal trajectory $x^*(t)$ and the safe set C .

We use decoder-only neural networks [26], in which conditioning is performed via a hypernetwork [32] to train the trajectory synthesis model. These architectures have fewer parameters to learn and work robustly even with a small number of data samples [17]. In decoder-only neural networks, instead of training an encoder to predict latent vectors from demonstrations, we directly assign a learnable embedding to each demonstration, which is optimized jointly with the network weights during training via backpropagation.

At inference, we estimate the nominal trajectory and the safe set by generating novel trajectories from latent vectors h' , sampled from the convex hull of the training embedding vectors h_i . This sampling procedure en-

sures that the generated trajectories remain within the interpolation region defined by the human demonstrations. We utilize the trajectory generation model to approximate the nominal trajectory $x^*(t)$ by selecting, among the generated trajectories $\{\tilde{x}_n(t)\}$, the one that minimizes the sum of dynamic time warping (DTW) [33] distances to the demonstrations, i.e., $x^*(t) = \arg \min_{\tilde{x}_n(t)} \sum_{i=1}^M \text{DTW}(\tilde{x}_n(t), x_i(t))$. In addition, the beginning of the $x^*(t)$ is extended using a first-order polynomial to ensure the continuity of nominal energy at t_1 .

We formally define the safety set, i.e., the safety region inferred from demonstrations, as $\mathcal{C} = \{x \mid \text{SDF}(x) \leq 0\}$, where the continuous signed distance function (SDF), obtained from demonstration data, quantifies the distance to the boundary with negative values inside the region and positive values outside. We employ our trajectory model to generate additional samples, thereby enhancing the density of the safety region and the precision of boundary approximation. Specifically, we generate a large number of trajectories and label the states visited along these trajectories as belonging to the safety region, thereby constructing the set X_{in} . Points outside the safety region, denoted X_{out} are obtained by sampling the state space and selecting those farther than a threshold ϵ_{SDF} from X_{in} . The SDF is then defined as the distance to the closest point in the opposite set: $\text{SDF}(x) = \min_{x_{\text{in}} \in X_{\text{in}}} \|x - x_{\text{in}}\|$ if $x \in X_{\text{out}}$, and $\text{SDF}(x) = -\min_{x_{\text{out}} \in X_{\text{out}}} \|x - x_{\text{out}}\|$ if $x \in X_{\text{in}}$.

3.2 Modeling energy functions for task representation

Human demonstrations specify correct behavior only in the regions they visit, i.e., the safe set. Accordingly, we assume the robot can reliably complete the task by either (i) following the nominal trajectory within the safe set or (ii) returning to the safe set when outside.

Consequently, we derive two energy functions from the demonstrations: a safety energy function ϕ_{safety} , which guides the robot back to the safe set, and a nominal energy function ϕ_{nominal} , which captures the nominal behavior within the safe set. We first derive these functions based solely on the state x . Then, in the following subsection, we show how nominal energy is used as a phase variable.

The safety energy is represented using the SDF estimated in the previous section and is given by $\phi_{\text{safety}}(x) = \text{relu}(\lambda_{\text{safety}}(\text{SDF}(x)))$ where λ smooths the function near 0, ensuring that the gradients of ϕ_{safety} decay gradually as $\text{SDF}(x)$ approaches 0. By construction, $\phi_{\text{safety}}(x) \geq 0$ and vanishes for all

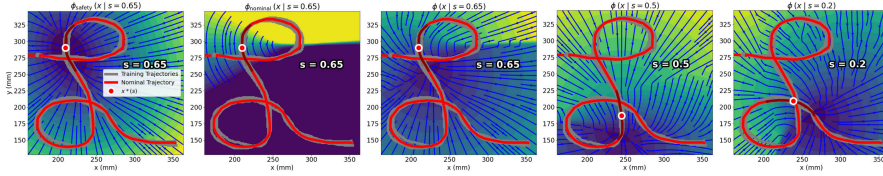


Figure 4: Example visualizations of $\phi_{\text{safety}}(x | s)$ and $\phi_{\text{nominal}}(x | s)$ used for estimating $\phi(x | s)$, along with $\phi(x | s)$ at different values of s . s is normalized to be between 0 and 1.

$x \in \mathcal{C}$.

To define the nominal energy function, we use the remaining arc length of the nominal trajectory. Given the discretized nominal trajectory $\{x_j^* = x^*(t_j)\}_{j=0}^N$, where N denotes the final index of the trajectory, the nominal energy at point x_k^* is defined as $\phi_{\text{nominal}}(x_k^*) = \sum_{j=k}^{N-1} \|x_{j+1}^* - x_j^*\|$. This scalar decreases monotonically as the robot progresses along the trajectory and satisfies $\phi_{\text{nominal}}(x_N^*) = 0$. For states not on the nominal path, we approximate the energy by projecting the state onto the closest point on x_k^* , resulting in

$$\phi_{\text{nominal}}(x) = \phi_{\text{nominal}}(x^*(\arg \min_k \|x_k^* - x\|)), \quad (\text{C.1})$$

which encourages motion parallel to the nominal trajectory and ensures $\phi_{\text{nominal}}(x) \geq 0$ for all x . The final potential function ϕ is expressed as the summation of safety and nominal energy functions: $\phi(x) = \phi_{\text{nominal}}(x) + k_{\text{safety}} \times \phi_{\text{safety}}(x)$ where $k_{\text{safety}} > 0$ weights the safety term ($k_{\text{safety}} = 1$ in all experiments). By construction, $\phi(x) \geq 0$ and becomes 0 only at states inside the safety set where the nominal energy reaches zero, i.e., data-driven region corresponding to the terminal states of the demonstrated trajectories. This is illustrated in the example potential function shown in Figure 3.

3.3 Phase-varying potential function design

The potential function ϕ described in the previous section models many tasks effectively, but it cannot represent behaviors that revisit the same state at different timesteps. Since ϕ assigns a single scalar value to each state, it cannot encode multiple energy levels at the same location, making revisits impossible under gradient flow. To address this, we represent the potential

function with a new phase variable derived from the nominal energy.

As the robot follows the nominal trajectory $x^*(t)$, the nominal energy decreases monotonically to 0. We can therefore reparameterize x_k^* as $x^*(s_k)$, where $s_k = \phi_{\text{nominal}}(x_k^*)$. Using s as a phase variable, we represent ϕ dynamically by limiting the estimation of the nominal and safety energy functions to the region around the current s_t value². Specifically, given the energy window s_w , we estimate $\text{SDF}(x \mid s_k)$ and $\phi_{\text{safety}}(x \mid s_k)$ using the points $x \in X(s_k : s_k + s_w)$, and $\phi_{\text{nominal}}(x \mid s_k)$ based on the segment of the nominal trajectory $x^*(s_k - s_w : s_k + s_w)$. With these updates, the PNPF is expressed as:

$$\phi(x \mid s) = \phi_{\text{nominal}}(x \mid s) + k_{\text{safety}} \Lambda_{\text{safety}}(s) \phi_{\text{safety}}(x \mid s) \quad (\text{C.2})$$

where $\Lambda_{\text{safety}}(s)$ increases the strength of the safety term as s decreases². While not required, it biases the potential toward stronger safety convergence as the task progresses. Figure 4 illustrates an example composition of the energy functions and potential functions at different s values, showing how the PNPF model outputs different velocities at similar states. With these formulations, given the initial state x and the phase variable s , the following equations are used to estimate the control signal and update the phase variable:

$$\dot{x} = -\alpha \nabla_x \phi(x \mid s), \quad s \leftarrow \phi_{\text{nominal}}(x' \mid s) \quad (\text{C.3})$$

where $\alpha > 0$ is the control gain and x' is the new state after applying $\dot{x}$. This controller enables the robot to perform the task correctly while providing robustness to perturbations due to the state-dependent phase evolution.

Due to the compositional nature of energy functions, we can integrate additional terms into ϕ , such as for obstacle avoidance or goal reaching. Following the standard artificial potential formulation [24], we represent goals as attractive potentials and obstacles as repulsive potentials. When the phase variable s falls below a given threshold, the attractive potential term is enabled to ensure asymptotic convergence to the goal, while the repulsive term remains active throughout to maintain obstacle clearance.

²In practice, s is normalized to be between 1 and 0, by dividing it by s_0 .² We use $\Lambda(s) = 1 + (1 - \tilde{s})^{10}$ with $\tilde{s} = s/s_0$ in all experiments.

3.4 Stability of PNPF

We consider the closed-loop system $\dot{x} = -\alpha \nabla_x \phi(x | s)$ with phase update $s \leftarrow \phi_{\text{nominal}}(x | s)$, and use $V(x, s) = \phi(x | s)$ as a Lyapunov candidate. The potential is the sum of nominal and safety energies, is nonnegative, and vanishes only on the target set S given by the intersection of the safety set and the zero level set of ϕ_{nominal} . Both terms are parameterized with ReLU neural fields, yielding continuous, piecewise-smooth energy functions (see Sec. 3.5) with gradients well-defined almost everywhere, except on a measure-zero set where activations switch ³, inducing a locally Lipschitz closed-loop vector field in the data-supported region ⁴. Since summation may introduce spurious stationary points via gradient cancellation, we assume the demonstrations are locally consistent, i.e., $\nabla_x \phi_{\text{nominal}}(x | s)^\top \nabla_x \phi_{\text{safety}}(x | s) \geq -\varepsilon \|\nabla_x \phi_{\text{nominal}}(x | s)\|^2$, for small $\varepsilon > 0$, which preserves a descent direction for V , since $\dot{V} = -\alpha \|\nabla_x \phi\|^2$ and $\|\nabla_x \phi\|^2 \geq \|\nabla_x \phi_{\text{safety}}\|^2 + (1 - 2\varepsilon) \|\nabla_x \phi_{\text{nominal}}\|^2 > 0$. This assumption limits the method to unimodal tasks, where aligned velocities at each phase yield locally consistent motion directions. The phase update is discrete (see Eq. (7)); for analysis, we use a local continuous-time approximation valid for $|\Delta s| < s_w$, under which the energy landscape can be treated as locally invariant in s , yielding $\dot{s} = \nabla_x \phi_{\text{nominal}}(x | s)^\top \dot{x} = -\alpha \nabla_x \phi_{\text{nominal}}(x | s)^\top \nabla_x \phi(x | s)$. Under the above assumption $\nabla_x \phi_{\text{nominal}}(x | s)^\top \nabla_x \phi(x | s) \geq (1 - \varepsilon) \|\nabla_x \phi_{\text{nominal}}(x | s)\|^2 > 0$, and therefore $\dot{s} < 0$ for $s_0 > s(0) > 0$, implying monotonic phase decrease and $s(t) \rightarrow 0$. Along trajectories, $\dot{V} \leq 0$, and by LaSalle’s invariance principle, trajectories initialized in the data-supported region converge to the largest invariant subset of S , establishing regional asymptotic convergence.

3.5 Modeling smooth potential functions with neural fields

We model $\phi_{\text{safety}}(x | s)$ and $\phi_{\text{nominal}}(x | s)$ using neural fields, as they are well suited to represent continuous functions and support conditional modeling, including with phase variables. These networks are fully differentiable, allowing us to estimate $\dot{x}$ as $-\nabla_x \phi(x | s)$. Maintaining continuous gradients

³At these non-differentiable points, automatic differentiation returns a valid subgradient, ensuring the controller remains numerically well-defined. This is consistent with our piecewise formulation based on phase-windowed energies and a clipped safety term.

⁴The data-supported region is the subset of the state space on which the model is trained. For each phase value s , training samples are generated as $x^*(s) + u$, where u is drawn from a bounded Euclidean ball.

in the potential function supports smooth robotic motion, as discontinuities or plateaus can lead to getting stuck in local minima.

Positional encoding [34] is commonly used in neural fields to represent complex functions but may introduce gradient discontinuities [27]. To overcome this, we use a hyper-network-conditioned MLP, where the hyper-network takes the position-encoded phase variable s , and the MLP takes the non-position-encoded state variable x . This design leverages the known spectral bias of standard MLPs with ReLU activations [34], which tend to represent smoother, low-frequency functions when operating directly on coordinates, and we harness this property to produce smooth energy landscapes in x . This design captures motion patterns with sharp curvature changes and loops over time. At the same time, it preserves locally smooth energy landscapes in x by modeling phase-windowed regions using non-position-encoded state inputs. Training uses states sampled from neighborhoods of $x^*(s)$ at each phase. However, since the energy functions are learned, we additionally include a lightweight sampling-based safeguard that activates only when gradient-based progress stalls. The safeguard uses the potential as a running cost by sampling candidate actions and evaluating them via one-step rollouts. We train all energy functions using the AdamW [35] optimizer.

3.6 Modeling periodic motions

To model periodic motions, we modify the input of the trajectory generation model and the energy functions to be periodic. Given period $\mathcal{T}$ corresponding to one cycle, the trajectory generation model then takes input $t^\circ = [\sin(2\pi \frac{t}{\mathcal{T}}), \cos(2\pi \frac{t}{\mathcal{T}})]$, and the energy functions take phase input $s_k^\circ = [\sin(2\pi \frac{s_k}{s_{\mathcal{T}}}), \cos(2\pi \frac{s_k}{s_{\mathcal{T}}})]$, where $s_{\mathcal{T}}$ denotes the arc length of one nominal period. Accordingly, ϕ is now expressed as:

$$\phi(x | s) = \begin{cases} \phi_{\text{nominal}}(x | s^\circ) + k_{\text{safety}} \phi_{\text{safety}}(x | s^\circ) & \text{if } s \geq 0, \\ k_{\text{safety}} \phi_{\text{safety}}(x | s^\circ) & \text{otherwise.} \end{cases} \quad (\text{C.4})$$

When $s \leq 0$, the second condition will lead the motion to conclude while still providing a stable DS equation. With this new formulation, we use the following phase update:

$$s \leftarrow s - (\phi_{\text{nominal}}(x | s^\circ) - \phi_{\text{nominal}}(x' | s^\circ)). \quad (\text{C.5})$$

3.7 Modeling 6D motions

For motions with both position and orientation (unit quaternions), we follow [36], [37] and represent orientations in axis-angle form. Given a quaternion trajectory $q_{1:T}$, we express the axis-angle trajectory $r_{1:T}$ in the tangent space of the final quaternion q_T (for periodic motions, q_1 is used instead) as $r_t = \text{Log}(q_t * \bar{q}_T)$, where $\bar{q}_T$ is the conjugate of q_T , $*$ denotes quaternion multiplication, and Log is the logarithmic map from quaternions to axis-angle vectors [38]. The resulting $r_{1:T}$ is continuous and can be modeled in the same way as Cartesian trajectories. At inference time, $-\nabla\phi$ yields angular velocity for direct robot control, or predicted $\hat{r}_{1:T}$ can be mapped back via $\hat{q}_t = \text{Exp}(\hat{r}_t) * q_T$, with Exp denoting the exponential map from axis-angle to quaternions [38].

4 Simulation-based evaluations

In this section, we first evaluate the performance of our model on three state-of-the-art simulated datasets: LASA [2] (2D), LAIR [6] (2D), RoboTasks [37], [39] (6D). We also introduce two new 2D datasets, CHAR and CHAR-Periodic, both of which are more challenging than LAIR⁵. In CHAR, trajectory intersections are preceded by similar motion segments but require different onward action predictions. As a result, while second-order state representations are sufficient for LAIR, they fail on CHAR. We also demonstrate how our model can react to obstacles and spatial disturbances without affecting the geometry of the motion. For comparative evaluation, we benchmark against two baseline models: NODE [7] and CONDOR [6]⁶. Note that CONDOR natively supports second-order state representations.

Most reactive motion generation methods for periodic motion modeling rely on first-order state-only DS formulations, which are insufficient for the CHAR-Periodic dataset. For experiments with this dataset, we introduce a naive baseline that repeats the first periodic segment of the training trajectory three times to match the test length.

For the 2D motion datasets, we evaluate the performance of the model using four metrics: DTW Distance (DTWD), Fréchet Distance (FD) [40], Final

⁵We limit the evaluation to unimodal tasks.

⁶The original implementation for CONDOR uses the same trajectories for training and testing; we modified it to separate them, leading to slightly lower performance than originally reported.

Table 1: Method performance across different datasets.

Dataset	Method	DTWD	FD	FP Error	Accuracy
LASA (cm)	PNPF	1.81	3.38	0.02	1.00
	CONDOR	2.10	4.14	0.10	1.00
	NODE	2.12	4.33	1.82	0.76
LAIR (mm)	PNPF	4.82	11.75	5.04	1.00
	CONDOR	7.62	17.43	7.91	0.96
CHAR (mm)	PNPF	2.96	8.44	2.00	1.00
	PNPF-50	3.16	8.70	2.02	1.00
	PNPF-H	3.35	9.40	2.01	1.00
	CONDOR	17.67	45.56	3.25	1.00
Char-Periodic (mm)	PNPF	7.77	14.30	NA	NA
	Naive	10.70	24.28	NA	NA
Dataset	Method	DTWD-P (cm)	DTWD-O (rad)	FP Error (cm)	F0 Error (rad)
RoboTasks	PNPF	2.19	0.08	0.25	0.17
	NODE	2.62	0.10	1.51	0.13

Position (FP) error, and accuracy. To measure the accuracy, we define a prediction as successful if the distance between the final point of the ground truth and the predicted trajectory is within 2 cm. The FP error is omitted for periodic motions, as these typically lack a well-defined endpoint. For RoboTasks, we use the DTWD for positions (DTWD-P) and orientations (DTWD-O), FP error, and final orientation (FO) error as evaluation metrics. Among these metrics, DTWD and FD measure geometric fidelity, while FP error, FO error, and accuracy measure the stability of the predicted trajectories. To ensure consistency, all predicted trajectories are rescaled to length 1000 and DTWD is normalized accordingly. In addition, to improve the reliability of performance estimates and assess robustness to random initialization, all models are trained and evaluated using three different random seeds.

The overall mean performance across all datasets ⁷ is reported in Table 1, where our method achieves the highest average score across all baselines. In particular, in the CHAR dataset, which contains motions with repeated state visits at different timesteps, our method reduced DTWD by around 80% com-

⁷NODE results are omitted for LAIR and CHAR, as the method cannot model tasks with repeated states.

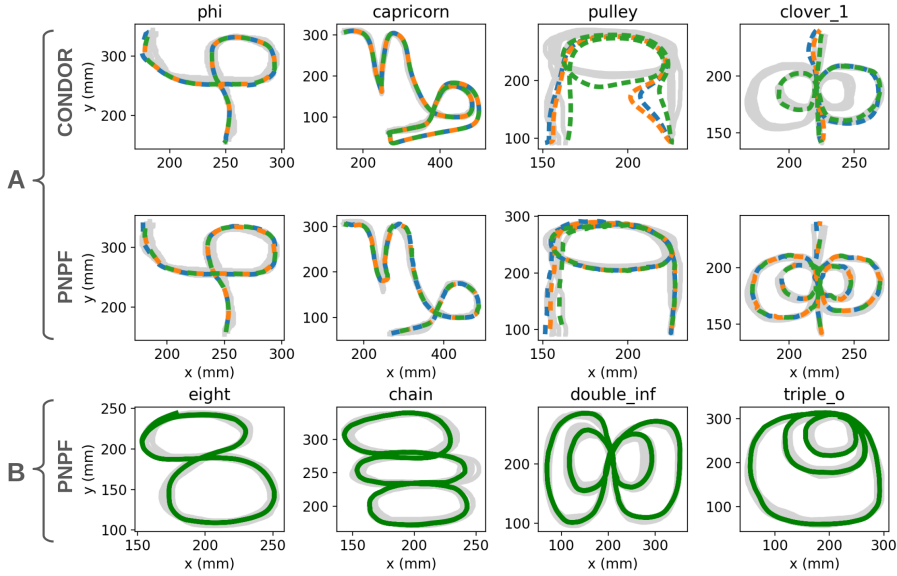


Figure 5: Comparison with CONDOR (A), and periodic motions from our method (B). Grey lines are demonstrations; dashed colored lines (A) are reproductions from different initial states. PNPF reproduces motions more faithfully, while CONDOR mismodels (capricorn) and misses segments (pulley, clover_1). In capricorn, our method smooths out sharp corners.

pared to the baseline. We additionally evaluate the robustness of our method on this dataset to suboptimal nominal trajectory selection by selecting the candidate (i) with the highest DTWD (PNPF-H) and (ii) from a reduced pool of 50 trajectories (PNPF-50). While both settings degrade performance compared to the default configuration, they still achieve better results than the baseline across all metrics.⁸ Table 2 presents individual results for LAIR and CHAR to further isolate the effect of phase-based modeling versus second-order state input. On LAIR, where both methods can model the tasks, our approach produces more consistent results across seeds, while CONDOR occasionally mismodels segments (e.g., capricorn in Figure 5). On CHAR, our method consistently outperforms the baseline, maintaining correct mo-

⁸Note that artifacts in human demonstrations (e.g., knots) can propagate to the selected nominal trajectory and degrade performance.

Table 2: LAIR and CHAR RESULTS

	DTWD (mm)		FD (mm)		FP Error (m)		Accuracy	
	PNPF	CONDOR	PNPF	CONDOR	PNPF	CONDOR	PNPF	CONDOR
capricorn	6.22	18.91	18.33	49.99	3.30	10.59	1.00	0.67
double_lag	3.86	4.84	10.35	10.99	5.54	7.32	1.00	1.00
double_loop	5.81	10.09	14.50	17.16	8.07	8.63	1.00	1.00
e	4.81	4.54	11.15	12.36	2.90	4.71	1.00	1.00
Lag	4.15	3.70	8.52	8.53	4.07	5.38	1.00	1.00
phi	4.05	5.79	10.25	12.01	6.51	11.32	1.00	1.00
triple_loop	4.43	6.94	9.81	14.49	3.31	6.55	1.00	1.00
two	5.24	6.16	11.10	13.87	6.64	8.78	1.00	1.00
pulley	3.35	15.71	9.54	55.46	1.23	2.81	1.00	1.00
eyes	1.97	18.69	5.87	46.76	1.27	1.31	1.00	1.00
clover_1	3.18	25.64	9.35	64.88	3.01	3.41	1.00	1.00
clover_2	3.31	15.68	9.77	23.18	3.62	5.57	1.00	1.00
double_knot	3.00	12.62	7.65	37.53	0.88	3.15	1.00	1.00

tion through multiple intersections, whereas CONDOR misses key segments (e.g., pulley, clover_1 in Figure 5). These results highlight the benefits of the phase-varying potential formulation. However, one limitation we observe is that our method tends to smooth out sharp corners, as seen in the capricorn task.

In addition, to empirically assess the stability, a finite-horizon rollout analysis across all tasks and seeds on LASA, LAIR, and CHAIR was added. The phase variable was discretized into 100 values and 1000 perturbed states was sampled per phase of the form $x^*(s) + u$, where u was drawn uniformly from a local Euclidean ball. Evaluation was conducted for $\|u\| \leq 0.25r$, where r denotes the radius of the data-supported region. We disabled the safeguard mechanism to evaluate the prevalence of undesired critical points. Across 100-step closed-loop rollouts, in over 99.999% of sampled cases either both the phase variable and total potential decreased monotonically or the system reached the terminal region. Inspection of the rare failure cases indicates that they are associated with local gradient cancellation between the nominal and safety energy terms; enabling the safeguard resolves these cases, yielding a 100% success rate.

We evaluate reactivity to two types of perturbations: (i) obstacles, using the GShape task (LASA), and (ii) spatial disturbances, using the double_loop task (LAIR). For obstacles, we demonstrate basic avoidance capabilities using simple repulsive potentials. For spatial disturbances, we additionally com-

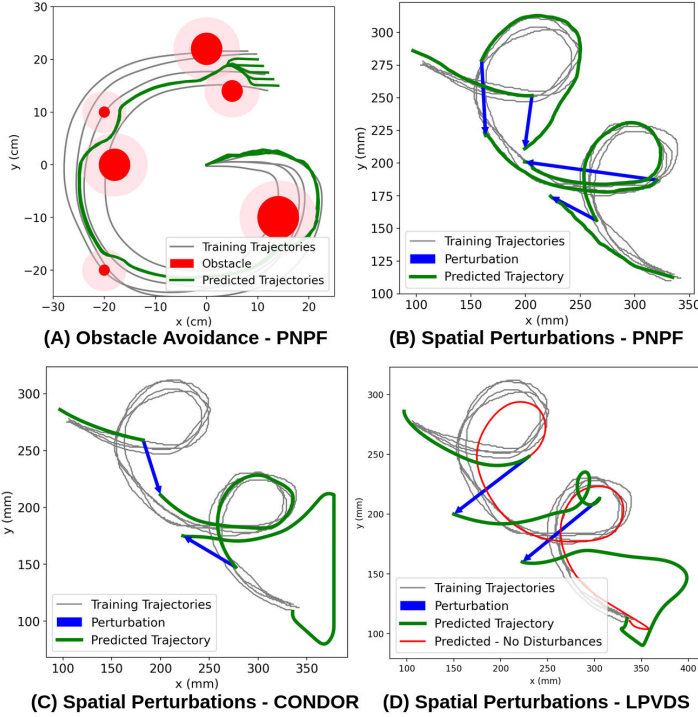


Figure 6: PNPF results for obstacle avoidance (A) and perturbations (B). CONDOR skips or repeats motion segments after perturbations (C), while LPVDS struggles to recover and drifts from the demonstrated path (D).

pare against CONDOR and Linear Parameter-Varying Dynamical Systems (LPVDS)⁹. After the perturbation, the desired behavior is to resume the task smoothly without skipping or repeating segments. All methods restart from the disturbed position; baselines reset velocity while keeping the phase unchanged. This design introduces both temporal and spatial disturbances: velocity resets disrupt implicit progression, while the unchanged phase represents a temporal disturbance that the model must correct. As shown in Fig. 6,

⁹We use the LPVDS implementation (github.com/penn-figueroa-lab/lpvds), extended to second-order systems with PQLF-based stability and slack-variable relaxation of the Lyapunov constraints. Clusters are defined from time-aligned trajectory segments, as EM-based clustering performed poorly.

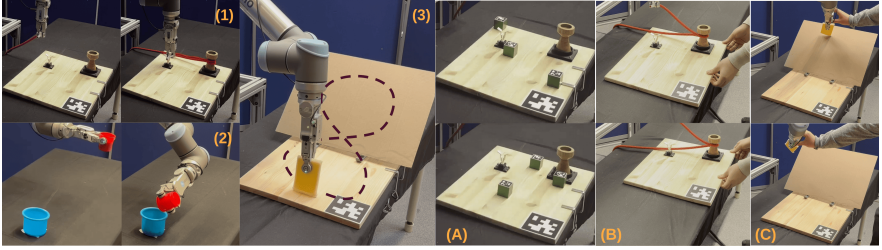


Figure 7: Real robot experiments: Start and end configurations (left), with periodic motion in Task 3 indicated by a dashed line. (A) Obstacle variations with AprilTag-tracked cubes, (B) task frame shifts from moving the board mid-task, and (C) end-effector displacements from manual robot repositioning.

our method successfully recovers as expected, since it updates the phase in a progress-aware and locally consistent manner, whereas CONDOR and LPVDS tend to skip or repeat segments, relying on velocity to infer task progression. Moreover, CONDOR completes the task from an unnatural direction, and LPVDS deviates from the demonstrated path. This shows that our method remains robust to velocity disturbances, in contrast to second-order DS models, which may perform poorly with imprecise velocity observations.

5 Real world experiments

To showcase our method’s capabilities, we design and execute three real-world robotic tasks, illustrated in Figure 7. The first is a knotting task, where the robot ties a rope around a cylinder for one full turn before inserting the rope’s end into a designated socket. The second task is pouring, where the robot transfers a glass of beans to another container. The final task is a 3D wiping operation that features a periodic figure-eight motion. All experiments are carried out using a UR10 robot equipped with an RG2-FT gripper.

For model training, we collect demonstration data through kinesthetic teaching, recording six trajectories for each of the first two tasks. Given the periodic nature of the final task, fewer trajectories are used (only two). For the first and third tasks, we use positions and quaternions, while for the second task, we use joint angles, demonstrating that our method can handle various state representations. All training trajectories are time-aligned using DTW.

We first evaluate whether our model can generate appropriate motions when the robot starts from varying positions. For each task, we run tests from 10 different initial locations to assess the model’s consistency and robustness. In the third task, to demonstrate periodic behavior, we initialize the phase variable s as $2 \times s_T$, producing two complete motion cycles. In all tests, the generated trajectories closely resemble the demonstrated ones, successfully complete their tasks, and exhibit the expected behavior. In these experiments, our framework is deployed online, operating at frequencies of up to 50 Hz.

Next, we perform an obstacle avoidance experiment in the first task to evaluate the reactivity of our method to environmental perturbations. In this setup, three cubes with side lengths of approximately 3.5 cm are placed within the workspace. Their positions relative to the task frame are tracked using AprilTags [41]. These cubes are treated as static columns, and the obstacle energy function is computed based on their x and y coordinate¹⁰. For simplicity, the robot’s end-effector is modeled as a point agent, with the obstacle radius adjusted accordingly. The experiment is repeated with two distinct obstacle configurations, both shown in Figure 7A. In all cases, the robot avoids obstacles and successfully completes the tasks.

Finally, we conduct spatial disturbance experiments on the first and third tasks to further assess the responsiveness of our method to perturbations. Two types of disturbance are applied: (i) shifting the task frame and (ii) pausing the robot and manually displacing the end effector from its current pose. Examples of these disturbances are shown in Figure 7B and C. The first type is applied only to the first task, as the task frame in the third task is fixed due to the occasional occlusion of the AprilTag used for tracking. Each task is tested with two distinct perturbation scenarios. In all cases, the robot successfully completes the tasks while demonstrating appropriate and adaptive behavior. Notably, no essential steps are skipped. For example, in the first task, when the robot is pushed backwards, causing the rope to untie from the cylinder, it responds correctly by re-tying the rope, completing the task as intended.

¹⁰Incorporating the z coordinate trivializes the task, as the motion primarily occurs above the cubes.

6 Conclusion

This work introduces a novel LFD framework for reactive motion generation based on phase-varying neural potential functions. The proposed method models a wide range of tasks as dynamical systems governed by a state-dependent phase variable while preserving reactivity to external perturbations. Simulation results show that our approach outperforms baselines on the evaluated datasets in terms of average DTWD and FD metrics. The method effectively models dynamical systems that require time- or progress-based parameterization, improving upon the limitations of many existing DS-based reactive motion generation methods while maintaining reactive behavior. We demonstrate its effectiveness on real-robot control across three manipulation tasks, including both point-to-point and periodic motions. A current limitation is the tendency to oversmooth sharp corners due to ambiguities in the local energy formulation. Future work will refine the energy design to better preserve directional structure and improve robustness to demonstration artifacts, and incorporate visual and force feedback.

Acknowledgments

We thank Balázs Kulcsár and Torsten Wik for their feedback on the stability analysis.

References

- [1] 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.
- [2] S. M. Khansari-Zadeh and A. Billard, “Learning stable nonlinear dynamical systems with gaussian mixture models,” *IEEE Trans. Robot.*, vol. 27, no. 5, pp. 943–957, 2011.
- [3] N. Figueroa and A. Billard, “A physically-consistent bayesian non-parametric mixture model for dynamical system learning,” in *Conf. Robot. Learn. (CoRL)*, 2018, pp. 927–946.

-
- [4] N. Figueroa, S. Faraji, M. Koptev, and A. Billard, “A dynamical system approach for adaptive grasping, navigation and co-manipulation with humanoid robots,” in *IEEE Int. Conf. Robot. Autom. (ICRA)*, 2020, pp. 7676–7682.
 - [5] F. Khadivar, I. Lauzana, and A. Billard, “Learning dynamical systems with bifurcations,” *Robot. Auton. Syst.*, vol. 136, p. 103 700, 2021.
 - [6] R. Pérez-Dattari and J. Kober, “Stable motion primitives via imitation and contrastive learning,” *IEEE Trans. Robot.*, 2023.
 - [7] F. Nawaz, T. Li, N. Matni, and N. Figueroa, “Learning complex motion plans using neural odes with safety and stability guarantees,” in *IEEE Int. Conf. Robot. Autom. (ICRA)*, 2024, pp. 17 216–17 222.
 - [8] B. D. Argall, S. Chernova, M. Veloso, and B. Browning, “A survey of robot learning from demonstration,” *Robot. Auton. Syst.*, vol. 57, no. 5, pp. 469–483, 2009.
 - [9] S. Schaal, “Learning from demonstration,” *Advances in neural information processing systems*, vol. 9, 1996.
 - [10] S. M. Khansari-Zadeh and O. Khatib, “Learning potential functions from human demonstrations with encapsulated dynamic and compliant behaviors,” *Auton. Robots*, vol. 41, pp. 45–69, 2017.
 - [11] M. A. Rana et al., “Learning reactive motion policies in multiple task spaces from human demonstrations,” in *Conf. Robot. Learn. (CoRL)*, 2020, pp. 1457–1468.
 - [12] S. Bahl, M. Mukadam, A. Gupta, and D. Pathak, “Neural dynamic policies for end-to-end sensorimotor learning,” *Adv. Neural Inf. Process. Syst.*, vol. 33, pp. 5058–5069, 2020.
 - [13] D. Totsila, K. Chatzilygeroudis, V. Modugno, D. Hadjivelichkov, and D. Kanoulas, “Sensorimotor learning with stability guarantees via autonomous neural dynamic policies,” *IEEE Robot. Autom. Lett.*, 2025.
 - [14] A. Paraschos, C. Daniel, J. R. Peters, and G. Neumann, “Probabilistic movement primitives,” *Advances in neural information processing systems*, vol. 26, 2013.
 - [15] M. Y. Seker, M. Imre, J. H. Piater, and E. Ugur, “Conditional neural movement primitives,” in *Robotics: Science and Systems*, vol. 10, 2019.

- [16] G. Li, Z. Jin, M. Volpp, F. Otto, R. Lioutikov, and G. Neumann, “Prodmp: A unified perspective on dynamic and probabilistic movement primitives,” *IEEE Robot. Autom. Lett.*, vol. 8, no. 4, pp. 2325–2332, 2023.
- [17] A. Tekden, M. P. Deisenroth, and Y. Bekiroglu, “Neural field movement primitives for joint modelling of scenes and motions,” in *IEEE/RSJ Int. Conf. Intell. Robot. Syst. (IROS)*, 2023, pp. 3648–3655.
- [18] Y. Yildirim and E. Ugur, “Conditional neural expert processes for learning movement primitives from demonstration,” *IEEE Robot. Autom. Lett.*, 2024.
- [19] A. Billard, S. Mirrazavi, and N. Figueroa, *Learning for adaptive and reactive robot control: a dynamical systems approach*. Mit Press, 2022.
- [20] J. R. Medina and A. Billard, “Learning stable task sequences from demonstration with linear parameter varying systems and hidden markov models,” in *Conf. Robot. Learn. (CoRL)*, 2017, pp. 175–184.
- [21] N. Figueroa and A. Billard, “Locally active globally stable dynamical systems: Theory, learning, and experiments,” *Int. J. Robot. Res.*, vol. 41, no. 3, pp. 312–347, 2022.
- [22] A. D. Ames, X. Xu, J. W. Grizzle, and P. Tabuada, “Control barrier function based quadratic programs for safety critical systems,” *IEEE Trans. Autom. Control*, vol. 62, no. 8, pp. 3861–3876, 2017.
- [23] M. Saveriano and D. Lee, “Learning barrier functions for constrained motion planning with dynamical systems,” in *IEEE/RSJ Int. Conf. Intell. Robot. Syst. (IROS)*, 2019, pp. 112–119.
- [24] 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.
- [25] Y. Xie, T. Takikawa, S. Saito, et al., “Neural fields in visual computing and beyond,” *Computer Graphics Forum*, 2022, ISSN: 1467-8659.
- [26] 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.

-
- [27] 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.
 - [28] M. Z. Irshad et al., “Neural fields in robotics: A survey,” *arXiv preprint arXiv:2410.20220*, 2024.
 - [29] T. Weng, D. Held, F. Meier, and M. Mukadam, “Neural grasp distance fields for robot manipulation,” *IEEE International Conference on Robotics and Automation (ICRA)*, 2023.
 - [30] A. Tekden, M. P. Deisenroth, and Y. Bekiroglu, “Grasp transfer based on self-aligning implicit representations of local surfaces,” *IEEE Robot. Autom. Lett.*, vol. 8, no. 10, pp. 6315–6322, 2023.
 - [31] Y. Chen, X. Gao, K. Yao, L. Niederhauser, Y. Bekiroglu, and A. Billard, “Implicit articulated robot morphology modeling with configuration space neural signed distance functions,” in *2025 IEEE International Conference on Robotics and Automation (ICRA)*, IEEE, 2025, pp. 4558–4564.
 - [32] D. Ha, A. M. Dai, and Q. V. Le, “Hypernetworks,” in *International Conference on Learning Representations*, 2017.
 - [33] M. Müller, “Dynamic time warping,” in *Information Retrieval for Music and Motion*, Springer, 2007, pp. 69–84.
 - [34] M. Tancik et al., “Fourier features let networks learn high frequency functions in low dimensional domains,” *Adv. Neural Inf. Process. Syst.*, vol. 33, pp. 7537–7547, 2020.
 - [35] I. Loshchilov and F. Hutter, “Decoupled weight decay regularization,” in *Int. Conf. Learn. Represent. (ICLR)*, 2019.
 - [36] A. Ude, B. Nemec, T. Petrić, and J. Morimoto, “Orientation in cartesian space dynamic movement primitives,” in *IEEE Int. Conf. Robot. Autom. (ICRA)*, 2014, pp. 2997–3004.
 - [37] S. Auddy, J. Hollenstein, M. Saveriano, A. Rodríguez-Sánchez, and J. Piater, “Continual learning from demonstration of robotics skills,” *Robot. Auton. Syst.*, vol. 165, p. 104427, 2023.
 - [38] F. S. Grassia, “Practical parameterization of rotations using the exponential map,” *J. Graph. Tools*, vol. 3, no. 3, pp. 29–48, 1998.

- [39] S. Auddy, J. Hollenstein, M. Saveriano, A. Rodríguez-Sánchez, and J. Piater, *Scalable and efficient continual learning from demonstration via a hypernetwork-generated stable dynamics model*, 2024.
- [40] T. Eiter and H. Mannila, “Computing discrete fréchet distance,” *Technical Report CD-TR 94/64*, 1994.
- [41] J. Wang and E. Olson, “AprilTag 2: Efficient and robust fiducial detection,” in *IEEE/RSJ Int. Conf. Intell. Robot. Syst. (IROS)*, 2016, pp. 4193–4198, ISBN: 978-1-5090-3762-9.

PAPER D

**Compositional Motion Generation from Demonstration with
Object-Centric Neural Fields**

Ahmet Ercan Tekden, Yasemin Bekiroglu

IEEE Robotics and Automation Letters, vol. 11, no. 9, September 2026.
©2026 IEEE DOI: 10.1109/LRA.2026.3713713

The layout has been revised.

Abstract

Compositionality, by organizing complex behavior as combinations of simpler elements, enables robot learning that is scalable and data efficient. Leveraging this principle, we propose a generative learning-from-demonstration framework that enables compositional modeling of robotic behavior by connecting perception and motion through shared object-level representations. We render scenes from object-centric neural representations that integrate canonical neural fields with latent-conditioned deformations, capturing positional and geometric variations in a smooth, consistent, and interpretable way. For motion generation, a temporal mixture-of-experts (MoE) employs a gating mechanism to combine object-conditioned movement primitives over time, producing complete trajectories. This spatial-temporal compositionality maintains the data efficiency of movement primitives while grounding motion in visual structure, enabling systematic generalization across diverse scene configurations. In simulation, long-horizon manipulation tasks are successfully completed using the proposed model, which requires significantly less training data than other image-based baselines. Real-world experiments further demonstrate the method’s robustness to noise, its ability to generalize at the category level through language-based segmentation models, and its capacity to operate directly on 3D scene representations.

1 Introduction

Learning from Demonstration (LfD) [1] is a widely used approach for teaching robots task-specific skills from human demonstrations. A central challenge is representing diverse, long-horizon behaviors from a limited number of demonstrations. Treating motion as a single, monolithic sequence overlooks its compositional structure, which complicates the learning process. Everyday tasks unfold through sequential, object-level interactions—reaching, grasping, moving, and placing—that together accomplish the overall goal. This motion-

level compositionality [2] can be complemented by scene-level compositionality, grounding motion in object-centric scene representations and providing a natural foundation for compositional motion modeling. Figure 1 illustrates an example of such object-centric structure, showing how the scene is decomposed into object-specific representations and how those representations relate to different phases of a manipulation task. By supporting data efficiency, modularity, and generalization, compositional formulations present a compelling alternative to monolithic approaches [3].

Movement primitives (MPs) [4], [5], [6], [7], [8], [9], [10] are widely used to encode and generalize demonstrated skills. MPs provide a compact and smooth representation of trajectories from only a few demonstrations by explicitly incorporating temporal variables into motion modeling. Typically, they are employed compositionally to represent atomic skills, which can then be combined to model more complex behaviors. However, this compositionality is often imposed manually, e.g., by concatenating primitives or specifying splits with via-points. Moreover, MPs generally rely on hand-designed, low-dimensional features (e.g., object positions, goal locations). Identifying suitable features and estimating them consistently across tasks is challenging. Alternatives include learning features directly from images [6], but this typically requires large datasets and sacrifices the data-efficiency advantage of MPs.

To address these limitations, we propose a generative approach to model scenes with smooth object-level parameterizations¹. This enables task representations based on compact latent parameters that preserve the efficiency and compositionality of MPs while grounding them in perception. Building on this perspective, we propose a Mixture-of-Experts (MoE)-based motion generation framework that integrates perception and action. A spatial MoE captures smooth, object-centric generative representations that provide the building blocks for motion modeling. Object-centric structure is encouraged by soft mask-based importance sampling, ensuring that each expert learns independently without collapsing onto the same object; because masks are used only for sampling, they do not need to align precisely with object boundaries. Trajectories are represented compositionally through temporal gating, where multiple experts can be active within a segment, reflecting the relational nature of manipulation (e.g., a transfer motion depending on both grasp and

¹Project page: <https://fzaero.github.io/compositional/>

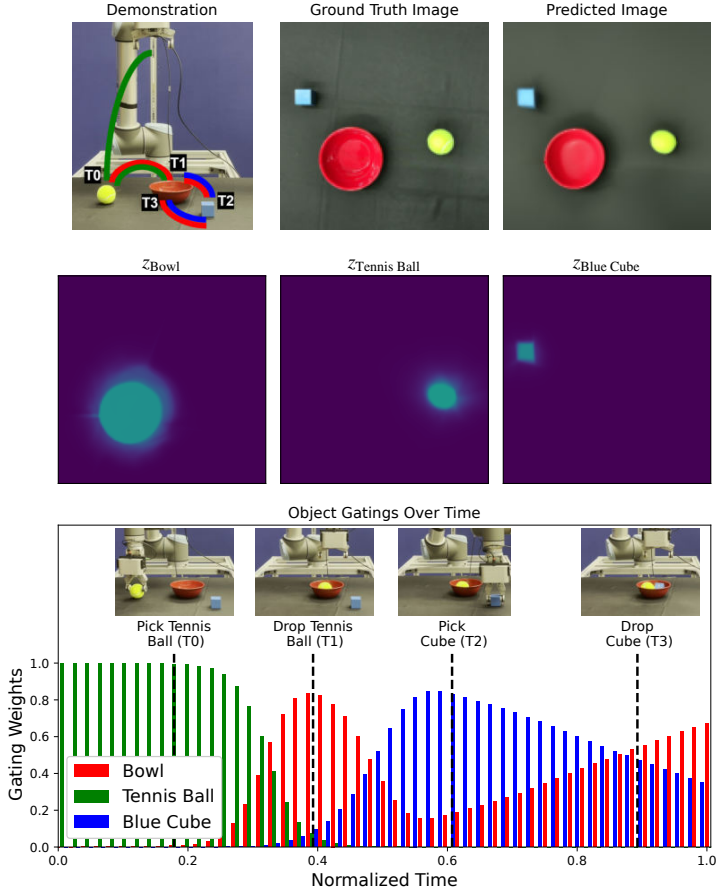


Figure 1: System overview. Top: Task sequence with object interactions, followed by the ground-truth and reconstructed images. Middle: Object-specific masks inferred from latent codes, illustrating object-centric scene decomposition. Bottom: Temporal gating weights showing each object’s influence over the trajectory; snapshots above the plot correspond to key events (T0–T3), marked by vertical dotted lines.

place poses). Together, this spatial–temporal MoE integrates perception and action, preserves the efficiency of primitive-based methods, and enables systematic generalization to novel scenes.

The contributions of this work are as follows. A scene modeling approach is proposed that learns smooth and interpretable object-centric representations, paired with a motion modeling approach that enables sequential compositionality through temporal gating across multiple experts. Building on these approaches, a data-efficient LfD framework is introduced that integrates compositional scene and motion modeling to learn movement primitives directly from visual input, supporting generalization from few demonstrations. The framework is validated in both simulation and real-world experiments, demonstrating improved efficiency and generalization compared with existing baselines.

2 Related Work

Movement primitives (MPs) are a dominant representation in LfD [1], providing smooth, low-dimensional encodings of trajectories. Extensions include probabilistic [5], [8], task-parameterized [6], [10], [11], and via-point-based formulations [7]. Despite their effectiveness, compositionality is usually imposed manually, and MPs typically rely on hand-crafted features. Neural field-based formulation has also been explored for scene- and motion-level modeling [9], but this approach operates at the scene level and scales poorly with the number of objects or scene parameters. Our framework addresses these limitations by modeling scenes and motions in a generative and compositional manner, thereby preserving the data efficiency of MPs. We further demonstrate category-level trajectory generation using object-level segmentation.

Recent imitation-learning and generative policy approaches, including diffusion-based and transformer-based methods [12], [13], have demonstrated strong performance in robotic manipulation. More recently, Vision-Language-Action (VLA) models and generalist robot policies have explored large-scale cross-task visuomotor learning using pretrained vision-language representations and large robot datasets [14], [15]. Several recent works further investigate object-centric and few-shot adaptation strategies for pretrained VLAs, such as ControlVLA [16], which combines online segmentation with pretrained visuomotor policies for object-conditioned manipulation. In contrast, our work focuses on few-demonstration, task-specific trajectory generation from a single static scene observation, where object-centric latent representations are

inferred from the scene and used to condition full trajectory generation. Our experiments show that this formulation achieves high task performance given limited demonstrations.

Unsupervised scene decomposition methods, including Slot Attention [17] and neural field-based approaches such as GIRAFFE [18], learn object-centric scene representations from images. However, these methods typically rely on large-scale image datasets and are not designed for data-efficient robotic learning. In contrast, our framework learns object-centric representations from substantially fewer images and couples them with compositional motion modeling.

Neural fields [19] provide powerful continuous representations and have been extended beyond reconstruction [20], [21], [22] to robotics tasks. For motion generation, prior work has applied them to correspondence [23], [24], trajectory-level modeling [9] and reactive motion generation [25]. However, existing approaches either model motion only at the waypoint or pose level, focus on motion generation without supporting visual conditioning or sacrifice object-level compositionality and data efficiency. In contrast, we introduce a neural field-based framework for compositional scene and motion modeling at the object level, enabling efficient full-trajectory generation from visual input.

3 Problem Formulation

We focus on learning to generate motion from demonstrations, leveraging compositional representations of scenes and motion to enable generalization from a few examples. The training data consist of a set of images with coarse object masks and corresponding motion trajectories:

$$\mathcal{D} = \left\{ \left(I^{(k)}, \{M_i^{(k)}\}_{i=1}^N, \tau^{(k)} \right) \right\}_{k=1}^K, \quad (\text{D.1})$$

where $I^{(k)}$ is the k -th image, $\{M_i^{(k)}\}_{i=1}^N$ are the object masks for N objects, and $\tau^{(k)} = \{q^{(k)}(t)\}_{t=1}^T$ is motion trajectory of length T , where $q^{(k)}(t)$ denotes the robot state at time t , such as the end-effector position. The dataset includes demonstrations that cover the boundary values of the task parameters (e.g., the minimum and maximum admissible object configurations). This ensures that the model observes the full range of object variability present in the task. Each object mask $M_i^{(k)}$ marks the image region of a single object.

The masks should sufficiently cover the target object, but do not need to precisely align with object boundaries². We assume that objects occupy distinct regions and that the background is consistent across demonstrations during training, including a fixed camera viewpoint and consistent scene layout. This makes annotation easier and ensures that image variations reflect object-level changes.

We assume trajectories are approximately time-aligned across demonstrations, so that each index t corresponds to the same task phase (e.g., approach, grasp, transport, release). Such trajectories can be obtained through motion-planned demonstrations or by aligning kinesthetic trajectories based on contact events or changes in motion profiles [11]. This establishes a shared temporal structure across demonstrations and enables sequential compositionality, where different object latent representations influence distinct trajectory segments.

The learning objective is (i) to learn a scene representation model along with latent vectors $\{z_i\}_{i=1}^N$ for each object such that each scene can be expressed as a composition of a fixed background and objects whose variations (e.g., in position or geometry) are captured smoothly in the latent space; and (ii) to train a motion generation model that learns to map these latent vectors to the demonstrated trajectories. In the inference phase, latent vectors for a new scene are extracted using the learned scene representation model and fed into the motion model to generate the corresponding trajectory.

4 Method

The proposed framework is shown in Fig. 2. We first learn an object-centric generative model from demonstration images (Sec. 4.1, Sec. 4.2), yielding latent representations that capture each object in the scene. We then construct a minimal canonical latent set that can equivalently reconstruct the training images (Sec. 4.3). Finally, we model motion compositionally by conditioning a temporal MoE on these canonical latent representations (Sec. 4.4), enabling trajectory generation grounded in object-centric structure.

²In practice, the masks are annotated using one or two rectangles.

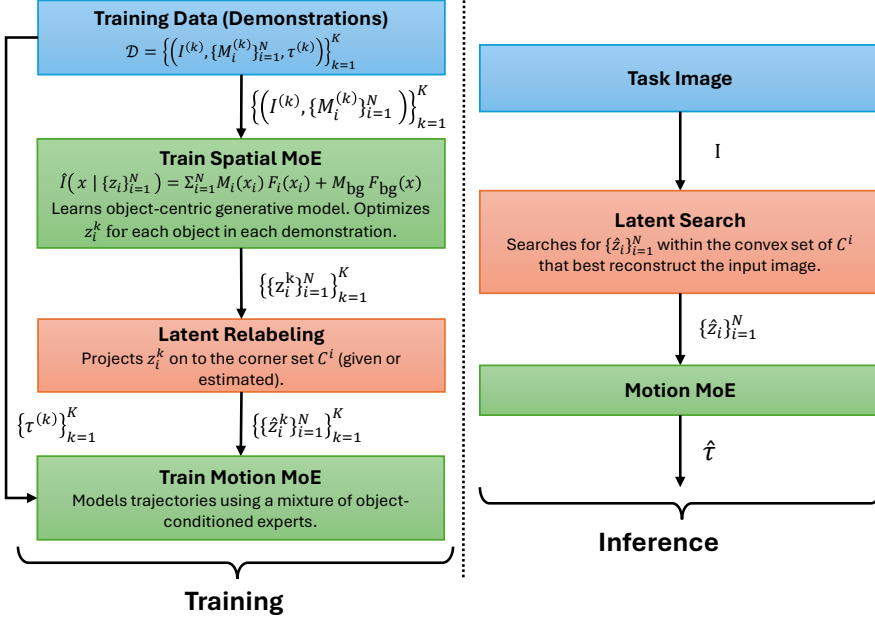


Figure 2: Overview of training and inference pipelines. The Spatial MoE learns object-centric latent vectors, which are canonicalized via latent relabeling and used to train the Motion MoE. During inference, object latents recovered via latent search are used by the Motion MoE to generate trajectories.

4.1 Compositional Image Modeling

We model each image using an object-centric spatial MoE:

$$\hat{f}(x | \{z_i\}_{i=1}^N) = \sum_{i=1}^N M_i(x_i) F_i(x_i) + M_{\text{bg}} F_{\text{bg}}(x), \quad (\text{D.2})$$

with deformed coordinates $x_i = x - \Delta_i(x | z_i)$, where $F_i(x)$ predicts the appearance of object i , $\Delta_i(x | z_i)$ is the deformation conditioned on latent vector z_i , and $M_{\text{bg}}, F_{\text{bg}}$ denote the background mask and appearance model, respectively. All fields F_i, F_{bg} , and deformation Δ_i are implemented as coordinate-based multilayer perceptrons (MLPs), i.e., neural fields. The spatial MoE is shown in Figure 3.

Object masks are computed via a spatial softmax across objects and background:

$$M_i(x_i) = \frac{\exp(l_i(x_i))}{\exp(l_{\text{bg}}) + \sum_{j=1}^N \exp(l_j(x_j))}, \quad (\text{D.3})$$

where the background logit l_{bg} is constant, and each object logit is given by a learned function $l_i(x_i)$. The logit functions are also implemented as coordinate-based MLPs. This formulation yields object masks that partition the image into spatially coherent regions, while deformations capture scene variations as smooth changes in object position and shape.

4.2 Object-Centered Training of Experts

Training the compositional image model proceeds in multiple stages. We first estimate the static background, then pretrain individual object fields using coarse masks, and finally refine all components jointly under the mixture formulation. Optimization in all stages uses an ℓ_2 reconstruction loss between predicted and ground-truth pixels. For deformation networks Δ_i , we additionally employ Lipschitz regularization [26] to encourage the mapping $z_i \mapsto \Delta_i(x | z_i)$ to vary smoothly with latent variables. While the neural-field parameterization already induces a generally smooth dependence on z_i , this regularization further helps by discouraging abrupt local changes and improving stability.

Background Modeling We first train F_{bg} using only non-masked pixels from all demonstrated images. Compared to naïve median-image-based supervision, this procedure provides a more consistent background estimation.

Object-Centric Training Each object model F_i is trained using its corresponding mask within an object-centric sampling framework. During training, pixels that do not belong to other objects are sampled from the demonstration image, with additional importance sampling applied to M_i to emphasize the appearance of the target object. Pixels that fall within the masks of other objects are instead sampled from the background model, preventing F_i from capturing regions associated with different objects. This sampling strategy ensures that variations across scenes are attributed solely to the correct object and enables the learned masks to refine coarse input masks into more

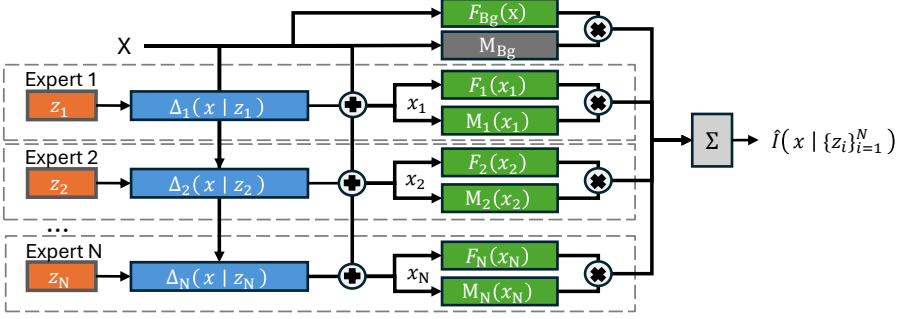


Figure 3: Given N expert-specific objects, orange denotes learnable embeddings, blue corresponds to FiLM-conditioned MLPs, and green represents standard MLPs. Light gray indicates mathematical operators, and dark gray denotes the background mask computed from a constant, non-trainable logit.

precise segmentations.

To further improve training stability, we employ a position-aware zoom-based curriculum around each object mask. This approach is particularly beneficial for smaller objects, as it increases their effective pixel coverage and provides more stable gradients during optimization. The zoom factor is gradually reduced over time, allowing the model to converge toward accurate object representations at their true scale.

Joint Compositional Training Finally, we continue training using the full compositional model in (D.2), with background and object models jointly optimized. The earlier stages act as pretraining, while this stage refines consistency between objects and background. Overall, this object-aware training scheme enables the model to explain variations across scenes as object-dependent differences, allowing robust generalization with as few as 10–30 training images.

Extension to 3D Representations While we describe F_i as 2D appearance fields, the mixture formulation is not restricted to images. Each F_i can also be defined as an occupancy field [22] over 3D space, enabling the construction of a compositional 3D scene representation.

Input: Reference image I , corner set C^i for object i , number of samples P , preselection size L' , maximum candidate count L , error threshold λ

Output: Top- L latent vector candidates $\{\hat{z}_{i,l}^{(k)}\}_{l=1}^L$

- 1 Sample P weight vectors $\{\alpha_p\}_{p=1}^P \sim \text{Dirichlet}(|C^i|)$;
- 2 **for** $p = 1$ **to** P **do**
- 3 $\hat{z}_{i,p}^{(k)} = \sum_j \alpha_{p,j} c_{i,j}, \quad c_{i,j} \in C^i$;
- 4 $E_{i,p}^{(k)} = \|I - \hat{I}(\cdot \mid \hat{z}_{i,p}^{(k)})\|_2$
- 5 **end**
- 6 Select the top- L' candidates with the lowest reconstruction errors;
- 7 Refine these candidates via gradient descent on $E_{i,l}^{(k)}$;
- 8 $E_{\min} = \min_l E_{i,l}^{(k)}$;
- 9 Retain all candidates satisfying $E_{i,l}^{(k)} < \lambda E_{\min}$, up to L total;
- 10 **return** $\{\hat{z}_{i,l}^{(k)}\}_{l=1}^L$;

Algorithm 1: Latent Search

4.3 Latent Relabeling

Each object is associated with a latent vector $z_i^{(k)} \in \mathbb{R}^{d_z}$ for each demonstration k . Smooth object variations are captured by interpolating between a small number of endpoint latent vectors that span the boundaries of the observed variation. For each object i , these endpoints form the corner set C^i . Given this set, the search (Algorithm 1) recovers object-specific latent representations for each demonstration by sampling convex combinations of the corners and refining the lowest-error candidates through gradient-based optimization. As multiple convex combinations can yield similar reconstructions, we retain the top- L candidates per demonstration to preserve valid alternatives for downstream motion modeling. This effectively increases the training set size from K to $K \times L$.

We estimate the corner sets by iteratively expanding C^i . We initialize C^i with two embeddings that are maximally distant within the set $\{z_i^{(k)}\}_{k=1}^K$ and use Algorithm 1 to test whether the remaining instances can be reconstructed from the current corner set. For any instance that cannot be reconstructed, we expand C^i by selecting the embedding most distant from the latents sampled via convex combinations of the current corner set, leveraging the observation that larger visual differences are generally reflected by greater distances in the latent space. This process iteratively widens the representable region of

the latent space until all training instances can be reconstructed from the corner set, ensuring that every demonstration latent can be canonicalized as a convex combination of the corners. As a result, generalization is naturally constrained to variations represented within the demonstrated latent support, and extrapolation beyond this region is not guaranteed.

4.4 Motion Generation

Given the relabeled object latent vectors $\{\hat{z}_i\}_{i=1}^N$, we assume sequential compositionality: different trajectory segments depend on different subsets of $\hat{z}_i$. At an abstract level, we model this through a time-dependent conditioning signal constructed via softmax mixture weights. The exact conditioning mechanism is specified later in Sec. 4.5.

$$c(t) = \sum_{i=1}^N w_i(t) c_i(\hat{z}_i), \quad w_i(t) = \frac{\exp(s_i(t))}{\sum_{j=1}^N \exp(s_j(t))} \quad (\text{D.4})$$

where $s_i(t)$ are learnable relevance scores and $c_i(\hat{z}_i)$ denotes object-specific conditioning variables. The generator then predicts a trajectory in the state space $q(t) \in \mathbb{R}^d$ as

$$q(t) \sim \mathcal{N}(\mu(t), \text{diag}(\sigma^2(t))). \quad (\text{D.5})$$

where $\mu(t)$ and $\sigma(t)$ are estimated via

$$[\mu(t), \sigma(t)] = f(t \mid c(t)), \quad (\text{D.6})$$

where $\sigma(t)$ is constrained to be positive via a softplus transform applied to the network output. The motion generation framework is shown in Figure 4.

The model parameters are optimized by minimizing the negative log-likelihood of the demonstrated trajectories, where each trajectory is modeled as a Gaussian:

$$\mathcal{L}_{\text{traj}} = - \sum_{k,t} \log \mathcal{N}(q^{(k)}(t); \mu(t \mid c(t)), \Sigma(t \mid c(t))), \quad (\text{D.7})$$

where $\Sigma(t \mid c(t)) = \text{diag}(\sigma^2(t \mid c(t)))$. To discourage the model from learning spurious dependencies on irrelevant latent variables, we apply latent dropout during early training by randomly masking a subset before pooling. This

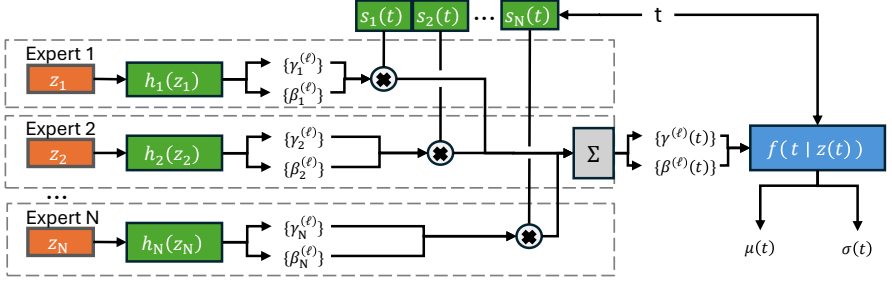


Figure 4: Given N expert-specific object embeddings, FiLM weights and object relevance scores are predicted via MLPs (green). These are pooled to generate FiLM parameters used to condition the trajectory generation.

warm-up regularization encourages the generator to rely primarily on informative variables, while in later training and at test time, all variables are used. Additionally, the probabilistic trajectory formulation mitigates overfitting to spurious correlations by allowing irrelevant latent variables to be absorbed into the variance term rather than biasing the mean.

Modeling Orientation For modeling orientation, we follow [27] and represent the quaternion trajectories $q_{1:T}$ in axis-angle form $r_{1:T}$, expressed in the tangent space of the initial quaternion q_1 , yielding continuous rotation trajectories that can be modeled similarly to Cartesian trajectories.

4.5 Latent Conditioning with FiLM

Both the image model and the trajectory generator are conditioned on latent vectors. We implement this conditioning through *Feature-wise Linear Modulation (FiLM)* [28], which reconfigures the parameters of a base network using latent-dependent scale and shift terms. For a given z_i , FiLM produces parameters

$$h_i(z_i) = \{\gamma_i^{(\ell)}, \beta_i^{(\ell)}\}_{\ell=1}^L, \quad (\text{D.8})$$

where $\gamma_i^{(\ell)}, \beta_i^{(\ell)} \in \mathbb{R}^H$ for layer width H . Given base layer parameters $(w^{(\ell)}, b^{(\ell)})$,

the modulated parameters are

$$\tilde{w}^{(\ell)} = \gamma^{(\ell)} \odot w^{(\ell)}, \quad \tilde{b}^{(\ell)} = \gamma^{(\ell)} \odot b^{(\ell)} + \beta^{(\ell)}, \quad (\text{D.9})$$

with $\odot$ denoting elementwise multiplication. FiLM thus provides a lightweight mechanism to adapt a shared architecture to different objects or tasks. This role is analogous to hypernetworks [29], which generate full network weights from latent codes, but FiLM achieves similar conditioning capability with fewer parameters and improved stability.

In the compositional image model, FiLM parameters are derived from each z_i and applied directly when predicting object deformations. In the trajectory generator, the conditioning variables in (D.4) correspond to the FiLM modulation parameters, i.e., $c_i(\hat{z}_i) \equiv \{\gamma_i^{(\ell)}, \beta_i^{(\ell)}\}_{\ell=1}^L$. Accordingly, the aggregated conditioning signal is implemented through

$$\gamma^{(\ell)}(t) = \sum_{i=1}^N w_i(t) \gamma_i^{(\ell)}, \quad \beta^{(\ell)}(t) = \sum_{i=1}^N w_i(t) \beta_i^{(\ell)}. \quad (\text{D.10})$$

This mechanism realizes sequential compositionality: different trajectory segments are influenced by different subsets of z_i , depending on which $w_i(t)$ are active.

4.6 Inference

At test time, we apply the same latent search procedure (Algorithm 1) using the corner sets from training. Objects are processed sequentially, with each contribution added incrementally to condition subsequent reconstructions. In rare cases where visually similar objects (e.g., of the same color) lead to ambiguity, larger objects are prioritized to provide more reliable reconstruction cues. Since reconstruction is based on visual similarity to the learned object representations, the method tolerates moderate lighting variation and visually distinct distractor objects. During latent inference, representations that better reconstruct the learned task-relevant object structure are favored, allowing distractor objects that are not well represented by the learned object model to be ignored. However, distractors that are visually similar to task-relevant objects may still lead to ambiguity or degraded performance. This procedure recovers object-specific latent representations for novel task instances while

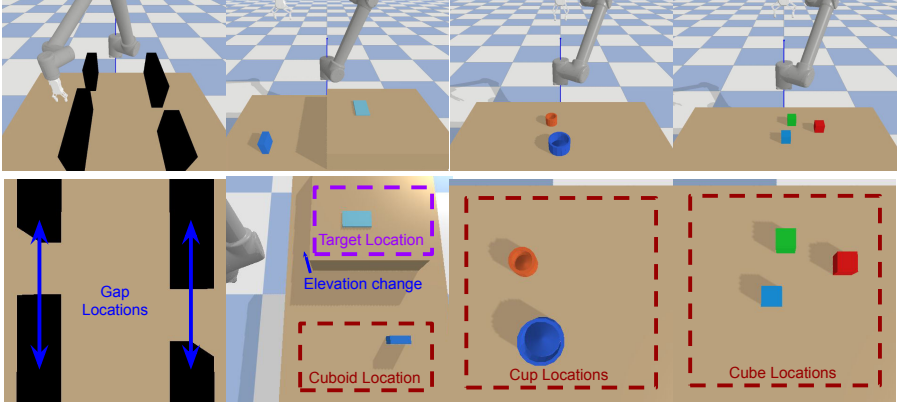


Figure 5: Simulation tasks with corresponding scene variations. The bottom row shows the camera images used by the methods. In these images, arrows highlight geometric changes and rectangles mark possible object locations.

remaining within a compact and well-structured region of the latent space. The resulting latent representations then condition the trajectory generator. In the current implementation, running on an Intel(R) Xeon(R) W-2245 CPU @ 3.90GHz and an NVIDIA Quadro RTX 6000 GPU, inference takes approximately 1.4 and 0.8 seconds per object for RGB and 3D settings, respectively ³.

5 Experiments

We evaluate the framework through simulation benchmarks and real-world robotic experiments. The simulation studies show how compositional motion modeling through object-centric latent representations and time-varying latent aggregation improves generalization and data efficiency. Real-world experiments demonstrate the method’s ability to handle perception noise, integrate 3D scene representations, and enable category-level generalization through vision-language-based segmentation.

³3D inference employs sparsely sampled coordinates near surfaces instead of the full coordinate space.

5.1 Simulation Experiments

We first assess the proposed method on four manipulation tasks of increasing complexity. In Wall Avoidance (easy), the robot reaches a goal by navigating through openings in two walls. Incline Pick-and-Place (medium) requires transporting a cuboid onto a platform at varying elevations, with both the cuboid and target locations changing across trials. Cup Stacking (medium) involves inserting a small cup into a larger one, while Cube Stacking (hard) requires stacking three cubes from diverse initial positions. These tasks introduce scene variations including geometric changes and varying object placements, as illustrated in Figure 5.

We compare the proposed method against several baselines, including conditional neural movement primitives (CNMP) [6] with CNN and Diffusion Policy (DP) [12]. CNMP supports image-conditioned trajectory generation, making it one of the closest comparable approaches for our setting. DP is a recent imitation-learning framework that has demonstrated strong performance in robotic manipulation ⁴. To evaluate the proposed perception pipeline, we include CNN-FiLM and DINO-FiLM ablations that replace the object-centric latent representation with global image features. CNN-FiLM uses end-to-end learned CNN features, while DINO-FiLM uses pretrained DINO [30] features. We further include an ablation without temporal gating (“Ours w/o gating”) to isolate the contribution of time-varying latent aggregation. Finally, we include a modified Neural Field Movement Primitive baseline (NFMP) [9], where we replace the hypernetwork with FiLM layers and omit positional encodings for a fairer comparison. Since NFMP relies on grid-based training over all combinations of task parameters, scaling as 3^n with n task parameters, it is evaluated only on the Wall Avoidance task, where the required number of demonstrations remains tractable.

Performance is evaluated using three criteria: mean Euclidean distance (MED) between predicted and demonstrated trajectories, task success rate, and data efficiency across varying training sizes. All models are trained with three random seeds, and results are reported as the mean and standard deviation across seeds. We consider two data regimes: a low-data regime with limited demonstrations per task, and a high-data regime with a larger demonstration set. Baselines are evaluated in both regimes, whereas the pro-

⁴To align the prediction setting across methods, normalized time is provided as a state input to DP.

Table 1: Simulation results (MED/Accuracy).

Method (data regime)	Wall Avoidance	Incline Pick-and-Place	Cup Stacking	Cube Stacking
CNN-CNMP (Low)	2.72±0.08/0.67±0.04	4.35±0.27/0.11±0.03	10.85±1.39/0.02±0.01	14.22±0.59/0.00±0.00
CNN-CNMP (High)	1.70±0.26/0.95±0.02	2.39±0.43/0.51±0.15	3.08±0.27/0.16±0.07	6.07±0.32/0.00±0.00
DP (Low)	2.31±0.04/0.79±0.02	3.60±0.08/0.25±0.01	11.13±1.29/0.02±0.02	11.55±0.19/0.00±0.00
DP (High)	0.55±0.03/ 0.99±0.00	1.49±0.14/0.86±0.04	2.72±0.09/0.26±0.04	4.77±0.61/0.11±0.03
NFMP (Grid)	0.85±0.18/0.98±0.01	NA	NA	NA
DINO-Film (Low)	4.46±0.19/0.65±0.02	5.03±0.13/0.21±0.00	9.81±0.34/0.00±0.00	15.10±0.07/0.00±0.00
DINO-Film (High)	2.27±0.12/0.84±0.00	3.71±0.06/0.31±0.06	6.69±0.06/0.01±0.00	10.48±0.08/0.00±0.00
CNN-Film (Low)	1.41±0.19/0.89±0.02	3.01±0.55/0.40±0.13	8.74±1.83/0.03±0.01	9.26±0.74/0.00±0.00
CNN-Film (High)	0.46±0.09/0.99±0.00	1.18±0.09/0.92±0.05	2.11±0.20/0.26±0.06	1.32±0.03/0.66±0.10
Ours w/o gating (Low)	0.71±0.04/ 0.99±0.00	0.89±0.03/0.99±0.01	1.10±0.03/0.86±0.05	1.33±0.05/0.85±0.06
Ours (Low)	0.51±0.03/0.99±0.00	0.71±0.02/1.00±0.00	0.65±0.04/0.99±0.00	0.72±0.05/0.97±0.02

Low/High training sizes per task: Wall Avoidance (10/30), Incline Pick-and-Place (20/50), Cup Stacking (20/50), and Cube Stacking (30/100). Bold numbers indicate the best performance in each task under the low- and all-data regimes, respectively. Lower MED and higher accuracy indicate better performance.

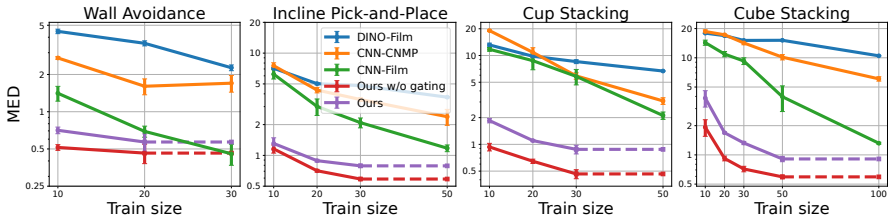


Figure 6: Training size versus MED, illustrating the data efficiency of the proposed approach. The proposed method achieves low error with far fewer demonstrations than the baselines, particularly on more challenging tasks.

posed method is evaluated only in the low-data regime, demonstrating that it matches or outperforms baselines trained with substantially more demonstrations.

Results are presented in Table 1. Across tasks, the proposed method, both with and without object-centric temporal gating, consistently outperforms baselines in the low-data regime. The CNN-FiLM baseline generally achieves higher performance than CNN-CNMP and Diffusion Policy, suggesting that FiLM-conditioned trajectory generation itself provides a strong inductive bias in our setting. Even when trained with many more demonstrations, baseline performance is generally comparable to, or lower than, that of the proposed

model. Comparing CNN-FiLM with the proposed method without temporal gating further highlights the contribution of the proposed object-centric representation. The only exception is the Wall Avoidance task, where CNN-FiLM achieves a marginally lower MED when trained with three times as many demonstrations. However, both methods successfully avoid the wall, and the relatively low task complexity reduces the data-efficiency advantage of the proposed method. To further illustrate data efficiency, Figure 6 shows MED as a function of training size, where each point represents the mean and standard deviation across seeds. The proposed method achieves consistently low error using few demonstrations, while baselines require significantly more data to reach similar performance, particularly on more challenging tasks. The figure also includes an ablation where F_{bg} is trained using median-image-based supervision. While the average performance remains broadly comparable, the median-image-based variant occasionally exhibits less stable behavior and larger error cases.

Finally, we evaluated robustness to incomplete masks and imperfect temporal alignment on the Cube Stacking task across three seeds. The method maintained MED comparable to the original setup when up to 3 out of 30 demonstrations contained incomplete masks, while larger numbers prevented reliable object decomposition during scene-model training. For temporal alignment, we introduced temporal warping within sub-task segments while preserving key task phases. This increased the mean MED to 2.03 and reduced the task success rate to approximately 0.90, suggesting that the method retains reasonable performance under moderate intra-phase timing variation while still relying on coarse phase alignment across demonstrations.

5.2 Real-Robot Experiments

Validation of the framework is performed on four real-world robot tasks. These experiments demonstrate the method’s robustness to perception noise, its ability to generalize across object categories using segmentation masks obtained from vision-language models, and its compatibility with 3D scene representations. These tasks are shown in Figure 7. The demonstrations are six-degree-of-freedom trajectories collected via kinesthetic teaching, except for the fourth task, where motion planning is used due to the complexity of the required movements. All demonstration trajectories are temporally aligned using gripper-state transitions (open/close), which serve as consistent seman-

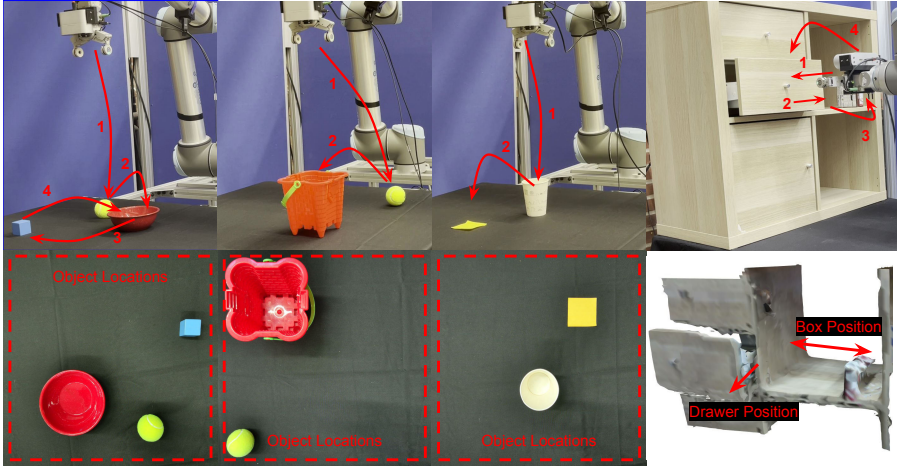


Figure 7: Real-world task and scene variations. The top row illustrates the task steps, and the bottom row provides sample inputs, with rectangles and arrows marking possible scene parametrization. As occupancy values are hard to visually discern, we show the underlying mesh for the final task.

tic markers across demonstrations. During testing, we simplify motion execution by segmenting the predicted trajectories at gripper-state transitions (detected via rising and falling edges). Each sub-trajectory is then executed sequentially, and the corresponding gripper command is issued at the transition point. Trajectory execution is performed using MoveIt [31]. A small validation set is used for early stopping during the training of the motion generation model.

In the first task, the robot picks up tabletop objects and places them in a bowl, with varying locations across trials. It requires approximately 1–2 cm horizontal grasping accuracy to reliably grasp the tabletop objects. With 30 training demonstrations, the system successfully completes all 25 test cases ⁵. We further evaluate robustness by adding visually distinct distractor objects in 5 additional test cases, all completed successfully. Representative input images and their corresponding reconstructions for this task are shown in Figure 8.

⁵With 20 training demonstrations, it completes 3 out of 5 additional trials. Failures occur when grasping the blue cube, which requires higher precision.

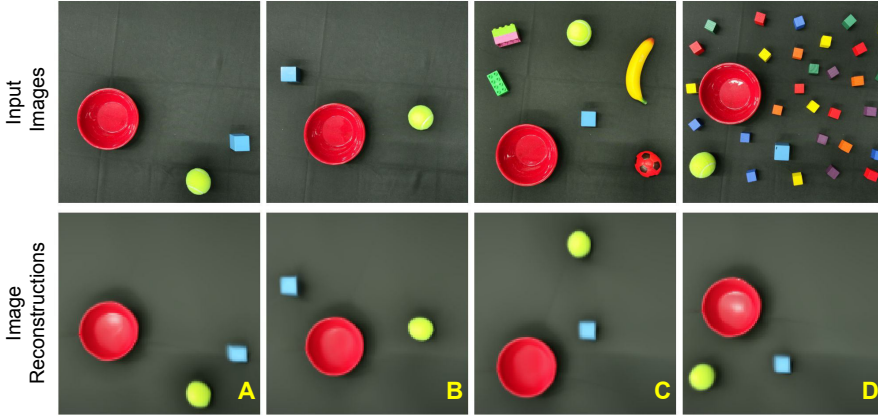


Figure 8: Input Images, and corresponding image reconstructions with (C, D) and without (A, B) distractor objects for the first task.

The second and third tasks evaluate category-level generalization using language-guided segmentation models. In the second task, the robot picks up a ball and places it in a container; in the third, it picks up a cup and places it at a target location marked by a piece of paper. Reliable grasping requires approximately 2 cm and 1 cm horizontal accuracy, respectively. We use a language-based segmentation algorithm [32] to detect object masks. The prompts “ball” and “box” (Task 2), and “cup” and “rectangle paper” (Task 3) are used to obtain segmentation masks, which are then used to generate colorized images and object masks. During training, the same objects are used across demonstrations. The models for the second and third tasks are trained with 20 and 30 demonstrations, respectively. At test time, we additionally evaluate on unseen objects from the same semantic categories. For the third task, these include two mugs and a watering can with similar cylindrical geometry and a top opening, allowing the demonstrated grasping strategy to transfer without modification. Figure 9 illustrates the perception pipeline, showing input images, language-conditioned colorized masks, and the corresponding reconstructions.

We evaluate both tasks on 15 test cases each. In the second task, the system succeeds in 14 out of 15 trials, while in the third task, it completes all test cases. The single failure in the second task occurs when testing with a

previously unseen baseball, for which the model’s predicted grasp was slightly misaligned; the harder surface of the baseball reduces compliance and imposes stricter precision requirements on the grasp. Under identical test conditions, but using the training object (a tennis ball) instead of the baseball, the task succeeds, as the tennis ball’s softer, more compliant surface reduces the precision required for successful grasping. We additionally evaluate a multi-object, language-conditioned setting for both tasks. For each trial, a language prompt (e.g., “tennis ball,” “blue ball” for Task 2; “red paper,” “playing card,” “glass cup,” “mug” for Task 3) specifies the target object. These prompts are used to obtain segmentation masks and corresponding colorized images⁶. The model then generates a trajectory for the specified object. We test this over five trials per task, and in each scenario, we perform three executions—each using different prompts and colorized images. Figure 10 shows representative scenes, prompts, colorized images, and reconstructions used for trajectory generation. Across all trials, the system successfully completes all three pick-and-place executions.

In the final task, the robot places a carton box inside a drawer. This task requires opening the drawer, retrieving the box from a neighboring compartment, and placing it inside the drawer, while the initial drawer opening and the box’s initial position vary across trials. Grasping the drawer handle requires about 1 cm accuracy. This task tests the method’s ability to generalize across full three-dimensional geometry and variations in scene surfaces. To represent the scene, we scan it from several predefined camera poses and estimate a Euclidean Signed Distance Field (ESDF) [33], which is then converted into an occupancy field by marking locations with $\text{ESDF} < \epsilon$ (with $\epsilon = 0.02$) as occupied. The model is trained using four such occupancy fields paired with their corresponding motion trajectories⁷. Evaluation on ten novel test configurations results in successful task completion in all cases. This level of generalization from only a few demonstrations is enabled by the object-centric representation, which captures the drawer opening and box position through interpolation between two latent corner vectors for each factor (open and closed for the drawer, and leftmost and rightmost positions for the box). Figure 11 shows the meshes acquired from scene scans together with the corre-

⁶This evaluation assumes that the language prompt produces a correct segmentation mask for the intended target object.

⁷With fewer than four examples, the scene representation model is unable to capture the drawer’s articulation.

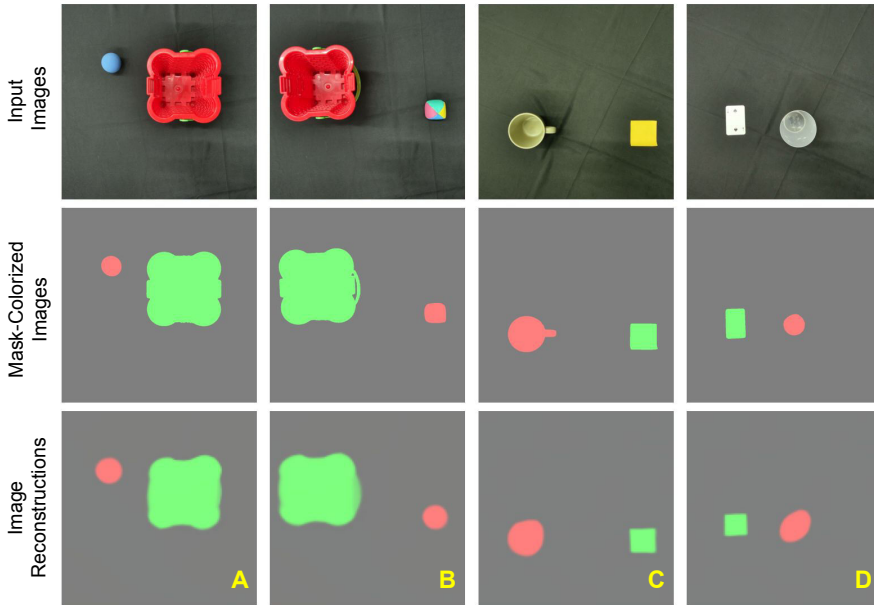


Figure 9: Illustration of the perception pipeline: input images, language-conditioned mask colorizations produced by the vision-language model, and the resulting image reconstructions.

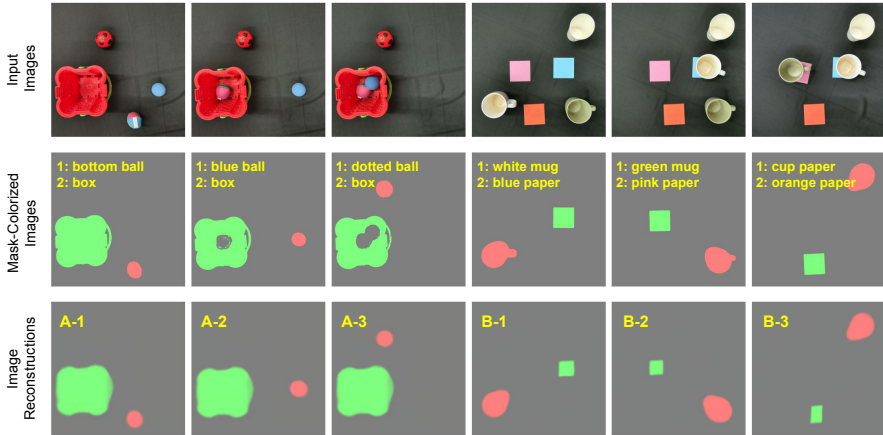


Figure 10: Example multi-step trials for the second (A-1, A-2, A-3) and third (B-1, B-2, B-3) tasks. Language prompts in the second row allow the user to specify the object to be manipulated.

sponding reconstructions obtained from the predicted occupancy fields, along with snapshots from task execution. The reconstructed meshes accurately reflect the positions of both the drawer and the box. The snapshots highlight the key poses that result in successful task execution. We further evaluate the model on eight held-out demonstrations collected under the same conditions as the training data. The average position error at key drawer-handle and box-grasp poses is 0.32 cm and 0.35 cm, respectively. The measured prediction errors indicate that the learned model maintains effective generalization across previously unseen configurations.

6 Conclusion

This work presents a compositional framework for motion generation directly from visual demonstrations by coupling object-centric neural representations with a temporal mixture-of-experts model. The spatial mixture-of-experts learns smooth and compact latent representations that capture object-level variability from a small number of images, while the temporal mixture-of-experts exploits these latent representations to model motion as a sequence

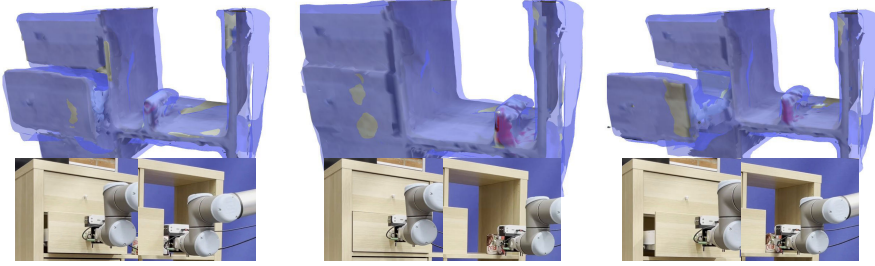


Figure 11: Input meshes and reconstructions produced by the proposed method. Snapshots from robot motion corresponding to key grasp poses are shown right below the reconstructions. The reconstructed surfaces (shown in blue) capture the drawer openings and box positions, supporting motions that enable successful grasps of the drawer handle and the box.

of object-conditioned primitives. The results demonstrate that compositional modeling at the object level provides a powerful paradigm for learning from demonstration, offering interpretable latent structure, data-efficient motion generation, and systematic generalization in both simulation and real-world environments. To support broader and more flexible generalization at the scene level, the framework will be extended to incorporate neural radiance field-based scene representations.

References

- [1] S. Schaal, “Learning from demonstration,” *Advances in neural information processing systems*, vol. 9, 1996.
- [2] R. R. Burridge, A. A. Rizzi, and D. E. Koditschek, “Sequential composition of dynamically dexterous robot behaviors,” *Int. J. Robot. Res.*, vol. 18, no. 6, pp. 534–555, 1999.
- [3] Y. Du and L. P. Kaelbling, “Position: Compositional generative modeling: A single model is not all you need,” in *Int. Conf. Mach. Learn. (ICML)*, 2024.

- [4] 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.
- [5] A. Paraschos, C. Daniel, J. R. Peters, and G. Neumann, “Probabilistic movement primitives,” *Advances in neural information processing systems*, vol. 26, 2013.
- [6] M. Y. Seker, M. Imre, J. H. Piater, and E. Ugur, “Conditional neural movement primitives.,” in *Robotics: Science and Systems*, vol. 10, 2019.
- [7] 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.
- [8] G. Li, Z. Jin, M. Volpp, F. Otto, R. Lioutikov, and G. Neumann, “Prodmp: A unified perspective on dynamic and probabilistic movement primitives,” *IEEE Robot. Autom. Lett.*, vol. 8, no. 4, pp. 2325–2332, 2023.
- [9] A. Tekden, M. P. Deisenroth, and Y. Bekiroglu, “Neural field movement primitives for joint modelling of scenes and motions,” in *IEEE/RSJ Int. Conf. Intell. Robot. Syst. (IROS)*, 2023, pp. 3648–3655.
- [10] Y. Yildirim and E. Ugur, “Conditional neural expert processes for learning movement primitives from demonstration,” *IEEE Robot. Autom. Lett.*, 2024.
- [11] S. Calinon, “A tutorial on task-parameterized movement learning and retrieval,” *Intelligent service robotics*, vol. 9, pp. 1–29, 2016.
- [12] C. Chi et al., “Diffusion policy: Visuomotor policy learning via action diffusion,” *The International Journal of Robotics Research*, vol. 44, no. 10–11, pp. 1684–1704, 2025.
- [13] T. Zhao, V. Kumar, S. Levine, and C. Finn, “Learning fine-grained bimanual manipulation with low-cost hardware,” *Robotics: Science and Systems XIX*, 2023.
- [14] B. Zitkovich et al., “Rt-2: Vision-language-action models transfer web knowledge to robotic control,” in *Conference on Robot Learning*, PMLR, 2023, pp. 2165–2183.

-
- [15] Y. J. Ma et al., “Vision language models are in-context value learners,” in *International Conference on Learning Representations*, vol. 2025, 2025, pp. 33 984–34 009.
 - [16] P. Li et al., “Controlvla: Few-shot object-centric adaptation for pre-trained vision-language-action models,” in *Conference on Robot Learning*, PMLR, 2025, pp. 1898–1913.
 - [17] F. Locatello et al., “Object-centric learning with slot attention,” *Adv. Neural Inf. Process. Syst. (NeurIPS)*, vol. 33, pp. 11 525–11 538, 2020.
 - [18] M. Niemeyer and A. Geiger, “Giraffe: Representing scenes as compositional generative neural feature fields,” in *IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, 2021, pp. 11 453–11 464.
 - [19] Y. Xie, T. Takikawa, S. Saito, et al., “Neural fields in visual computing and beyond,” *Computer Graphics Forum*, 2022, ISSN: 1467-8659.
 - [20] 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.
 - [21] 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.
 - [22] 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.
 - [23] 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.
 - [24] A. Tekden, M. P. Deisenroth, and Y. Bekiroglu, “Grasp transfer based on self-aligning implicit representations of local surfaces,” *IEEE Robot. Autom. Lett.*, vol. 8, no. 10, pp. 6315–6322, 2023.
 - [25] A. Tekden, D. Kanoulas, A. Billard, and Y. Bekiroglu, “Reactive motion generation via phase-varying neural potential functions,” *IEEE Robotics and Automation Letters*, 2026.

- [26] H.-T. D. Liu, F. Williams, A. Jacobson, et al., “Learning smooth neural functions via lipschitz regularization,” in *ACM SIGGRAPH 2022 Conference Proceedings*, 2022, pp. 1–13.
- [27] A. Ude, B. Nemec, T. Petrić, and J. Morimoto, “Orientation in cartesian space dynamic movement primitives,” in *IEEE Int. Conf. Robot. Autom. (ICRA)*, 2014, pp. 2997–3004.
- [28] E. Perez, F. Strub, H. De Vries, V. Dumoulin, and A. Courville, “Film: Visual reasoning with a general conditioning layer,” in *Proceedings of the AAAI conference on artificial intelligence*, vol. 32, 2018.
- [29] D. Ha, A. M. Dai, and Q. V. Le, “Hypernetworks,” in *International Conference on Learning Representations*, 2017.
- [30] O. Siméoni et al., *DINOv3*, 2025.
- [31] D. T. Coleman, I. A. Sukan, S. Chitta, and N. Correll, “Reducing the barrier to entry of complex robotic software: A moveit! case study,” *J. Softw. Eng. Robot.*, vol. 5, no. 1, pp. 3–16, 2014.
- [32] L. Medeiros, *Lang segment anything*, GitHub repository, <https://github.com/luca-medeiros/lang-segment-everything>, 2023.
- [33] A. Millane et al., “Nvblox: Gpu-accelerated incremental signed distance field mapping,” in *IEEE Int. Conf. Robot. Autom. (ICRA)*, 2024, pp. 2698–2705.