Robotic Grasping via Implicit Policies on Neural Radiance Fields

Zur Erlangung des akademischen Grades eines

Doktor der Ingenieurwissenschaften

von der KIT-Fakultät für Informatik des
Karlsruher Instituts für Technologie (KIT)

genehmigte

Dissertation

von

Gergely Sóti

Tag der mündlichen Prüfung:	26.01.2026
1. Referent:	Prof. Dr.-Ing. habil. Björn Hein
2. Referent:	Prof. Dr. rer. nat. Sven Behnke

Abstract

This dissertation introduces a framework for robotic grasping that leverages Neural Radiance Fields (NeRF) as scene representations for implicit grasp policies. The central contribution of this study is the demonstration that geometric understanding acquired through volumetric rendering for novel view synthesis can be effectively transferred to manipulation tasks, thereby enabling grasp planning through continuous optimization in $SE(3)$.

The proposed framework extends an image-conditioned NeRF to multi-camera robotic setups through multi-view VisionNeRF (mVNeRF) and introduces Support Pose Decomposition, to enable the trasfer of NeRF's 5-DoF query space to full 6-DoF gripper poses. Grasp planning is posed as an optimization problem: an implicit policy that performs gradient ascent on a learned grasp value function that serves as the objective function. The approach is further improved via trajectory-informed learning, which incorporates demonstrated trajectories to shape the objective function's landscape.

Comprehensive experiments are conducted to validate the framework across simulation and real-world scenarios. The system demonstrates a 71% success rate in real-world grasping experiments, despite being trained exclusively on simple simulated objects, and exhibits strong zero-shot sim-to-real transfer without extensive domain randomization, adaptation or augmentation techniques. Extensions demonstrate the framework's versatility through federated NeRF training and CLIP-based vision-language conditioning.

The fundamental insight is that the volumetric rendering loss, which enforces consistency with the fundamental physics of light transport and camera projection, embeds geometric understanding that proves valuable for manipulation tasks. This

development establishes a novel paradigm for leveraging advances in computer vision in the field robotics, demonstrating that this geometric understanding can be transferred effectively across domains. The work opens pathways for developing more capable and generalizable robotic systems that can understand and interact with complex, unstructured environments.

Kurzfassung

In der vorliegenden Arbeit wird ein Framework für das Greifen mit Robotern präsentiert, welches Neural Radiance Fields (NeRFs) als Szenendarstellungen für implizite Greifstrategien einsetzt. Der zentrale Erkenntnisgewinn dieser Arbeit ist, dass durch den Erwerb geometrischen Verständnisses mittels volumetrischem Rendering für novel view synthesis eine effektive Übertragung auf Manipulationsaufgaben möglich ist. Dies ermöglicht eine Greifplanung durch kontinuierliche Optimierung in $SE(3)$.

Der vorgeschlagene Ansatz sieht eine Erweiterung bildkonditionierter NeRFs auf Multi-Kamera Roboter-Systeme durch multi-view VisionNeRF (mVNeRF) vor. Zudem wird die Support Pose Decomposition eingeführt, um die 5-DoF Definitionsmenge von NeRF mit 6-DoF Greiferposen zu verbinden. Die Greifplanung wird als eine implizite Strategie formuliert, die Greifkandidaten durch Gradientenanstieg auf einer gelernten Greifwertfunktion optimiert. Der Ansatz wird ferner durch Trajectorien-informiertes Lernen verbessert, das demonstrierte Trajektorien einbezieht und so die Landschaft der Zielfunktion formt.

Umfassende Experimente wurden durchgeführt, um das System in Simulationen und realen Szenarien zu validieren. Das System weist eine Erfolgsrate von 71% in realen Greifversuchen auf, obwohl es ausschließlich an einfachen simulierten Objekten trainiert wurde, und zeigt einen starken Zero-Shot Sim-to-Real-Transfer ohne umfangreiche Domänenrandomisierung oder Augmentierungstechniken. Erweiterungen demonstrieren die Vielseitigkeit des Frameworks durch föderiertes NeRF-Training und CLIP-basierte Vision-Language-Konditionierung.

Die grundlegende Erkenntnis besteht darin, dass der volumetrische Rendering-Loss, welcher die Konsistenz mit der fundamentalen Physik des Lichttransports

und der Kameraprojektion erzwingt, ein geometrisches Verständnis ermöglicht, das sich für Manipulationsaufgaben als vorteilhaft erweist. Diese Entwicklung etabliert ein neues Paradigma für die Nutzung von Fortschritten in der Computer Vision im Bereich der Robotik, in dem dieses geometrische Verständnis effektiv über Domänen hinweg transferiert werden kann. Die vorliegende Arbeit leistet einen Beitrag zur Entwicklung leistungsfähigerer und verallgemeinerbarer Robotersysteme, die in der Lage sind, komplexe, unstrukturierte Umgebungen zu verstehen und mit diesen zu interagieren.

Preface

Completing this PhD has been a remarkable journey. One that, I imagine, resonates with anyone who has undertaken doctoral studies. It has been a time of profound learning, growth, and discovery across many different areas of science and life. The path has been filled with moments of excitement and breakthrough, with perhaps more than necessary periods that were exhausting and demanded an extraordinary amount of work.

I am deeply grateful for the unwavering support of my now fiancée, Dóri, especially during the more challenging times. She endured these difficulties alongside me with far more grace than I possessed, providing strength and encouragement when I needed it most. I am also thankful to my family and friends, whose presence has shaped the person I have become and without whom I would not have been able to come this far.

I would especially like to thank my friend and colleague Ilshat Mamaev, who also supervised my master's thesis. He introduced me to the world of robotics with just the right balance of guidance and freedom, allowing me to explore while providing the support I needed. This approach ultimately helped me find the research niche that proved to be the perfect fit for my interests and abilities.

I am equally grateful to my PhD supervisor, Björn Hein, without whom this work would not have been possible at all. I appreciate his trust in allowing me to venture into yet unprobed directions and his willingness to support my exploration of novel research paths.

I would also like to express my gratitude to my immediate colleagues, who have been excellent companions and discussion partners throughout these years. Their

thoughtful criticism and insights have been invaluable in helping me progress and refine my work.

Finally, while it may seem obvious, I am profoundly thankful to the entire research community. The brilliant and inspiring ideas published by countless researchers form the foundation upon which my entire work builds. Science is truly a collaborative endeavor, and I am honored to contribute to this ongoing conversation.

Contents

1 Introduction

Robotic grasping in unstructured environments remains one of the most challenging problems in robotics. While humans effortlessly grasp and manipulate objects of varying shapes, sizes, and materials, robots struggle to achieve similar dexterity and adaptability. Unstructured environments present a complex array of challenges that distinguish them from controlled laboratory settings: objects appear in arbitrary poses and configurations, scenes contain clutter with partial occlusions, object geometries span from simple geometric shapes to complex household items with irregular surfaces, and lighting conditions vary unpredictably. These environments require robots to perceive object boundaries accurately despite visual noise, reason about 3D geometry from limited viewpoints, and adapt grasp strategies to diverse object properties – all while operating with imperfect sensory information and mechanical constraints. Traditional approaches rely on explicit geometric representations and discrete search over predefined grasp candidates, limiting their ability to generalize beyond training distributions and adapt to the rich variability encountered in real-world scenarios.

Recent advances in NeRF have revolutionized computer vision by demonstrating that continuous, differentiable representations can capture complex 3D geometry and appearance from 2D observations for novel view synthesis – the task of generating photorealistic images of a scene from arbitrary viewpoints given only a limited set of input images. NeRFs achieve this by learning an implicit volumetric representation that maps 3D coordinates and viewing directions to color and density values, which are then integrated via differentiable volume rendering to synthesize images. Crucially, the volumetric rendering loss enforces consistency with the fundamental physics of light transport and camera projection, embedding rich geometric understanding within the learned representations without requiring

explicit 3D supervision. This geometric awareness acquired through novel view synthesis is particularly valuable for robotic grasping tasks, which inherently require spatial reasoning about object shapes, poses, and contact surfaces. The continuous nature of NeRF representations enables smooth interpolation across pose space, while their differentiability allows gradient-based optimization – properties that could enable more effective grasp planning compared to discrete, non-differentiable approaches that struggle to capture the continuous nature of 6-DoF grasp poses and object geometries.

This dissertation explores the fundamental research question:

> *Can an implicit policy based on a neural scene representation trained for novel view synthesis serve as a foundation for robotic grasping?*

The central hypothesis is that geometric understanding learned through volumetric rendering provides valuable inductive biases for grasping tasks – that is, the learned geometric representations encode structural knowledge that helps guide grasp learning even when applied to new objects and scenarios. To test this hypothesis, an optimization-based approach is developed that learns objective functions for continuous grasp planning in $SE(3)$. These learned functions evaluate the success likelihood of candidate grasp poses and enable gradient-based optimization to refine initial grasp candidates into successful grasps. This work makes three main contributions, summarized in the dissertation roadmap shown in Figure 1.1:

NeRF-based Implicit Grasping: This dissertation introduces an image-conditioned multi-view VisionNeRF (mVNeRF) – an adaptation of VisionNeRF (Lin et al. (2023)) as the scene representation, proposes Support Pose Decomposition (SPD) to parameterize and process 6-DoF TCP poses with the NeRF, and designs the *GraspValueReadout* architecture that maps intermediate NeRF activations to a scalar grasp value. A training procedure is developed that embeds this readout

within an implicit policy, enabling gradient-based optimization of continuous poses in $SE(3)$[1] rather than discrete candidate search.

Trajectory-Informed Learning: A new improved training mechanism for the GraspValueReadout leverages full demonstration trajectories, not only terminal grasp poses, to supervise the value function, shaping the optimization landscape and improving convergence and generalization. This addresses a key limitation of traditional implicit policies: the quality of the optimization landscape. By aligning the model's gradients with demonstrated trajectories, the resulting value functions provide better guidance for gradient-based optimization during inference, leading to more reliable convergence to successful grasp poses.

Framework Extensions: To demonstrate compatibility and versatility, federated learning is applied to the mVNeRF representation. In addition, a language-conditioned variant is developed by designing a fusion architecture that combines CLIP image/text features with the NeRF-based implicit grasping pipeline.

Comprehensive evaluation in simulation demonstrates the feasibility of the approach across diverse scenarios, while real-world experiments achieve 71% success rate through zero-shot sim-to-real transfer on unseen objects without domain adaptation or intermediate perception steps, establishing that geometric understanding from novel view synthesis effectively transfers to grasping tasks.

The dissertation is structured as follows: Chapter 2 provides background on Neural Radiance Fields and image-conditioned variants. Chapter 3 introduces the core framework with extending NeRFs and their utilization for implicit grasp policies (mVNeRF, SPD, GraspValueReadout). Chapter 4 enhances the approach through trajectory-informed learning, offering theoretical improvements to learning grasping. Chapter 5 explores framework versatility by integrating federated learning and language conditioning. Chapter 6 surveys related literature and positions this work. Chapter 7 analyzes results and limitations, while Chapter 8 summarizes contributions and broader implications.

[1] $SE(3)$ denotes the Special Euclidean group in three dimensions, representing the space of all rigid transformations (rotations and translations) in 3D space.

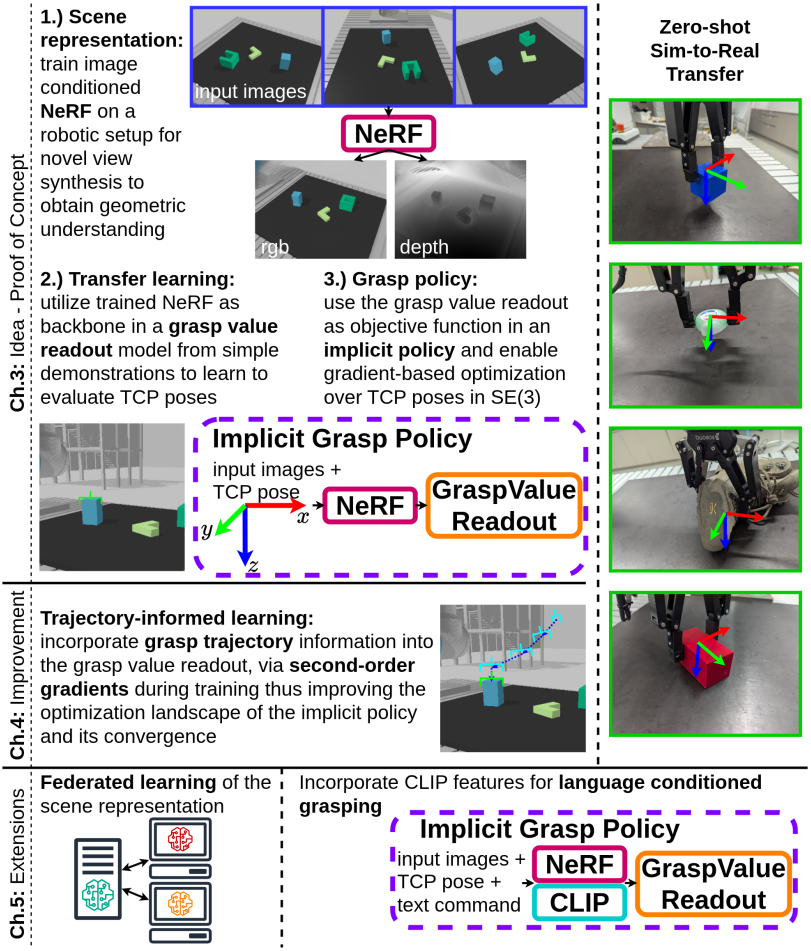


Figure 1.1: Dissertation roadmap. The core idea of this work is that rendering with NeRFs requires accurately learned scene geometry, which can be leveraged for robotic grasping. Chapter 3 presents a proof of concept: an image-conditioned NeRF trained in simulation for novel view synthesis serves as the scene representation. Intermediate NeRF activations are transferred into a grasp-value readout that scores candidate TCP poses, yielding a differentiable objective for an implicit policy that optimizes continuous poses in $SE(3)$. Chapter 4 enhances the method with trajectory-informed learning. Chapter 5 integrates complementary directions, including federated learning and language conditioning with CLIP features. Trained purely in simulation, the framework demonstrates zero-shot sim-to-real transfer without additional adaptation.

In summary, this work establishes neural scene representations as a viable approach for robotic grasping. The NeRF component uses differentiable volume rendering via ray marching together with a calibrated camera model. Known intrinsics and extrinsics ground the representation in the 3D world, so camera effects need not be learned. The implicit-policy formulation operates directly in pose space and, when combined with the NeRF, explicitly exploits the relative pose between cameras and candidate grasps. Complementing this geometric grounding, trajectory-informed learning supervises the value function with full demonstrations such that gradients with respect to the TCP pose correspond to displacements along the trajectory, further anchoring optimization in the physical world and improving effectiveness. These design choices embed real-world physical structure into the model, reduce learning burden, and help bridge the sim-to-real gap, opening pathways for leveraging advances in computer vision in robotics applications.

2 Preliminaries

This chapter lays the groundwork for understanding the technological foundations essential for this dissertation. It introduces the fundamental principles of how NeRFs learn to represent spatial, geometric, and visual information for novel view synthesis.

The chapter is organized as follows: Section 2.1 begins with NeRFs in their original formulation, explaining the core concepts of volumetric rendering and implicit scene representation. Section 2.2 covers image-conditioned variants that enable generalization across scenes, covering pixelNeRF (Section 2.2.1) for multi-view processing and VisionNeRF (Section 2.2.2) for enhanced single-view synthesis using transformer architectures. Section 2.3 introduces iNeRF, demonstrating how trained NeRF models can be inverted for camera pose estimation through gradient-based optimization.

These concepts lead up to the central proposal of this thesis: using NeRF representations for robotic grasping through implicit policies and gradient-based optimization. Section 2.4 connects these components and provides the conceptual foundation for how NeRF representations can be adapted for grasping tasks. The following chapters present the detailed technical implementation, exploring these concepts in depth through implicit grasping policies, trajectory-informed learning, and framework extensions including federated learning and language-conditioned grasping.

> **Note:** This chapter presents work from other researchers and research groups that form the foundation for this dissertation. Each section corresponds to

specific published papers and aims to provide a comprehensive overview of the relevant concepts. However, for complete technical details and derivations, readers are encouraged to consult the original publications. To maintain consistency with the source material, the notations and conventions used in each section follow those of the respective original papers, which may not be consistent across sections or with the notation used in the remainder of this thesis.

2.1 Neural Radiance Field

Neural Radiance Fields, by Mildenhall et al. (2020), merge deep neural networks with classical volume rendering to synthesize novel views of a scene from a set of images. In a NeRF, a scene is represented as a continuous function that maps a 5D input – comprising a 3D location $\mathbf{x} = (x, y, z)$ and a 2D viewing direction (θ, ϕ) where θ is the polar angle and ϕ is the azimuthal angle in spherical coordinates – to an emitted RGB color $\mathbf{c} = (R, G, B)$ and a volume density σ. In practice, the viewing direction is expressed as a 3D Cartesian unit vector d derived from these spherical angles. This function is represented by a neural network F_Θ:

$$F_\Theta(\mathbf{x}, \mathbf{d}) = (\mathbf{c}, \sigma). \tag{2.1}$$

It is designed in a way, that σ only depends on the input 3D position $\mathbf{x}$, while the color $\mathbf{c}$ depends on both position $\mathbf{x}$ and viewing direction $\mathbf{d}$. The overall NeRF rendering pipeline is illustrated in Figure 2.1.

Unlike traditional rendering methods in computer graphics that rely on explicit geometry and material properties, NeRF learns these as an implicit representation directly from images. To render an image from a given camera viewpoint, rays are cast through each pixel using known intrinsic and extrinsic camera parameters. Each ray is parameterized as $\mathbf{r}(t) = \mathbf{o} + t\mathbf{d}$, where $\mathbf{o}$ is the ray origin, $\mathbf{d}$ is the ray direction, and t is the distance along the ray. Points are sampled along each ray and passed to the network, which produces both a color $\mathbf{c}(\mathbf{r}(t), \mathbf{d})$ and a density

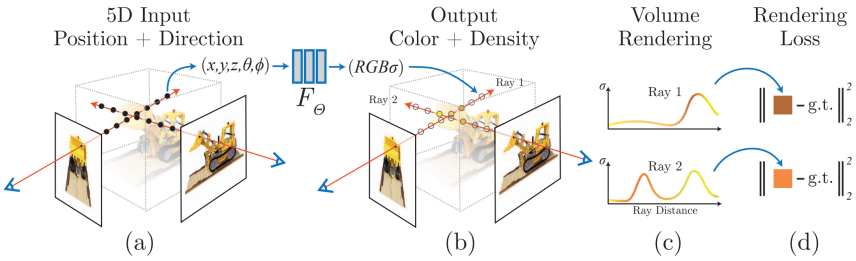


Figure 2.1: NeRF schematic rendering pipeline. Images are rendered by sampling 5D poses (position and viewing direction) along camera rays (a) and using the neural network F_Θ for estimating color and density (b). These can be composed to pixel values via volume rendering (c), and finally used to compute the difference to ground truth pixel values for training the model. Image source: Mildenhall et al. (2020).

$\sigma(\mathbf{r}(t))$ for each sampled point. The expected color $C(\mathbf{r})$ of a ray, and thus a pixel, is then computed using classical volume rendering from near to far bounds, t_n and t_f:

$$C(\mathbf{r}) = \int_{t_n}^{t_f} T(t)\sigma(\mathbf{r}(t))\mathbf{c}(\mathbf{r}(t), \mathbf{d})dt, \quad \text{with} \quad T(t) = \exp(-\int_{t_n}^{t} \sigma(\mathbf{r}(s))ds). \tag{2.2}$$

The near and far bounds, t_n and t_f, specify the distances from the camera origin to the closest and farthest scene points that contribute to the rendered image and must be chosen to match the spatial extent of the target environment. For the robotic workspace discussed in later chapters, the workspace is a $0.5\text{m} \times 0.5\text{m} \times 0.3\text{m}$ volume with the cameras arranged to lie on a sphere with a radius of 0.8m around its center. The values for the near and far bounds are $t_n = 0.3\text{m}$ and $t_f = 1.3\text{m}$.

In practice, the integral is estimated numerically using stratified sampling. The interval $[t_n, t_f]$ is partitioned into N bins, and one sample is drawn uniformly from each bin resulting in a set of samples $t_1, t_2, ..., t_N$. The color along a ray is then approximated by

$$\hat{C}(\mathbf{r}) = \sum_{i=1}^{N} T_i(1 - \exp(-\sigma_i\delta_i))\mathbf{c}_i, \quad \text{with} \quad T_i = \exp(-\sum_{j=1}^{i-1} \sigma_j\delta_j), \tag{2.3}$$

where $\delta_i = t_{i+1} - t_i$ is the distance between adjacent samples and σ_i and $\mathbf{c}_i$ are the density and color corresponding to the ith sample. As for the edge cases, $T_1 = 1$ and $\delta_N = \infty$. A rough interpretation of this equation is that the density σ_i is a measure of how likely the ray terminates at sample i. Terminating means that the ray intersects a surface which is largely responsible for the ray's color. The renderer then blends the sampled colors $\mathbf{c}_i$, assigning greater weight to points with higher density, to produce the final pixel color.

Similar to color rendering, depth maps can also be generated by applying the same weighting to the distances of the sampled points to the camera:

$$\hat{D}(\mathbf{r}) = \sum_{i=1}^{N} T_i(1 - \exp(-\sigma_i\delta_i))\mathbf{t}_i. \tag{2.4}$$

To improve sampling efficiency, NeRF employs a hierarchical sampling strategy that focuses computational resources on the most important regions along each ray. This approach uses two networks with identical architectures – a coarse network and a fine network – in a two-stage process:

Stage 1 (Coarse sampling): Points are sampled uniformly along the ray using stratified sampling, and the coarse network predicts densities at these locations. This provides a rough estimate of where objects are likely located along the ray.

Stage 2 (Importance sampling): The coarse density predictions are used to construct a piecewise linear probability distribution that concentrates sampling in regions with higher predicted density (i.e., where objects are more likely). Additional points are sampled from this distribution, and the fine network processes both the original stratified samples and these new importance-weighted samples to compute the final pixel color.

During training, both the colors computed from the coarse network and the colors computed from the fine network are compared to the target pixel color $\mathbf{c}_{gt}$ to form the volume rendering loss:

$$L = ||\hat{C}(\mathbf{r})_{coarse} - \mathbf{c}_{gt}||_2 + ||\hat{C}(\mathbf{r})_{fine} - \mathbf{c}_{gt}||_2. \tag{2.5}$$

Training both networks simultaneously ensures that the coarse network learns to provide useful guidance for the fine network's sampling, while both networks are optimized to produce accurate color predictions for their respective sample sets. During inference, volume rendering results of the fine network are used to synthesize images for novel views.

Standard neural networks are most effective at learning low-frequency functions. However, the volumetric representation that NeRFs learn need to model abrupt changes, especially at objects' boundaries. This is mitigated by employing a positional encoding $\gamma(\cdot)$, that lifts the input coordinates into a higher dimensional space:

$$\gamma(p) = (\sin(2^0\pi p), cos(2^0\pi p), \ldots, sin(2^{L-1}\pi p), cos(2^{L-1}\pi p)), \tag{2.6}$$

with L the number of frequency bases. This transformation is applied independently to each component of the 3D point and the unit vector representing the 2D viewing direction. It allows the network to model high-frequency variations in geometry and appearance, enabling photorealistic view synthesis.

Raycasting introduces an inductive bias and geometric consistency that reflects how cameras perceive scenes, simplifying learning by eliminating the need to train explicitly for features inherently understood through volume rendering. By embedding scene geometry into a continuous volumetric representation, NeRFs enable the learning of physical structures, which is particularly relevant to robotic grasping and potentially other robotics tasks. However, in their original formulation, NeRFs only learn to represent static scenes, making them impractical for robotics, where interaction with and modification of the environment are

fundamental. Retraining a NeRF whenever the environment changes requires significant computational resources and time, limiting its applicability for robotics in changing and unstructured environments.

There are two main ways to address this limitation. The first focuses on accelerating NeRF training, with works such as InstantNGP (Müller et al. (2022)) and PlenOctrees (Yu et al. (2021a)) achieving very fast optimization, reducing training and rendering times significantly. The second approach aims to generalize NeRFs by introducing mechanisms such as image conditioning, as seen in models like pixelNeRF (Yu et al. (2021b)) and VisionNeRF (Lin et al. (2023)). In this work, the latter approach is adopted, as the goal is to use the latent scene representation learned by NeRFs. This requires a model that produces consistent latent features across different views and scenes, something that would be infeasible if the representation had to be reconstructed from scratch each time.

2.2 Image Conditioned NeRFs

While standard NeRFs provide an effective way to synthesize novel view, they require per-scene optimization and many input views, making them impractical for many real-world applications. This limitation has led to the development of image-conditioned NeRFs, which generalize across scenes by leveraging prior knowledge learned from a dataset of multi-view images. Rather than training a network to reconstruct a single scene (a specific environment or object arrangement), these models are trained on multiple scenes, allowing them to infer representations from a sparse set of input images in a feed-forward manner without requiring additional training or optimization.

To achieve this, image-conditioned NeRFs extract meaningful scene features from input images, typically using Convolutional Neural Networks (CNNs) or Vision Transformers (ViTs). These extracted features serve as additional inputs to the radiance field, conditioning the reconstruction process on the visual information provided by the given images. By learning a shared prior over many scenes,

Table 2.1: Summary of key differences and similarities between NeRF, PixelNeRF, and VisionNeRF.

Property	**NeRF (original)**	**PixelNeRF**	**VisionNeRF**
Image conditioned	no	yes	
Rendering quality	very high	moderate	moderate to high
Source image features	N/A	CNN	CNN and ViT
Image feature conditioning	N/A	residual addition at multiple layers	concatenation at input only
Number of source images	many	few (one or several)	one
Positional encoding	applied to position and direction	applied to position only	
Direction conditioning	input at second to last layer	concatenated to encoded position and used at first layer	

image-conditioned NeRFs become more flexible compared to traditional NeRFs. Conditioning on image features enables better generalization to unseen scenes, making these models suitable for environments where objects or lighting conditions may change.

In the following, two approaches to image-conditioned NeRFs are discussed: pixelNeRF and VisionNeRF. These models employ different strategies to extract and incorporate source image information into the radiance field for novel view synthesis from limited observations. To provide a clear overview before exploring their details, the main similarities and differences between the original NeRF, pixelNeRF and VisionNeRF are summarized in Table 2.1. Note that while rendering quality serves as a useful proxy for representation quality, rendering speed is not a primary concern since this work uses NeRFs as scene representations rather than for actual rendering applications.

2.2.1 PixelNeRF

PixelNeRF (Yu et al. (2021b)) uses a fully convolutional encoder to extract pixel-aligned features from input source images. The extracted feature grid is denoted as:

$$\mathbf{W} = E(I), \tag{2.7}$$

where $E(\cdot)$ is the convolutional encoder applied to the input image I. These features remain aligned with the original image pixels, ensuring that spatial information is preserved. The encoder extracts a feature pyramid using a Fully Convolutional Network (FCN) ResNet34 backbone (He et al. (2016)) – consisting of 1 convolutional layer and 16 convolutional ResNet blocks – pre-trained on ImageNet (Russakovsky et al. (2015)). These pyramid features are upsampled and concatenated to form pixel-aligned latent feature maps.

PixelNeRF employs the same volume rendering strategy as NeRF, including hierarchical sampling. However, instead of predicting density and color solely from positional and directional inputs, pixelNeRF also conditions the predictions on the extracted spatial features. For each 3D point sampled along a ray, the corresponding spatial features are obtained by projecting the point onto the source image and retrieving the pixel-aligned features at that location. The pipeline for the single-view case is shown schematically in Figure 2.2.

For multi-view processing, pixelNeRF employs five ResNet blocks divided into two main components: f_1 and f_2. The first three blocks, denoted by f_1, process the source views independently. Their outputs are averaged to fuse the intermediate latent features, which are then processed by the last two blocks, f_2, to compute the density and color values. The multi-view architecture is illustrated in Figure 2.3.

For per-view processing in f_1, the position and direction vectors are transformed into the coordinate frame of the corresponding source camera. Unlike vanilla NeRF, which operates in a fixed canonical coordinate frame, pixelNeRF must answer the question: "What does this 3D point look like when observed from this viewing direction, given these input images?" To achieve view-independent

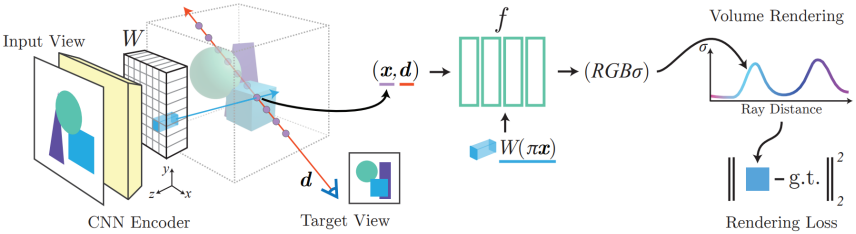


Figure 2.2: PixelNeRF schematic rendering pipeline with a single source image. Like in NeRFs images are rendered by sampling 5D poses along camera rays (in the source camera's coordinate system) and projecting the 3D positions onto the pixel-aligned CNN features to obtain the corresponding spatial features. The 5D poses and the spatial features are processed by the neural network f to estimate color and density. The obtained color and density values are combined via volume rendering to obtain the estimated pixel values and can then be compared to the ground truth pixel. Image source: Yu et al. (2021b).

representations, each query is processed from the perspective of each input image. This requires transforming the query poses into each source camera's coordinate frame as a preprocessing step before feeding them into f_1.

Given the ith source image $I^{(i)}$ and its associated camera transformation $[R^{(i)}, \mathbf{t}^{(i)}]$, the sampled points $\mathbf{x}$ and viewing directions $\mathbf{d}$ along a ray are transformed as follows:

$$\mathbf{x}^{(i)} = R^{(i)}\mathbf{x} + \mathbf{t}^{(i)}, \quad \mathbf{d}^{(i)} = R^{(i)}\mathbf{d} \tag{2.8}$$

where $\mathbf{x}^{(i)}$ and $\mathbf{d}^{(i)}$ are the transformed position and direction in the ith camera's coordinate frame.

The corresponding spatial features are obtained by projecting $\mathbf{x}^{(i)}$ onto the image plane using the camera's intrinsics, yielding projected image coordinates $\pi(\mathbf{x}^{(i)})$. Since the projection typically produces non-integer pixel coordinates, bilinear interpolation is applied between the encoded image features $\mathbf{W}^{(i)} = E(I^{(i)})$ at the four neighboring pixel locations to obtain the corresponding spatial features $\mathbf{W}^{(i)}(\pi(\mathbf{x}^{(i)}))$. These features are then passed to f_1 along with the transformed position and viewing direction to obtain intermediate latent features $\mathbf{V}^{(i)}$:

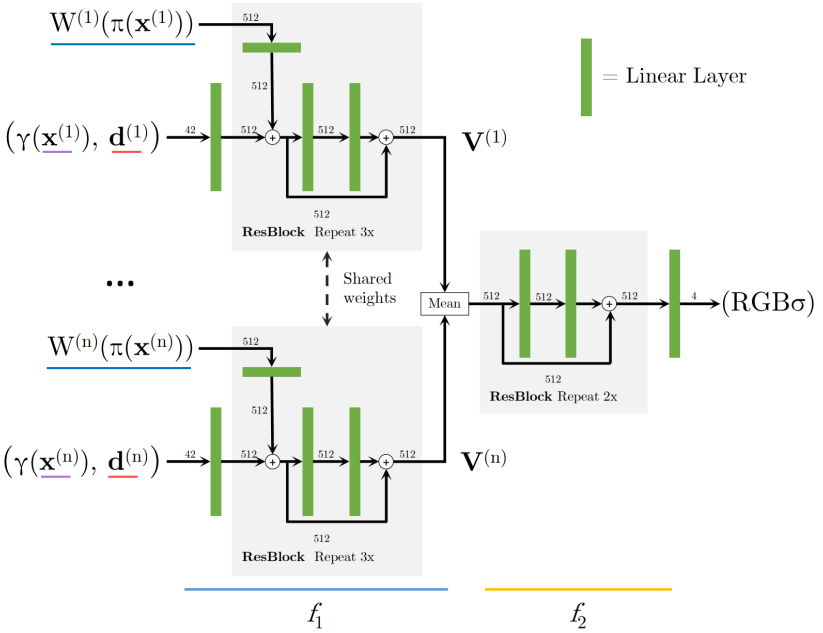


Figure 2.3: PixelNeRF multi-view architecture. Both position and direction vectors are used as input to the initial layer of f_1 in the source images' coordinate system, while only the 3D position is positionally encoded with γ. The spatial features extracted from the source images are used in each ResNet block of f_1, and the obtained per-image latent features are aggregated before being processed by f_2. Image source: Yu et al. (2021b).

$$\mathbf{V}^{(i)} = f_1(\gamma(\mathbf{x}^{(i)}), \mathbf{d}^{(i)}, \mathbf{W}^{(i)}(\pi(\mathbf{x}^{(i)}))) \tag{2.9}$$

Transforming the sampled poses into the coordinate frame of each source camera before they are processed by f_1 makes the representation independent of the cameras' absolute positions and orientations. In this local frame the network can learn a view-consistent mapping, enabling it to generalize across arbitrary camera setups. Intuitively, the transformation lets the network answer the question, "Given what is visible in this source image, how would this 3D point appear if it was observed from that (target camera) direction?" Aligning the poses with the source

image therefore provides a consistent reference that ties the pixel-aligned features to the sampled 3D locations.

Unlike the original NeRF, the viewing direction is already input to f_1 at the first layer and does not undergo positional encoding. Before the first ResNet block, the encoded position and raw direction vector are passed through a fully connected layer. Furthermore, at the beginning of each ResNet block, the projected spatial features are added to the block's input, reinforcing the conditioning on the extracted image features, as illustrated in Figure 2.3.

The per-image latent features $\mathbf{V}^{(i)}$ for $i = 1, \ldots, n$ are aggregated using an average pooling operator (Mean) and processed by f_2 to obtain the final outputs:

$$f_2(\text{Mean}(\mathbf{V}^{(1)}, \ldots, \mathbf{V}^{(n)})) = (\mathbf{c}, \sigma) \tag{2.10}$$

where $\mathbf{c}$ is the emitted color and σ is the predicted volume density.

These values are then used in the volume rendering pipeline to compute the pixel color for a given ray, enabling both model training and novel view synthesis as described in Equation 2.3.

Overall, pixelNeRF offers several advantages over standard NeRFs. It eliminates the need for per-scene optimization, enabling generalization to unseen scenes and object categories. By operating in view space rather than a canonical coordinate frame, it allows for flexible scene representations. Additionally, its fully convolutional architecture ensures spatial consistency between input images and the reconstructed 3D representation. However, this generalization comes at a cost: the rendering quality of pixelNeRF is lower than that of standard NeRFss, as it prioritizes flexibility over scene-specific optimization. Moreover, it requires the relative transformations between source images and the target view to be known, as these are essential for projecting and aligning features correctly across different viewpoints.

2.2.2 VisionNeRF

VisionNeRF (Lin et al. (2023)) extends the single-view image conditioned novel view synthesis capabilities of NeRF-based models by combining both convolutional and transformer-based feature extraction. Similar to pixelNeRF, it provides pixel-aligned features for volume rendering, however, VisionNeRF also leverages the global context learned by a ViT.

As a first step, VisionNeRF employs a significantly smaller fully convolutional network than pixelNeRF also without pre-trained weights. It is comprised of 1 convolutional layer and 3 – instead of 16 – convolutional ResNet blocks to extract a local feature map W_L from the input image:

$$\mathbf{W}_L = \mathcal{G}_L(I), \tag{2.11}$$

with convolutional encoder $\mathcal{G}_L$ and input image I.

To complement these local features, VisionNeRF further employs a ViT-based encoder (Dosovitskiy et al. (2020)), pretrained on ImageNet. In this implementation, the input image is divided into 16×16 non-overlapping patches (for a 224×224 image), which are flattened to form $14 \times 14 = 196$ patch embeddings. Typical for ViT architectures, a class token is prepended to the sequence of patch embeddings, and a learnable positional encoding is added to each token. This sequence is then passed through 12 transformer blocks, each composed of multi-head self-attention and feed-forward layers. Unlike classification tasks, VisionNeRF interprets the class token as a background token, encoding scene context not directly visible in the source image.

After every 3rd transformer block, the intermediate latent features are reshaped back into 14×14 2D latent embeddings and are further processed using convolutional or deconvolutional layers (Zeiler et al. (2010)), producing feature maps of resolutions 7×7, 14×14, 28×28, and 56×56. These 2D feature maps are upsampled to match the resolution of $\mathbf{W}_L$, for obtaining the multi-level ViT-embeddings $\mathbf{W}_G^1$, $\mathbf{W}_G^2$, $\mathbf{W}_G^3$ and $\mathbf{W}_G^4$. Concatenating the CNN feature map and

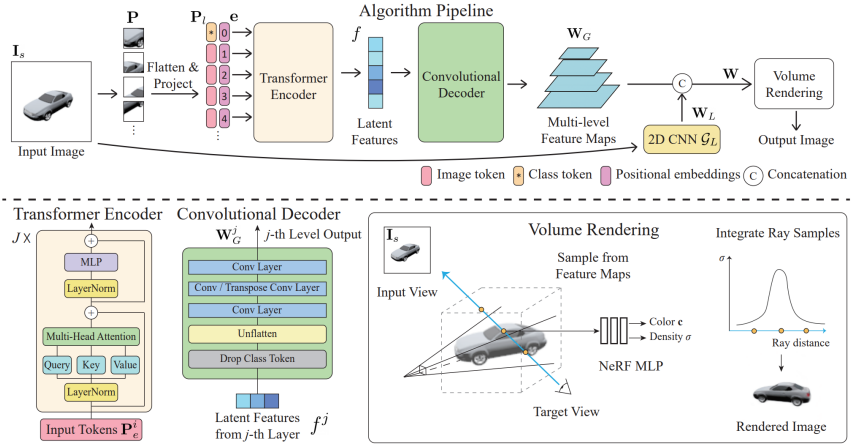


Figure 2.4: VisionNeRF rendering pipeline and architecture. The transformer encoder extracts latent features from non-overlapping image patches of the source image and the convolutional decoder produces the multi-level global feature maps. Additionally, an FCN extracts local features from the source images. These are concatenated and passed to the volume rendering pipeline. Image source: Lin et al. (2023).

these multi-level ViT-embeddings in their feature dimension results in a combined final feature grid:

$$\mathbf{W} = \mathbf{W}_L \oplus \mathbf{W}_G^1 \oplus \mathbf{W}_G^2 \oplus \mathbf{W}_G^3 \oplus \mathbf{W}_G^4, \tag{2.12}$$

with $\oplus$ the concatenation operator. A schematic VisionNeRF model architecture is shown in Figure 2.4.

Following the standard rendering pipeline introduced in Section 2.1, VisionNeRF casts rays from the target camera's viewpoint and samples points along each ray. Like in pixelNeRF (see Section 2.2.1), each 3D point $\mathbf{x}$ is projected into the source image space (using known camera intrinsics and extrinsics) to obtain its corresponding 2D location $\pi(\mathbf{x})$. Bilinear interpolation on $\mathbf{W}$ yields spatial features for that point $\mathbf{W}(\pi(\mathbf{x}))$. Along to the spatial features, VisionNeRF also

uses $\gamma(\mathbf{x})$, the positional encoding of $\mathbf{x}$, and the (unencoded) viewing direction $\mathbf{d}$ as input into a fully connected network f with six ResNet blocks:

$$f(\gamma(\mathbf{x}), \mathbf{d}, \mathbf{W}(\pi(\mathbf{x}))) = (\mathbf{c}, \sigma). \tag{2.13}$$

These color $\mathbf{c}$ and density σ estimates are integrated via the volumetric rendering Equation 2.3, yielding an output pixel color. Since VisionNeRF only processes a single source image, there is no need for dividing f into two parts, since there is no need for fusing the latent features obtained from multiple source images. An other significant architectural difference to pixelNeRF is, that the instead of adding the projected spatial features to the encoded position and unencoded direction vector and using them at the beginning of each ResNet, these vectors are concatenated and are only used as input to the initial layer of f.

By integrating both a CNN and a ViT, VisionNeRF learns to infer robust 3D representations from a single input image. The CNN supplies spatially-aligned local features, while the transformer-based encoder captures global scene cues. Combined with the volumetric rendering framework of NeRF, this approach achieves improved single-view novel view synthesis.

2.3 iNeRF: Inverting Neural Radiance Fields for Pose Estimation

While NeRFs and their image-conditioned variants have primarily been explored for synthesizing novel views, iNeRF (Yen-Chen et al. (2021)) applies their capabilities to the inverse problem of camera pose estimation. Given an image and a pre-trained NeRF representation of the scene, iNeRF utilizes an iterative, gradient-based optimization that minimizes the difference between the pixels observed in the input image and pixels rendered from the pre-trained NeRF model. The objective of this optimization is formulated as:

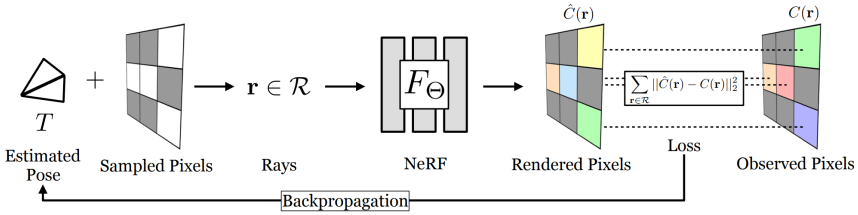


Figure 2.5: Camera pose estimation with iNeRF. An iterative gradient-based optimization refines the initial pose estimate by minimizing the pixel-wise difference between the rendered and input images. Image source: Yen-Chen et al. (2021).

$$\hat{T} = \underset{T \in SE(3)}{\operatorname{argmin}} \sum_{r \in R[T]} ||\hat{C}(r) - C(r)||_2^2, \tag{2.14}$$

analogous to the volume rendering loss from Equation 2.5. Here $\hat{T}$ is the estimated camera pose, $R[T]$ denotes the set of rays used for volume rendering at camera pose T, $\hat{C}(r)$ is the color rendered by the NeRF's fine network, and $C(r)$ is the same pixel of the input image. The computation, including neural network inference, is differentiable. During optimization, this loss is backpropageted to update the pose, rather than the weights of the network, aligning the rendered image with the given input image. The process is illustrated in Figure 2.5.

In practice, rendering a full image at every optimization step is computationally expensive. Thus, iNeRF employs efficient ray sampling strategies, specifically focusing on rays that provide the most informative gradients.

During optimization, the relative transformation from an initial pose T_0 is adjusted. To ensure that poses remain valid on the $SE(3)$ manifold throughout the optimization, the relative transformations are parametrized using the Lie algebra $\mathfrak{se}(3)$ representation:

$$\hat{\mathbf{v}} = \underset{\mathbf{v} \in \mathfrak{se}(3) \cong \mathbb{R}^6}{\operatorname{argmin}} \sum_{r \in R[\exp(\mathbf{v})T_0]} \|\hat{C}(r) - C(r)\|_2^2, \tag{2.15}$$

where $\mathbf{v} = (\mathbf{t}, \boldsymbol{\omega})$ is the relative transformation, with 3D translation vector $\mathbf{t}$ and rotation vector $\boldsymbol{\omega}$. The exponential map $e(\cdot)$ maps elements from the Lie algebra $\mathfrak{se}(3)$ onto the Lie group $SE(3)$, naturally constraining the optimization to valid rigid-body transformations.

Furthermore, iNeRF extends beyond pose estimation for individual scenes. It can also perform category-level object pose estimation when paired with a pre-trained pixelNeRF. This enables pose estimation even for previously unseen object instances, facilitating broader applicability without requiring explicit 3D geometry or mesh models.

2.4 NeRF-based Grasping: Connecting the Components

The techniques presented in this chapter – NeRFs, image-conditioned variants, and iNeRF – form the foundation for adapting neural scene representations to robotic grasping. At its core, a NeRF learns a volumetric scene function that maps 5-Degrees of Freedom (DoF) poses to scene properties. The central insight of this work is to apply the same representational framework but learn a different output: instead of predicting color and density for novel view synthesis, the network learns to predict grasp values that can be maximized to identify successful grasps.

This shift transforms the question from "what color would be observed from this viewpoint?" to "how likely would a grasp succeed at this pose?" Crucially, the geometric understanding acquired through training for novel view synthesis provides the spatial reasoning foundation necessary for grasping tasks. This enables a transfer learning approach where NeRF representations trained for view synthesis are leveraged for grasping without learning geometry from scratch.

For robotic applications, image-conditioned NeRFs are particularly well-suited as they can generalize across scenes without requiring retraining for each new configuration, ensuring consistent representations essential for transfer learning.

This work combines the multi-view capabilities of pixelNeRF with the enhanced feature extraction of VisionNeRF to process multiple camera observations.

Two key technical challenges arise in this adaptation: First, extending the architecture to process 6-DoF poses rather than 5-DoF inputs. Second, adapting the pre-trained network to output grasp values while preserving the learned geometric representations.

The resulting grasping pipeline draws inspiration from iNeRF's pose optimization approach. Where iNeRF optimizes camera poses by minimizing rendering error, the proposed method optimizes grasp poses by maximizing learned grasp values through gradient-based optimization in the differentiable pipeline.

This framework leverages the geometric priors learned through volumetric rendering while adapting the representational power of NeRFs to grasping tasks. The following chapters detail the technical implementation of this approach, demonstrating how neural scene representations trained for novel view synthesis enable robotic grasping through transfer learning.

3 NeRF as Scene Representation for Implicit Grasp Policies

Building on the foundational concepts introduced in the previous chapter, this chapter presents the technical implementation of NeRF-based implicit grasp policies. The core contribution is a method that extends image-conditioned NeRFs to process 6-DoF grasp poses and learns grasp value functions through transfer learning, enabling gradient-based optimization for robotic grasping across diverse scenarios, including zero-shot sim-to-real transfer.

The chapter begins by formulating the grasping problem in Section 3.1. Section 3.2 introduces the proposed grasp policy, including an extended VisionNeRF architecture and the utilization of its learned latent space to inform grasping actions. Experimental validation and results are presented in Section 3.3, and Section 3.4 summarizes the chapter.

The proposed method addresses grasping with a two-finger parallel jaw gripper, however, the underlying principles of NeRF-based scene representation and gradient-based pose optimization can be readily extended to other robotic platforms, gripper configurations, and manipulation primitives.

> **Note:** This chapter builds upon previous work (Sóti et al. (2024)), presenting the latest implementation of the grasping framework along with some modifications. These modifications are summarized in Appendix A.1.

3.1 Problem statement

Consider a robot equipped with a gripper operating in a workspace observed by calibrated cameras, tasked with grasping objects from the environment. The objective is to derive a grasp policy capable of predicting poses of the Tool Center Point (TCP) that result in successful grasps.

Definition 3.1.1 (Policy)**.** *A policy π is a function that maps an observation o from the observation space $\mathcal{O}$ to an action a in the action space $\mathcal{A}$:*

$$\pi : \mathcal{O} \rightarrow \mathcal{A}, \pi(o) = a.$$

To formulate a grasp policy, first the observation space $\mathcal{O}$ and the action space $\mathcal{A}$ are defined.

Given N cameras observing the workspace simultaneously, each camera provides a measurement consisting of an RGB image I, camera extrinsics $(R, \mathbf{t})$, and camera intrinsic matrix K. The extrinsics consist of a rotation matrix R and translation vector $\mathbf{t}$ that define the camera's pose relative to the world coordinate frame. The intrinsic matrix is solely used for constructing the projection function ρ to project the 3D positions onto the image plane (the function is defined further below in Equation 3.9). Thus, the intrinsic matrix is regarded as equivalent to the projection function in this work. A camera measurement is defined as the following tuple:

$$c = (I, R, \mathbf{t}, \rho) \in \mathcal{C}. \tag{3.1}$$

Let $\mathcal{C}$ denote the space of all camera measurements. The observation space $\mathcal{O}$ is defined as:

$$\mathcal{O} = \mathcal{P}(\mathcal{C}), \tag{3.2}$$

where $\mathcal{P}$ denotes the power set[1]. Consequently, an observation $o \in \mathcal{O}$ is a set of camera measurements:

$$o = \{c_1, c_2, \ldots, c_N\} \in \mathcal{O}. \tag{3.3}$$

The action space $\mathcal{A}$ is defined as the 6-DoF pose ($SE(3)$) space of the TCP.

With these definitions in place, the following section introduces the proposed grasp policy, leveraging an image-conditioned NeRF to predict suitable grasp poses from observations.

3.2 Method

NeRFs implicitly represent geometry by learning a mapping from a continuous 5-DoF input space to color and volume density. To leverage this in robotic grasping tasks, this input space is aligned with the robot's workspace and a trained NeRF is extended to a grasp value model. This grasp value model serves as the value function for an implicit grasp policy, allowing grasp poses to be determined by solving an optimization problem that maximizes the grasp value.

Definition 3.2.1 (Implicit policy)**.** *An* implicit policy π *is a policy that determines actions through the optimization of an intermediate scalar-valued function. Given an observation $o \in \mathcal{O}$ and an action space $\mathcal{A}$, an implicit policy can be defined as:*

$$\pi : \mathcal{O} \to \mathcal{A}, \pi(o) = \underset{a \in \mathcal{A}}{\operatorname{argmax}}\, Q(o, a). \tag{3.4}$$

where $Q : \mathcal{O} \times \mathcal{A} \to \mathbb{R}$ is a value function encoding the quality of action a given observation o.

[1] The power set $\mathcal{P}(\mathcal{C})$ is the set of all possible subsets of $\mathcal{C}$, including the empty set and $\mathcal{C}$ itself.

Note: Equivalently, this formulation can be interpreted as minimizing an energy or cost function $E(o, a)$:

$$\pi : \mathcal{O} \rightarrow \mathcal{A}, \pi(o) = \underset{a \in \mathcal{A}}{\operatorname{argmin}} E(o, a).$$

See Implicit Behavioral Cloning by Florence et al. (2022) for more details.

A standard NeRF, trained for novel view synthesis, inherently encodes scene geometry within its latent embeddings. However, NeRFs in their original formulation represent static scenes, limiting their direct applicability to robotics. In changing environments, interactions require retraining the NeRF for each new scene configuration – an impractical process that incurs computational overhead and inconsistencies in the learned representations.

To address this limitation, this work employs an image-conditioned NeRF, that generates consistent scene representations in a single forward pass from new visual observations. As discussed in the preliminaries chapter, two such image-conditioned NeRF variants are pixelNeRF (Section 2.2.1) and VisionNeRF (Section 2.2.2). PixelNeRF processes multiple views using CNN-based visual embeddings, while VisionNeRF enhances single-image processing through a ViT, enabling richer feature extraction. The proposed approach, *multi-view VisionNeRF (mVNeRF)*, is described in Section 3.2.1. It combines the multi-view capabilities of pixelNeRF with the enhanced feature extraction of VisionNeRF.

The grasping framework leverages this learned representation and extends it in two ways:

1. *Support Pose Decomposition:* The input representation is extended from the original 5-DoF poses to full 6-DoF grasp poses (Section 3.2.2).

2. *Grasp Value Model:* The NeRF's latent embeddings are used to map the decomposed grasp pose to a single scalar grasp value rather than to color and density values (Section 3.2.3).

The grasp value model forms the basis for the implicit grasp policy described in Section 3.2.4. The policy employs gradient-based optimization starting from randomly sampled grasp candidates to identify high-value grasp poses.

3.2.1 Multi-view VisionNeRF

While VisionNeRF (Lin et al. (2023), Section 2.2.2) excels at single-view novel view synthesis, grasping benefits from resolving 3D geometric ambiguities inherent to the single-view setting. When available, multiple calibrated views provide complementary constraints on the scene geometry, improving geometric consistency and occlusion handling. Motivated by the results of pixelNeRF (Yu et al. (2021b), Section 2.2.1) on real imagery with multiple input views, VisionNeRF is extended to a multi-view formulation, mVNeRF, that accepts an arbitrary number of input views.

A NeRF is a learned volumetric scene function that is queried along camera rays to compute pixel values by volume rendering. The volume rendering equation 2.3 governing how color contributions accumulate along a ray $\mathbf{r}$ is:

$$\hat{C}(\mathbf{r}) = \sum_{i=1}^{N} T_i(1 - \exp(-\sigma_i \delta_i))\mathbf{c}_i, \quad \text{with} \quad T_i = \exp(-\sum_{j=1}^{i-1} \sigma_j \delta_j), \tag{3.5}$$

where N is the number of sampled points along the ray, σ_i and $\mathbf{c}_i$ are the density and color at the ith sample and δ_i denotes the distance between adjacent samples.

The volumetric scene function f maps 5-DoF inputs (3D position and viewing 2D direction) to color $\mathbf{c}$ and density σ. Image-conditioned NeRFs extend this by incorporating scene information derived from input images. The rendering pipeline, illustrated in Figure 3.1, consists of two learnable components: visual embedding $\mathcal{G}$ for spatial feature extraction, and the mVNeRF network as the volumetric scene function. In the 1-view case, an image-conditioned NeRF

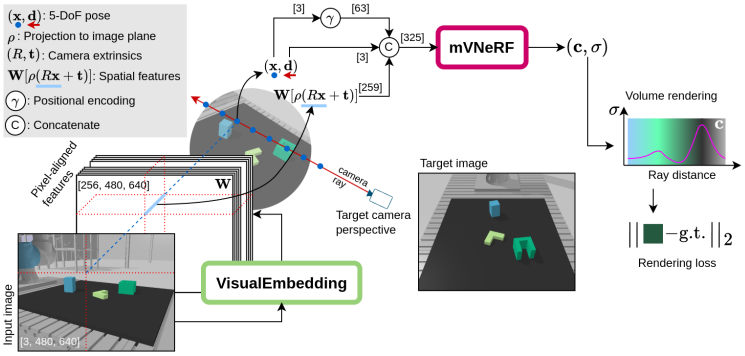


Figure 3.1: Volume rendering pipeline for mVNeRF. The process involves sampling points along camera rays, projecting these points onto source images to retrieve spatial features, and using these features along with position and direction information to predict color and density values for volume rendering.

estimates $\mathbf{c}$ and σ from a query pose $(\mathbf{x}, \mathbf{d})$ and scene information derived from the input image I:

$$f(\mathbf{x}, \mathbf{d}, \mathbf{w}) = (\mathbf{c}, \sigma), \tag{3.6}$$

where $\mathbf{w}$ is the spatial feature vector derived from the input image, as detailed below.

The visual embedding $\mathcal{G}$ (see Figure 3.2) follows VisionNeRF's approach, utilizing a convolutional encoder for local features (see architecture in Appendix B.1) and a pre-trained ViT for extracting global multi-level embeddings (see architecture in Appendix B.2). The combined visual embedding $\mathbf{W}$ concatenates and scales these features, providing pixel-aligned features:

$$\mathcal{G}(I) = \mathbf{W}. \tag{3.7}$$

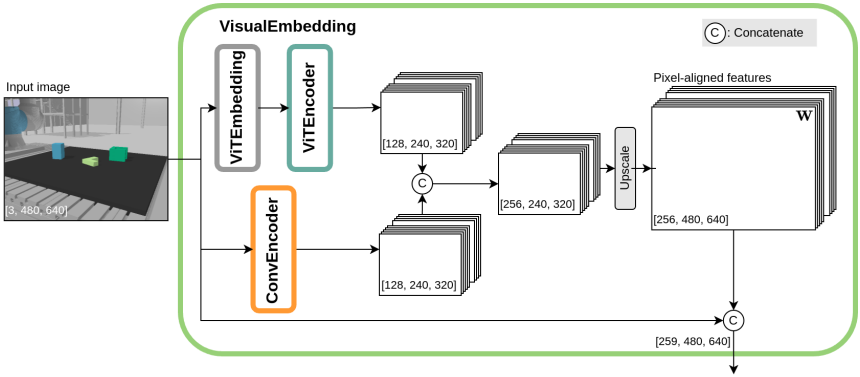


Figure 3.2: Combined visual embedding. The final feature grid concatenates local CNN features with multi-level ViT features, providing a comprehensive scene representation. For aligning the features with the input image pixels, the feature grid is scaled to the inputs resolution.

To compute spatial features for a 5-DoF pose $(\mathbf{x}, \mathbf{d})$, the camera extrinsics $(R, \mathbf{t})$ and intrinsics (K) are required. First, the 3D position $\mathbf{x}$ is transformed into the camera coordinate frame:

$$\mathbf{x}' = R\mathbf{x} + \mathbf{t}, \tag{3.8}$$

The projection onto the image plane is defined by:

$$\rho : \mathbb{R}^3 \to \mathbb{R}^2, \rho(\mathbf{x}') = \begin{pmatrix} f_x \frac{x'}{z'} + c_x \\ f_y \frac{y'}{z'} + c_y \end{pmatrix}, \tag{3.9}$$

with $\mathbf{x}' = (x', y', z')$ 3D position and $K = \begin{pmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{pmatrix}$ the camera intrinsics, where f_x and f_y are the focal lengths and c_x and c_y are the principal point coordinates (not to be confused with the neural network modules f_1, f_2 or the color $\mathbf{c}$).

Note: ρ is equivalent to the projection function π from pixelNeRF and VisionNeRF in section 2.2.1 and 2.2.2. the notation ρ is used to avoid confusion with the policy π from the implicit policy definition in section 3.2.4.

The spatial features are obtained by accessing the pixel-aligned feature map $\mathbf{W}$ at the projected pixel location $\rho(\mathbf{x}')$ using bilinear interpolation:

$$\mathbf{w} = \mathbf{W}[\rho(\mathbf{x}')], \tag{3.10}$$

where $[\cdot]$ denotes the bilinear interpolation operator that computes values for non-integer pixel coordinates.

MVNeRF follows pixelNeRF's multi-view strategy, employing a three-stage pipeline. Each input image is first processed independently to extract visual embeddings. Query poses and spatial features $\mathbf{w}$ are then independently processed per-view in f_1. Per-view embeddings are subsequently aggregated before final density and color predictions are produced by f_2. Figure 3.3 illustrates the architecture of mVNeRF. VisionNeRF's six ResNet blocks are split: f_1 comprises the first three blocks, and f_2 consists of the remaining three blocks processing aggregated features.

Formally, given a set of camera measurements $o = \{(I^{(i)}, R^{(i)}, \mathbf{t}^{(i)}, \rho^{(i)})\}_{i=1}^{N}$, a 5-DoF query pose $(\mathbf{x}, \mathbf{d})$ is transformed into each camera's coordinate frame:

$$\mathbf{x}^{(i)} = R^{(i)}\mathbf{x} + \mathbf{t}^{(i)}, \quad \mathbf{d}^{(i)} = R^{(i)}\mathbf{d} \tag{3.11}$$

The corresponding spatial features are obtained analogously to the 1-view case:

$$\mathbf{w}^{(i)} = \mathbf{W}^{(i)}[\rho^{(i)}(\mathbf{x}^{(i)})] \tag{3.12}$$

with $\mathbf{W}^{(i)} = \mathcal{G}(I^{(i)})$ the visual embedding of the ith camera image and $\rho^{(i)}$ the projection function of the ith camera using its intrinsics. In practice, both in simulation and the real world, the camera intrinsics are identical for all observations,

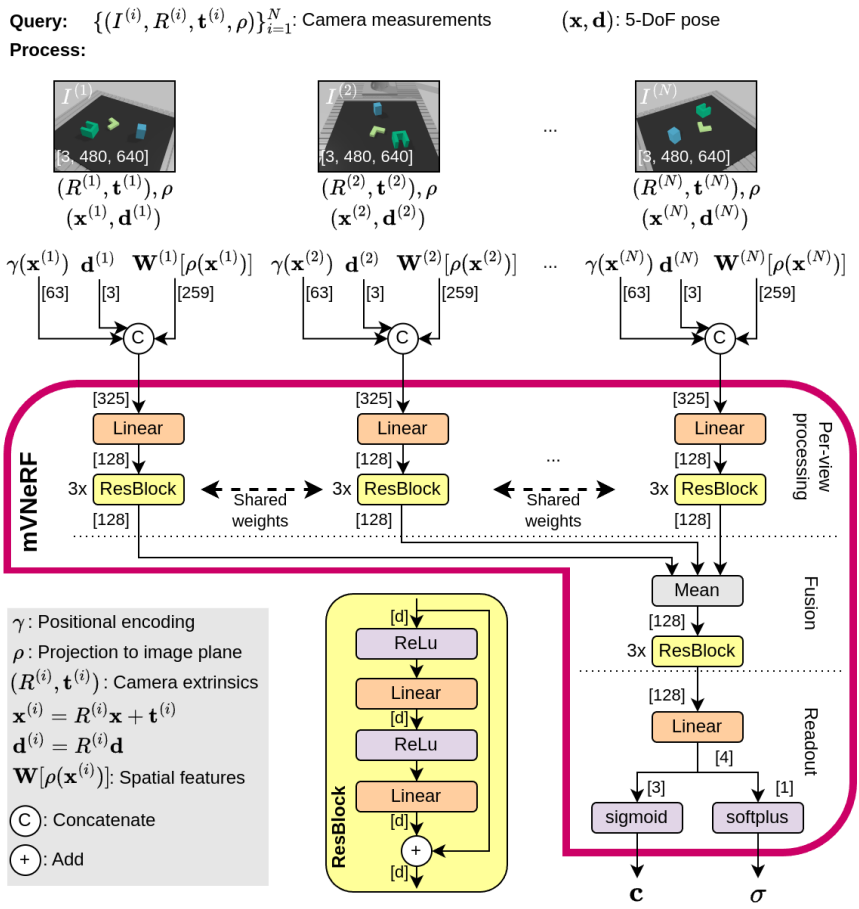


Figure 3.3: Multi-view VisionNeRF architecture. The model processes multiple input views by extracting features independently, aggregating them via mean pooling, and producing final color and density predictions.

thus the projection function is the same for all cameras and is simply denoted as ρ.

The per-view NeRF embeddings are then computed as:

$$f_1(\gamma(\mathbf{x}^{(i)}), \mathbf{d}^{(i)}, \mathbf{w}^{(i)}) = \mathbf{V}^{(i)}, \tag{3.13}$$

where $\gamma(\cdot)$ is the positional encoding (Equation 2.6).

These embeddings are aggregated using average pooling and are processed by the second part of the network

$$\mathbf{F} = \frac{1}{N} \sum_{i=1}^{N} \mathbf{V}^{(i)}, \quad f_2(\mathbf{F}) = (\mathbf{c}, \sigma), \tag{3.14}$$

resulting in the color and density predictions. The mVNeRF f_θ composes f_1 and f_2

$$f_\theta(\mathbf{x}, \mathbf{d}, o) = f_2\Big(\frac{1}{N} \sum_{\substack{(I,R,\mathbf{t},\rho) \\ \in o}} f_1\big(\gamma(R\mathbf{x}+\mathbf{t}), R\mathbf{d}, \mathcal{G}(I)[\rho(R\mathbf{x}+\mathbf{t})]\big)\Big), \tag{3.15}$$

to learn a function that maps 5-DoF poses to the scene properties color $\mathbf{c}$ and density σ, given a set of N camera measurements o.

The 1-view NeRF corresponds almost exactly to VisionNeRF, differing only in higher input resolution (640×480) and the ray sampling strategy during training. In the original VisionNeRF, rays are sampled within object bounding boxes with a white background, whereas here, rays are sampled uniformly at random.

To learn the geometrical and visual representation of the scene, mVNeRF is trained for novel view synthesis using the standard volume rendering loss (Equation 2.5). Hierarchical sampling with coarse and fine networks is also employed as described in Section 2.1.

Although this work targets grasping rather than novel view synthesis, rendering serves as validation that consistent geometric and visual representations are learned. Training details and results are presented in Section 3.3.1.

3.2.2 Support Pose Decomposition

To integrate 6-DoF TCP poses into the 5-DoF input space of NeRF (defined by a 3D position and a viewing direction), a Support Pose Decomposition (SPD) strategy is introduced. SPD decomposes a 6-DoF TCP pose into a fixed set of 5-DoF query points arranged to spatially bound the gripper's grasp volume. Each of these 5-DoF points can then be queried independently using the NeRF model. While NeRF typically returns color and density values for novel view synthesis, here the intermediate latent embeddings are extracted and used for grasp evaluation. This entire SPD process is differentiable, a critical property that enables end-to-end gradient flow from the NeRF representations back to the 6-DoF TCP pose parameters. This differentiability is essential for the gradient-based optimization employed by the implicit policy.

The decomposition is motivated by the physical structure of a two-finger parallel-jaw gripper, where a successful grasp corresponds to enclosing a region of the object between two opposing contact surfaces. The training data consists of prismatic objects with demonstrated grasps on these parallel surfaces, and the SPD ensures that the sampled 5-DoF query points within the gripper volume are positioned to capture information about these contact surfaces. Each sample point is paired with a viewing direction pointing towards the center region of the grasp volume and used to query the NeRF model.

A single action a (a 6-DoF TCP pose) is decomposed into a set of 5-DoF query points $\mathcal{F}(a)$ that bound the grasp volume:

$$\mathcal{F} : a \rightarrow \{s_1, s_2, ..., s_M\}, \tag{3.16}$$

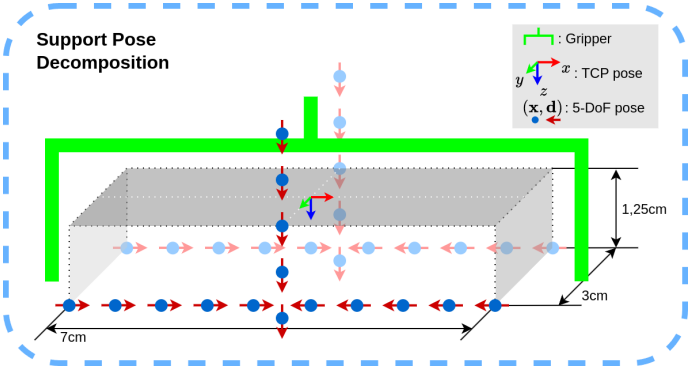


Figure 3.4: Support Pose Decomposition. A 6-DoF gripper pose a is decomposed into a set of 5-DoF query points $\mathcal{F}(a)$ that span the grasp volume. These points are used to query the NeRF model.

where $s_i = (\mathbf{x}_i, \mathbf{d}_i)$ defines a pose in the NeRF's 5-DoF space. Figure 3.4 illustrates this decomposition, depicting six principal axes – two for each gripper finger and two for the palm – along which support poses are selected. For visualization clarity, five poses are shown per axis in the figure, however, in the experiments, seven poses are selected along each axis, resulting in a total of 42 support poses.

This decomposition enables the NeRF to be queried using its original 5-DoF formulation, while still providing a mechanism to evaluate full 6-DoF TCP poses. The generation of the 5-DoF query points from an initial 6-DoF TCP pose is achieved by applying a fixed set of affine transformations to the input pose, followed by a projection to the 5-DoF space.

3.2.3 Grasp Value Model

The grasp value model Q defines the objective function for the implicit policy introduced in Section 3.2.4. It takes as input an observation $o \in \mathcal{O}$ and a candidate TCP pose $a \in \mathcal{A}$, and outputs a scalar value $q \in \mathbb{R}$ representing the likelihood that the grasp will succeed:

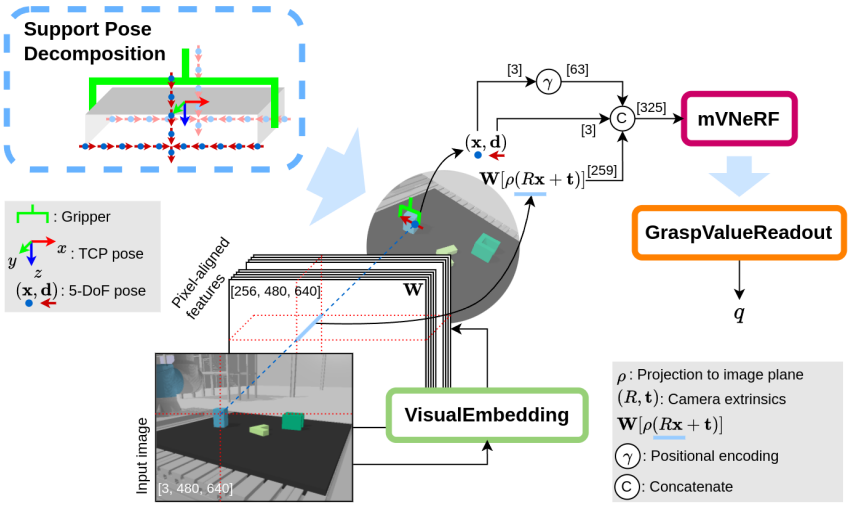


Figure 3.5: Grasp Value Model. The model consists of three stages: (1) Support Pose Decomposition, (2) latent embedding extraction using visual features and mVNeRF, and (3) aggregation and decoding into a scalar grasp value. The figure shows the pipeline with a single camera measurement and a single support pose.

$$Q : \mathcal{O} \times \mathcal{A} \to \mathbb{R}, \quad Q(o, a) = q. \tag{3.17}$$

The proposed model operates in three stages. First, the candidate pose a is decomposed into a set of 5-DoF position-direction pairs $\mathcal{F}(a)$ using SPD. Second, each of these 5-DoF queries is evaluated using the trained mVNeRF f_θ, conditioned on visual features extracted from the observation o via the visual embedding network $\mathcal{G}$. Rather than returning final color and density values as in novel view synthesis, intermediate activations from the fused processing stage of f_θ are extracted as latent embeddings. These serve as scene-aware descriptors for each support pose. Third, the embeddings are aggregated by a learnable grasp value readout module Q_ψ, which maps them to a scalar grasp value. This three-stage process is illustrated in Figure 3.5 for the 1-view case with one support pose highlighted.

The observation $o = \{(I^{(j)}, R^{(j)}, \mathbf{t}^{(j)}, \rho^{(j)})\}_{j=1}^{N} \in \mathcal{O}$ consists of calibrated camera measurements, where $I^{(j)}$ is an RGB image, $(R^{(j)}, \mathbf{t}^{(j)})$ are the extrinsics, and $\rho^{(j)}$ is the projection function derived from the camera's intrinsic matrix.

To compute the latent embeddings, the mVNeRF is queried at each support pose $s = (\mathbf{x}, \mathbf{d}) \in \mathcal{F}(a)$ using the visual features extracted from the observations. Rather than using the final network outputs (color and density values designed for novel view synthesis), intermediate activations are extracted that contain the learned geometric and spatial representations. These intermediate features encode the scene structure and geometric understanding acquired during volumetric rendering training, while the final outputs are specific to the rendering task. Given that the mVNeRF architecture is composed of six ResNet blocks – split into a per-view encoder f_1 and a fused processor f_2 – four activations per query are extracted: one from the aggregated output of f_1, and one from each of the three ResNet blocks in f_2. Together, these form the NeRF embeddings for s:

$$\hat{f}_\theta(s, o) = f_\theta^{\text{embed}}(\mathbf{x}, \mathbf{d}, o). \tag{3.18}$$

The grasp value function is defined by aggregating these embeddings across all support poses:

$$Q(o, a) = Q_\psi \left(\left\{ \hat{f}_\theta(s, o) \right\}_{s \in \mathcal{F}(a)} \right) = q, \tag{3.19}$$

where Q_ψ is the grasp value readout network. Q_ψ employs a two-stage design: first, each NeRF embedding is processed by a shared Support Pose Embedding Block to yield a single feature vector, then, the resulting features are concatenated and further processed to produce the scalar grasp value. The full architecture is shown in Figure 3.6.

The mVNeRF is pre-trained for novel view synthesis and remains frozen during training. This setup allows the grasp value model to leverage consistent, geometry-aware scene representations without requiring additional training. Only the grasp value readout Q_ψ is trained, constituting a transfer learning setup.

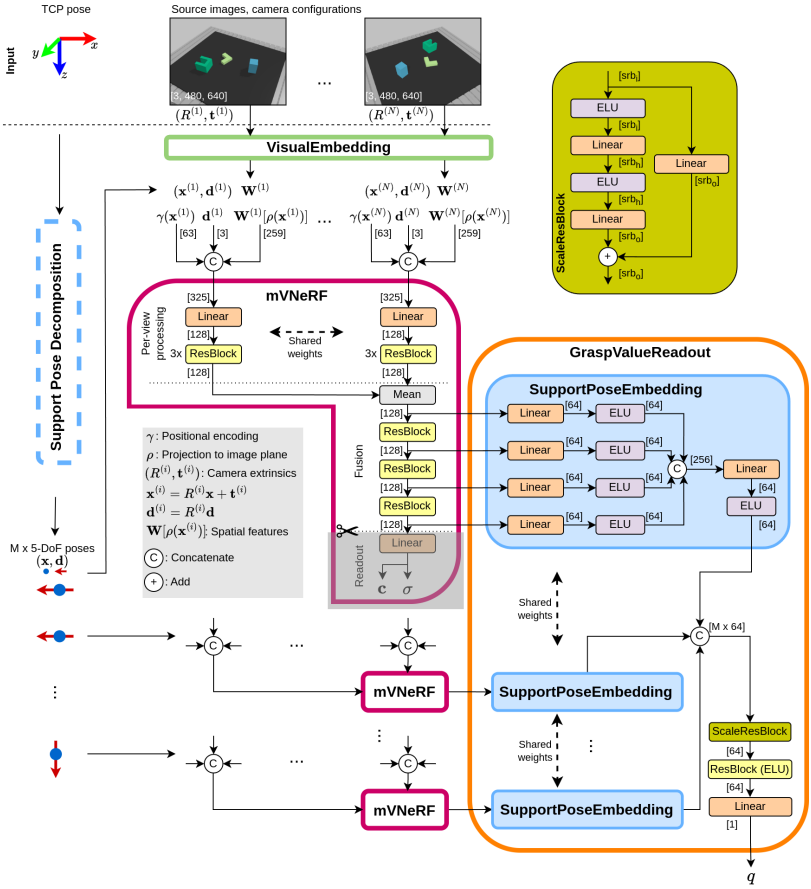


Figure 3.6: Grasp Value Readout. Each 5-DoF support pose is processed independently to produce a latent embedding. The embeddings are passed through a shared Support Pose Embedding Block and then aggregated via concatenation and further processed to produce a scalar grasp value. The ScaleResBlock's parameters are $srb_i = M \times 64$, $srb_h = 128$ and $srb_o = 64$.

Training the grasp value model is treated similarly to a binary classification task using negative-sampling. Demonstrated successful grasps are labeled with 1, while randomly sampled poses in the workspace are labeled with 0. The model is trained to assign high values to successful grasps and low values to unsuccessful ones.

Given an observation o and a set of TCP poses $\{a_i\}_{i=1}^{K}$, with a_0 being the demonstrated successful grasp and the rest are the negative samples, the grasp value model is applied to each candidate:

$$q_i = Q(o, a_i) \quad \text{for } i = 1, \ldots, K. \tag{3.20}$$

The outputs are normalized using softmax:

$$r_i = \frac{e^{Q(o,a_i)}}{\sum_{j=0}^{K} e^{Q(o,a_j)}}. \tag{3.21}$$

To reduce the difference between the predicted scores q and the labels p, Kullback-Leibler (KL) divergence loss is used:

$$\mathcal{L} = D_{\text{KL}}(p \| r) = \sum_{i=0}^{K} p_i \log \frac{p_i}{r_i}. \tag{3.22}$$

where $p_i \in \{0, 1\}$ denotes the binary label of candidate a_i.

Since only p_0, the label of the positive example a_0 is 1, and the rest are 0, the loss can be simplified to:

$$\mathcal{L} = \log \frac{1}{r_0} = -\log \frac{e^{Q(o,a_0)}}{\sum_{i=0}^{K} e^{Q(o,a_i)}}. \tag{3.23}$$

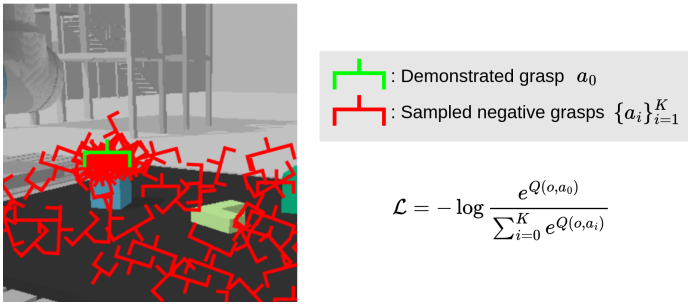


Figure 3.7: Grasp Value Loss. The single positive example a_0 is shown in green. Negative examples (red) are sampled uniformly across the workspace and additionally in the vicinity of the positive example. The loss encourages the model to assign high values to the positive example and low values to the negative ones.

Note: This is equivalent to the categorical cross-entropy loss with a single positive label:

$$H(p, r) = -\sum_{i=0}^{K} p_i \log r_i = -\log r_0, \tag{3.24}$$

but the implementation uses KL divergence loss instead.

To prepare the optimization landscape for gradient-based policy inference, negative examples are sampled using a dual strategy. First, grasp poses are sampled uniformly across the entire workspace. This encourages the model to learn broadly applicable gradients, guiding poor candidates toward more promising regions. Second, additional negative examples are sampled in the vicinity of the demonstrated grasp pose. This local sampling ensures that the optimization landscape around successful grasps is well-shaped, providing more informative gradients in regions that are critical for convergence. See Figure 3.7 for a visualization of the poses used for training.

3.2.4 Implicit Grasp Policy

The grasp value model Q, introduced in Section 3.2.3, serves as the objective function for the implicit grasp policy. Given an observation o, the optimal grasp pose $a^* \in \mathcal{A}$ is defined as the solution to the following optimization problem:

$$a^* = \underset{a \in \mathcal{A}}{\operatorname{argmax}} \, Q(o, a) \tag{3.25}$$

Solving this optimization problem is non-trivial. The search space $\mathrm{SE}(3)$ is large and continuous, and the objective function Q can exhibit multiple local optima. However, since Q is differentiable with respect to the pose parameters, gradient-based optimization can be applied. While gradient-based methods are typically susceptible to local optima, this is addressed by initializing optimization from multiple candidate poses in parallel, allowing the algorithm to explore different regions of the search space simultaneously. Moreover, multiple local optima are not necessarily problematic in this context, as they can correspond to different valid grasp configurations for the same object. By ascending the gradient of the grasp value function, candidate poses are iteratively refined toward regions of higher predicted grasp success.

This approach is conceptually related to iNeRF (Section 2.3), which uses gradient-based optimization to recover camera poses by minimizing image reconstruction error. While iNeRF optimizes camera poses using rays cast from the camera pose, the method proposed here optimizes 6-DoF TCP poses using the SPD. Furthermore, rather than minimizing reconstruction error, this work directly maximizes the estimated grasp value.

To improve robustness and to mitigate the risk of converging to undesirable local optima (i.e., ones that do not correspond to high-quality grasps), the optimization is initialized from a set of candidate poses $A = \{a_1, \ldots, a_L\} \subset \mathcal{A}$, sampled uniformly from the robot's workspace. Each candidate is refined independently using gradient ascent on the grasp value.

Algorithm 1 Implicit Grasp Policy

Require: Observation o
Ensure: Optimal grasp a^*
$A \leftarrow$ SampleInitialGraspCandidates() ▷ Uniformly in workspace
while not converged **do**
  for all $a \in A$ **do** ▷ Parallel refinement
    $a \leftarrow a + \eta \cdot \nabla_a Q(o, a)$ ▷ Gradient ascent
    $a \leftarrow$ NormalizePose(a) ▷ Ensure valid position + quaternion
  end for
end while
$a^* \leftarrow \mathrm{argmax}_{a \in A}\, Q(o, a)$
return a^*

Grasp poses are parameterized by a 3D position and an orientation represented as a unit quaternion. For both translation and orientation, parameters are updated by taking a linear step in the respective parameter space along the gradient; for the quaternion, this $\mathbb{R}^4$ step is applied element-wise to the components and the result is then normalized to unit length to ensure a valid rotation. This can be viewed as a linear update followed by a projection back onto the unit sphere. Although this procedure does not explicitly respect the manifold structure of $SO(3)$, it works well in practice due to the small step sizes employed during optimization. Alternative representations such as rotation vectors (used in iNeRF) and the dual vectors of screw theory were considered, quaternions combined with post-update normalization were chosen for their simplicity and stability in this application.

The optimization is performed using the Adam optimizer (Kingma and Ba (2014)), with separate instances for position and orientation. This setup allows independent learning rates and enables finer control over the update dynamics of each component. Additionally, the learning rate is decayed over time. After a fixed number of optimization steps, the candidate pose with the highest grasp value is selected. Empirically, separating the optimization of translation and rotation also yielded better results in this setting. The optimization procedure is outlined in Algorithm 1. Gradients are obtained via automatic differentiation through the grasp value model, including the NeRF.

3.3 Experiments and Results

The proposed pipeline is evaluated in four simulated scenarios and one real-world setting to systematically assess generalization capabilities across different object types, arrangements, and domains:

(a) *Prism-simple* - contains up to five simple monochromatic prismatic objects arranged flat non-overlapping in the workspace,

(b) *YCB-simple* - contains up to five objects from the YCB object set (Yale-CMU-Berkely object set, Calli et al. (2015)) arranged non-overlapping in the workspace,

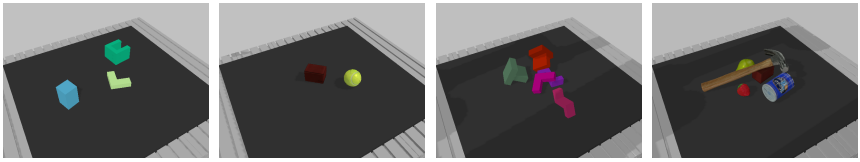
(c) *Prism-clutter* - contains up to five simple monochromatic prismatic objects arranged in a clutter,

(d) *YCB-clutter* - contains up to five objects from the YCB object set arranged in a clutter,

(e) *real-world* - contains a single object from fourteen 3D printed and everyday objects.

This experimental design enables evaluation along three key dimensions: (1) object geometry generalization from training prisms to complex YCB objects, (2) spatial arrangement complexity from isolated objects to cluttered scenes, and (3) domain transfer from simulation to real-world deployment. The *prism-simple* scenario serves as the training domain, while all others test out-of-distribution generalization. All experiments use an UR10e manipulator with a Robotiq 2F-140 gripper and an Intel RealSense D415 camera. Camera parameters in the simulation mirror the real setup. Prismatic object colors and light direction are randomized in simulation. Figure 3.8 shows examples from the simulated scenes, and Figure 3.9 shows the real-world objects.

The mVNeRF and the grasp value model are trained only on the *prism-simple* scenario, then evaluated unchanged in every other scenario – including the real robot – to test generalization and zero-shot sim-to-real transfer.



(a) *prism-simple* **(b)** *YCB-simple* **(c)** *prism-clutter* **(d)** *YCB-clutter*

Figure 3.8: Simulated scenarios.

Figure 3.9: Real world objects. Everyday objects used in the real world experiments.

The remainder of this chapter is organized as follows: Sections 3.3.1 and 3.3.2 describe the training details and results for the mVNeRF and the grasp value model, respectively, Section 3.3.3 examines the performance of the implicit policy in simulation, Section 3.3.4 reports the real-robot experiments that test zero-shot sim-to-real transfer, and Section 3.4 concludes the chapter with a summary.

> In the experiments, 1-view and 3-view models are trained and evaluated. The 1-view backbone serves as a single-image conditioned baseline to isolate the effect of multi-view fusion. The 3-view configuration provides cross-view constraints and occlusion handling with modest compute and data-capture overhead, and is inspired by pixelNeRF's real-world evaluation setup for novel view synthesis using three input images (Yu et al. (2021b)). For the grasp policy evaluation, even the 1-view backbone is used with three calibrated images with late fusion by summation to ensure comparability (see Section 3.3.3 for more details).

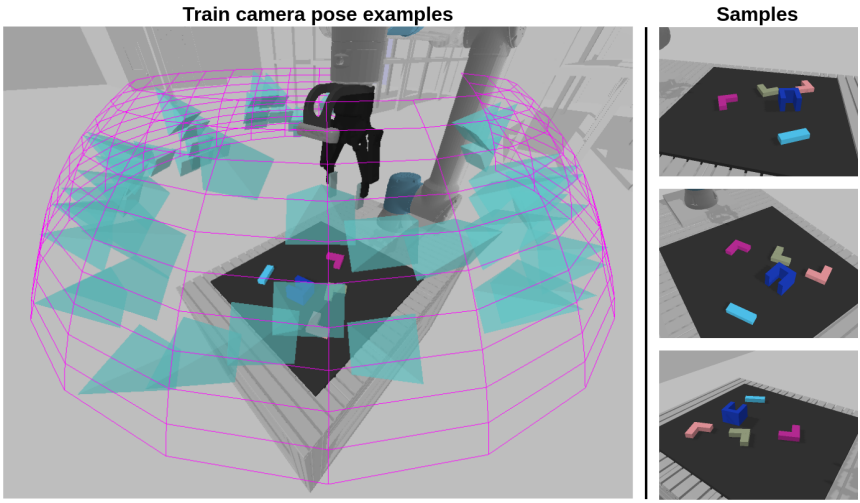


Figure 3.10: Example training camera setup. The camera poses for training the mVNeRF are sampled on the sphere surface indicated by the magenta grid. 50 random camera poses are indicated by the cyan pyramids, and sample images from the training dataset are shown.

3.3.1 Multi-view VisionNeRF

The mVNeRF is trained on 2504 scenes generated in the *prism-simple* setting. For each scene, 50 RGB images are rendered from random camera poses sampled on a sphere of radius 0.8m centred at the workspace origin. The polar angle is drawn uniformly from $[30°, 60°]$ and the azimuth from $[-150°, 150°]$ (Figure 3.10), yielding 125200 camera measurements in total.

Training mirrors VisionNeRF's training schedule. The models are trained for 1600 epochs with batch size of 8 resulting in 500k overall steps. The learning rates for the mVNeRF and the visual embedding are adapted during training using a warmup period and a fine-tuning phase. The learning rate schedule is summarized in table 3.1.

Table 3.1: Learning rate (lr) schedules for mVNeRF.

Phase	**Steps**	**mVNeRF** lr	**Visual-embed.** lr
Warm-up	0-10k	linear $\uparrow$ to 1×10^{-4}	linear $\uparrow$ to 1×10^{-5}
Main	10k-450k	1×10^{-4}	1×10^{-5}
Fine-tune	450k-500k	1×10^{-5}	1×10^{-6}

At each training step one target view and the required number of source views are sampled from the same scene and the volume rendering loss in Eq. (2.5) is evaluated on a random set of rays.

Held-out target images are compared to renderings with three standard metrics from the novel view synthesis literature: Peak Signal-to-Noise Ratio (PSNR), Structural Similarity Index (SSIM), and Learned Perceptual Image Patch Similarity (LPIPS).

PSNR measures the pixel-wise fidelity of the rendered image to the target image. With images normalized to the range $[0, 1]$, it is defined as

$$\mathrm{PSNR} = 10 \log_{10}\!\left(\frac{L_{\max}^2}{\mathrm{MSE}}\right) \text{ dB}, \tag{3.26}$$

where $L_{\max} = 1.0$ is the maximum pixel value and MSE is the mean squared error between the rendered and the target image. A single bright pixel error can drop PSNR sharply, while a small global luminance shift that leaves most pixels unchanged may still score high. Because it compares pixels one-for-one, PSNR says nothing about perceptual structure or texture – two images that look identical to a viewer can differ in PSNR if they are even half a pixel misaligned.

SSIM assesses local luminance, contrast, and structure (Wang et al. (2004)). It compares corresponding patches of two images and returns a single scalar value that can be aggreagted to characterize the overall similarity between the two images. For a pair of corresponding patches x and y the score is

$$\mathrm{SSIM}(x, y) = \frac{(2\mu_x\mu_y + c_1)(2\sigma_{xy} + c_2)}{(\mu_x^2 + \mu_y^2 + c_1)(\sigma_x^2 + \sigma_y^2 + c_2)}, \tag{3.27}$$

where μ_x, μ_y are patch means, σ_x^2, σ_y^2 their variances, σ_{xy} their covariance, and c_1, c_2 small stabilizing constants. Two patches are considered structurally similar when their means and variances match and their normalized covariance (cross-correlation) is high, which ties the score to shared edge and texture patterns rather than raw RGB equality. In regions rich in high-frequency detail (e.g., foliage or wires), PSNR can over-penalize slight blur or sub-pixel misregistration, whereas SSIM is more forgiving if the underlying structure remains intact. Conversely, SSIM may overlook perceptually important chromatic shifts in uniformly lit areas (e.g., sky tint) because those errors do not disturb local luminance or contrast.

LPIPS (Learned Perceptual Image Patch Similarity) measures how close two images appear to a human observer rather than how well their pixels line up (Zhang et al. (2018a)). It passes each image through a fixed, ImageNet-trained convolutional network (the AlexNet variant is used in this work), extracts the activations from several layers, and computes an averaged ℓ_2 distance between the two sets of features. Layers capturing low-level edges up to high-level semantics all contribute, each weighted by learned scalars tuned to match human similarity judgments. The result is a non-negative scalar where lower is better. Because it compares deep features, LPIPS is largely insensitive to sub-pixel misregistration yet picks up differences in texture, color tone, and object detail that PSNR and SSIM often miss.

The test dataset contains 320 scenes – 80 from each of the four simulated scenarios. In every scene, three fixed viewpoints serve as sources and six additional viewpoints as targets. All cameras sit on the same $0.8, \mathrm{m}$-radius sphere used for training and are oriented toward the workspace centre. The exact polar and azimuthal angles for every viewpoint are listed in Table 3.2, while Figure 3.11 visualises their spatial arrangement and shows example renderings. Using a fixed, disjoint set of three source and six target poses for every scene creates a consistent evaluation protocol: the network must render views that are never provided as inputs for that scene while still remaining within the overall viewing distribution encountered during training.

Table 3.2: Polar and azimuthal angles (in degrees) of the source and target views for the test dataset.

Angle	Source			Target					
polar	45	45	45	55	37.5	40	60	45	30
azimuthal	-120	0	120	-120	-60	0	30	90	110

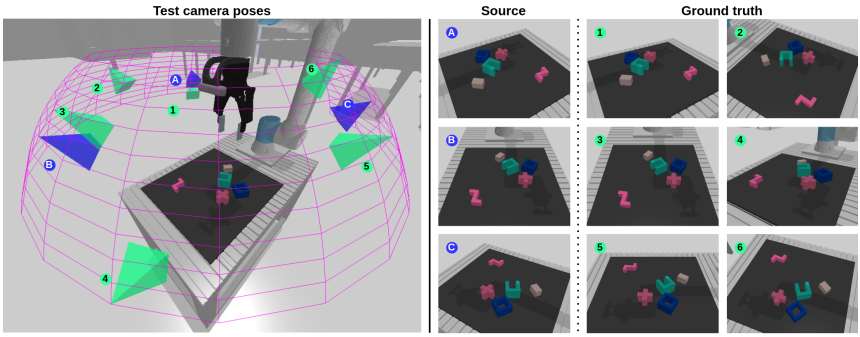


Figure 3.11: Evaluation camera setup. The source views are indicated by the blue pyramids and the target views by the green pyramids. The images show examples of the rendered images from the test dataset.

The novel view synthesis evaluation serves as a proxy test for the underlying scene representation: if a model can faithfully render a viewpoint it has never seen, it must have a representation of the workspace's geometry and appearance. Two models are trained: mVNeRF 1v (1-view) and mVNeRF 3v (3-view). The mVNeRF 1v model processes a signle view during training and evaluation, making it architecturally identical to VisionNeRF apart a few differences mentioned in Section 3.2.1, and acts as a baseline. The mVNeRF 3v model augments this baseline by fusing information from three calibrated images during training and evaluation. The experiment therefore investigates whether additional views translate into a measurably richer representation.

Figures 3.12 and 3.13 show rendering results that provide a qualitative measure of the models' performance.

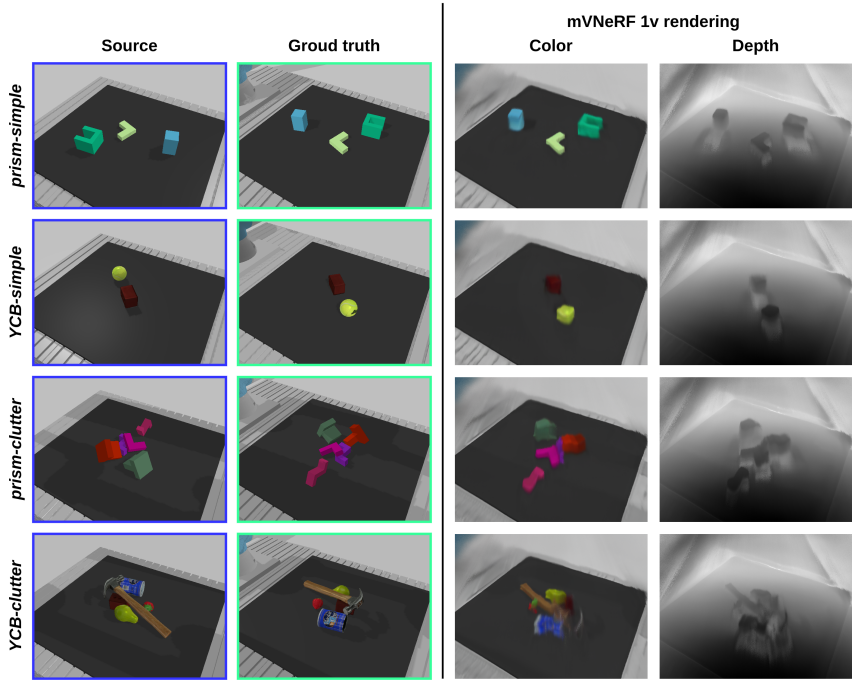


Figure 3.12: Rendering results mVNeRF 1v. The blue-bordered image serves as the source input to the NeRF along with its corresponding camera configuration (extrinsics and intrinsics). The model is used to render the scene from a different target camera configuration. The ground-truth image for that target configuration is highlighted with a green border. On the right are the model's renderings: RGB and depth.

MVNeRF 1v reproduces the scene's spatial layout but loses fine geometry and texture. Object boundaries are soft, small parts such as the hammer head almost merge with the dark work-surface, plausibly because both share similar gray values. Round items (e.g., the tennis ball, the can) appear faceted rather than smooth – an artefact that reflects the prism-only geometry seen during training. The texture and the geometry of the table are also smeared out. Depth predictions show approximately correct overall geometry but produce shadow-like artifacts that can be attributed to the single source view. For some objects, such as the hammer's shaft and the magenta L-shape, the model exhibits angular distortion

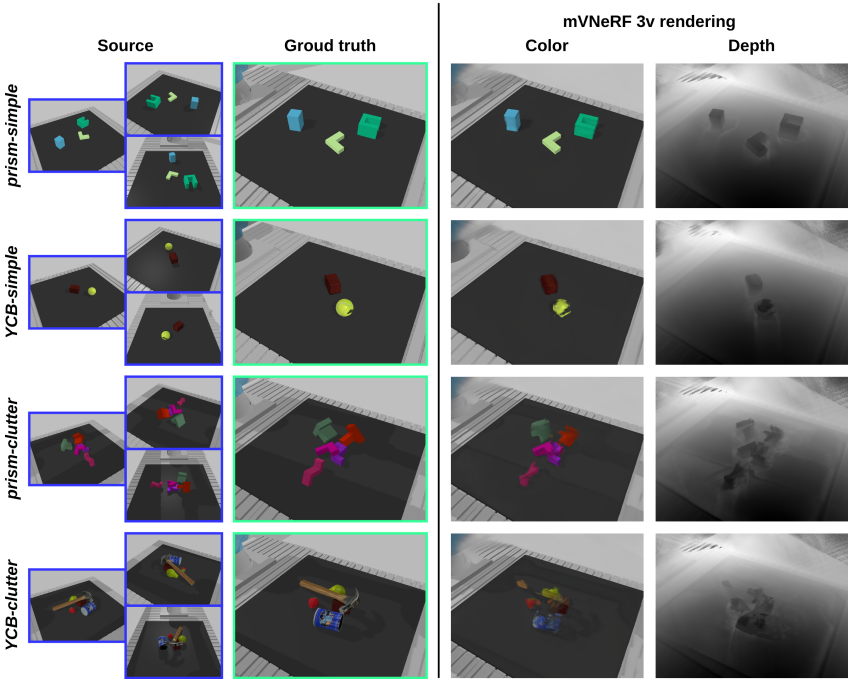


Figure 3.13: Rendering results mVNeRF 3v. The blue-bordered images serve as the source inputs to the NeRF together with their corresponding camera configurations (extrinsics and intrinsics). The source images are treated equivalently, their order carries no semantic meaning. The model is used to render the scene from a different target camera configuration. The ground-truth image for that target configuration is highlighted with a green border. On the right are the model's renderings: RGB and depth.

that makes them appear more flat on the table, an artefact likely due to the training data featuring only objects lying flat on the workspace.

The reconstructions of the mVNeRF 3v are sharper and photometrically more faithful. Lighting effects such as table reflections and shadows are also better captured. Overall, the rendering of the workspace is significantly more accurate than the 1-view model. In case of in-distribution scenes (*prism-simple*) the model produces very high quality results, displaying crisp edges and smooth surfaces in both RGB and depth. For out-of-distribution objects (YCB items

Table 3.3: MVNeRF novel view synthesis results. The average PSNR, SSIM and LPIPS for the 1-view and 3-view mVNeRF are summarized. The 1-view variant serves as our baseline and closely follows the VisionNeRF implementation (minor differences, such as the image resolution and ray sampling strategy during training, are discussed in the text). Here, 1-view and 3-view denote the number of input views used both during training and evaluation. Arrows indicate metric preference: ↑ higher is better, ↓ lower is better.

Metric	***prism-simple***		***YCB-simple***		***prism-clutter***		***YCB-clutter***		**mean**	
Views	**1**	**3**	**1**	**3**	**1**	**3**	**1**	**3**	**1**	**3**
PSNR ↑	27.02	**31.91**	25.88	**30.14**	25.75	**29.73**	24.92	**28.00**	25.89	**29.94**
SSIM ↑	0.92	**0.97**	0.90	**0.96**	0.91	**0.96**	0.89	**0.94**	0.91	**0.96**
LPIPS ↓	0.17	**0.07**	0.18	**0.08**	0.17	**0.08**	0.20	**0.10**	0.18	**0.08**

and cluttered layouts) edge sharpness still improves, yet many shapes are only partially reconstructed, object parts fade in both RGB and depth. This effect is especially well observed in the depth renderings, where the 1-view variant preserves object geometry more consistently than the 3-view model, showing that additional views cannot fully compensate for a mismatch between training geometry and test objects.

Table 3.3 shows the PSNR, SSIM, and LPIPS scores for the 1-view and 3-view mVNeRF variants. Across all four simulated scenarios, the 3-view model gains roughly $+4$ dB PSNR and $+0.05$ SSIM, while its LPIPS score is reduced by more than half. The improvement is consistent in both uncluttered and cluttered scenes and for both prism and YCB object sets.

While the numerical metrics confirm a clear overall advantage for the 3-view model, they mask the shortcomings visible in the qualitative results: because the objects cover only a small portion of each image, errors on those objects contribute little to global scores such as PSNR, SSIM, and LPIPS.

A complete study of novel view synthesis would examine several factors: how performance changes as the angle or distance between source and target views increases, how the model handles camera settings that differ from those seen in training, and how it reacts to significant scenario changes. While the reported scores include scenes with cluttered object layouts and unseen YCB items, both

outside the training distribution, no detailed analysis is performed to examine how these factors influence novel view synthesis. Such an investigation is beyond the scope of this work, which focuses on robotic grasping rather than view synthesis itself.

3.3.2 Grasp Value Model

The grasp value model is trained on 512 scenes from the *prism-simple* scenario. For each scene, 50 RGB images are rendered from random camera perspectives sampled identical to the camera sampling scheme used for training the mVNeRF (Figure 3.10). Additionally, a ground truth grasp pose is provided for each object of the scene.

The four configurations trained are summarized in Table 3.4. Here, *base grasp* denotes models that share the same mVNeRF-based architecture as in Section 3.2.3, but the mVNeRF backbone is *not* pre-trained for novel view synthesis. Instead, the backbone weights are initialized from scratch and optimized jointly with the grasp value readout (i.e., the mVNeRF weights are not frozen), while the visual transformer component remains the same pre-trained ViT used in mVNeRF. This configuration isolates the effect of novel view synthesis pretraining, enabling an explicit test of whether using view synthesis as an auxiliary pretraining task benefits grasping. For the variants labeled *mVNeRF grasp*, the mVNeRF backbone is first trained for novel view synthesis and then kept frozen during grasp training. Freezing preserves a reusable, task-agnostic visual scene representation that can support downstream tasks beyond grasping. The 1-view models use a single source camera measurement during training, while the 3-view models use three.

Training runs for 3200 epochs with a mini-batch size of 16, using the Adam optimizer. In the mVNeRF based models only the grasp value readout part is trained with a learning rate of 1×10^{-4}, and the visual embedding and the mVNeRF weights are frozen. For the base grasp models the whole network is trained. The ViT part has a learning rate of 1×10^{-5} while the visual embedding, the mVNeRF and the grasp value readout have a learning rate of 1×10^{-4}. At

Table 3.4: Grasp value model variants and learning-rate settings.

Model	mVNeRF backbone	# source views	Trainable parts / LR
base grasp 1v	untrained	1	Visual embedding: 10^{-5}, mVNeRF: 10^{-4}, grasp value readout: 10^{-4}
mVNeRF grasp 1v	pre-trained (frozen)	1	grasp value readout: 10^{-4}
base grasp 3v	untrained	3	Visual embedding: 10^{-5}, mVNeRF: 10^{-4}, grasp value readout: 10^{-4}
mVNeRF grasp 3v	pre-trained (frozen)	3	grasp value readout: 10^{-4}

every step the required number of camera measurements and a single demonstrated grasp are drawn from the same scene. For computing the loss from Eq. (3.23), 191 negative samples – grasp poses sampled randomly from the workspace – are drawn for each positive sample. Of these 191 negative samples, 24 are sampled in proximity to the positive sample.

The direct evaluation of a grasp value model is challenging, because its practical usefulness emerges only when it drives the implicit policy. As a qualitative assessment, Figure 3.14 visualizes the value landscape in a neighborhood of a successful grasp. It shows how the predicted score changes when the candidate pose is perturbed along each pair of translational axes (t_x, t_y, t_z) and rotational axes (r_x, r_y, r_z) by up to ± 5 cm in translation and $\pm 30°$ in rotation. As input camera measurements, the same 3 source perspectives as used for the mVNeRF evaluation are used (Figure 3.11).

All four models produce similar value landscapes, each with a clear peak at the center – the demonstrated grasp pose. The two base grasp models yield rougher surfaces with more local extrema, most notably when translation along t_z is involved. In contrast, the mVNeRF-based variants exhibit smoother, more regular landscapes.

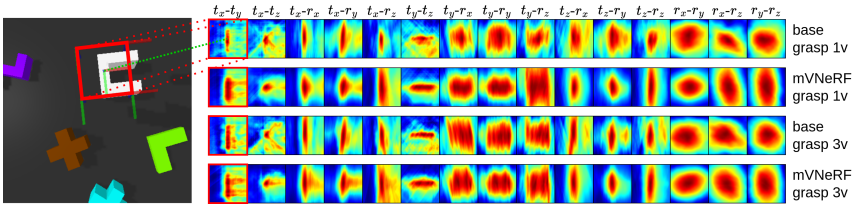


Figure 3.14: Grasp values landscapes around a successful grasp pose. The grasp values of the base and mVNeRF grasp models visualized in a neighborhood of a successful grasp pose. The patches correspond to perturbations of the grasp pose along the translational and rotational axes with the successful grasp pose in the center. Warm colors indicate high grasp values, while cool colors indicate low values.

The main distinction between the 1-view and 3-view mVNeRF models appears along the t_y direction. Lateral motion in t_y should still result in a valid grasp here. Both assign higher values overall in this direction but the 3-view model fragments the region, whereas the 1-view model maintains a more uniform, consistently elevated band.

Overall, transfer-learning from a NeRF that was first trained for novel view synthesis produces a noticeably smoother value surface – an advantage for any gradient-based search. Interestingly, the 1-view mVNeRF, although weaker than its 3-view counterpart in novel view synthesis, delivers the most uniform and optimization-friendly landscape. A likely reason is that, with only a single image available, the 1-view network is forced to encode richer geometric cues in its latent space, whereas the 3-view model can lean on redundant inputs and consequently yields a more fragmented value field.

3.3.3 Implicit Grasp Policy - Simulation

All grasp value model variants are evaluated on the four simulated scenarios with the gradient-based implicit policy introduced in Section 3.2.4. All policies use the Adam optimizer for translation and rotation. The translational optimizer starts at 0.01 and decays by 0.93 per step, the rotational optimizer starts at 0.98 with a

decay factor of 0.95. All policies are initialized with 8000 grasp candidates and are executed for 32 steps.

For each scenario, 20 scenes are generated and used for evaluation with the same test camera poses as used for the mVNeRF evaluation (Section 3.3.1). This makes a total of 80 test scens containing an overall of 214 objects to grasp.

While the 3-view models receive three calibrated images per scene, the 1-view models natively process only one. To allow a fair comparison, the latter are also run with three images by summing the independently computed grasp values:

$$\tilde{Q}(o, a) = \sum_{i=1}^{3} Q(c_i, a), \tag{3.28}$$

where c_i is a set containing a single camera measurement. This treats the perspective as a batch of inputs and fuses information after the forward passes, in contrast to the internal fusion performed by the native 3-view networks.

During the experiments, two metrics are measured for evaluation:

- **Strict success rate:** the same number of trials as objects are allowed, a trial counts as success if the object is lifted.
- **Object-lifted rate with recovery:** up to twice as many attempts as objects are allowed, the metric reports the fraction of objects eventually lifted.

Each experiment is repeated four times with mean values and one-standard-deviation bands reported. Figures 3.15 and 3.16 show performance progression for models after 400, 800, 1600 and 3200 training epochs. Tables 3.5 and 3.6 list the mean metrics of the final models (3200 training epochs).

During training progression, the mVNeRF based policies consistently outperform the base models in the strict success rate metric even during the early phases of training. The overall best performing model is the mVNeRF grasp 3v policy, closely followed by the mVNeRF grasp 1v policy.

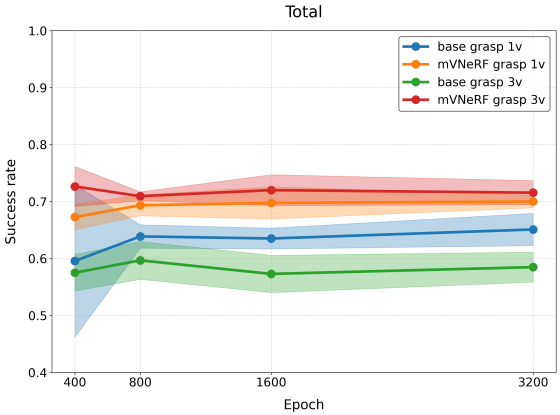


Figure 3.15: Strict success rate versus training epoch. Shaded regions indicate $\pm 1\sigma$.

Table 3.5: Strict success rate (final model at 3200 epochs).

Policy	prism-simple	YCB-simple	prism-clutter	YCB-clutter	mean
base grasp 1v	0.77	0.54	0.72	0.51	0.64
mVNeRF grasp 1v	0.83	**0.64**	**0.85**	0.47	0.70
base grasp 3v	0.71	0.46	0.61	0.52	0.58
mVNeRF grasp 3v	**0.85**	0.55	**0.85**	**0.55**	**0.71**

This difference is not clearly visible for the object-lifted rate with recovery metric, where all models perform similarly well during training progression. The mVNeRF grasp 3v policy achieves the highest score, and while in most scenarios the mVNeRF based policies perform better than the base models, in the cluttered YCB scenario the base grasp 3v policy outperforms the mVNeRF based policies by a significant margin.

Overall, the advantage of using a pre-trained mVNeRF for the grasp value model is evident, especially based on the strict success rate metric. All models are capable of a certain degree of generalization. In case of the YCB objects in simple arrangement, the mVNeRF based policies clearly outperform the base models, while in the cluttered YCB scenario this tendency is less pronounced.

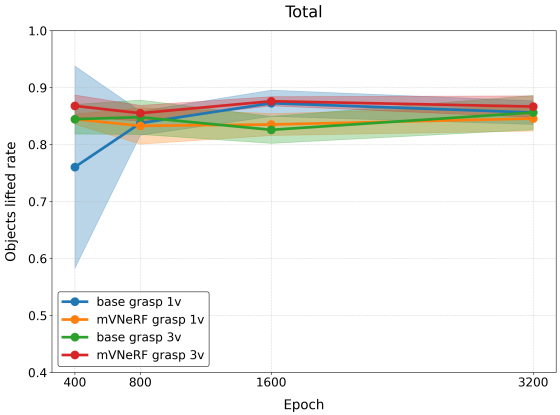


Figure 3.16: Object-lifted rate with recovery versus training epoch. Shaded regions indicate $\pm 1\sigma$.

Table 3.6: Object-lifted rate with recovery (final model at 3200 epochs).

Policy	prism-simple	YCB-simple	prism-clutter	YCB-clutter	mean
base grasp 1v	0.98	0.70	0.97	0.75	0.86
mVNeRF grasp 1v	0.98	**0.76**	**0.99**	0.64	0.85
base grasp 3v	0.95	0.68	0.94	**0.81**	0.86
mVNeRF grasp 3v	**0.99**	0.74	0.98	0.72	**0.87**

3.3.4 Implicit Grasp Policy - Real-World

The proposed pipeline is finally tested on the real robot to evaluate zero-shot sim-to-real transfer. Crucially, *no model retraining, domain adaptation, or intermediate processing steps are employed* – the policies trained exclusively on simulated *prism-simple* data are applied directly to real-world scenes without modification.

The setup uses the same UR10e manipulator with a Robotiq 2F-140 gripper and a wrist-mounted Intel RealSense D415. Observations are recorded by moving the camera to three predefined viewpoints, each $0.8\mathrm{m}$ from the workspace centre and oriented toward it. Limited reach prevents using the exact

Table 3.7: Real-world grasp success: successes out of ten trials per object, the rightmost column shows overall success percentage.

Policy (views)	blue cube (3D)	red block (3D)	red cylinder (3D)	tennis ball	crochet ball	rubber duck	hammer	soap dispenser	dental floss	large LEGO tire	small LEGO tire	black tri. prism (3D)	power drill	hiking boot	**Mean**
base grasp 1v	4	**9**	4	**6**	**6**	**8**	7	4	1	0	0	0	1	3	38%
mVNeRF grasp 1v	3	7	**5**	**6**	4	**8**	**9**	**6**	**8**	**7**	0	0	**5**	**4**	**51%**
base grasp 3v	0	8	4	5	**6**	2	**9**	2	3	0	0	0	3	1	31%
mVNeRF grasp 3v	**8**	7	1	0	1	6	**9**	1	6	3	0	0	**5**	**4**	36%

simulated poses, so the following azimuth/polar angles (degrees) are chosen: $(-80, 45)$, $(40, 30)$, $(160, 45)$. These retain the $120°$ azimuth spacing while lowering the second polar angle to $30°$.

Fourteen objects – ten everyday items and four 3D-printed objects – are placed one at a time at random positions in the workspace (Figure 3.9). Each model attempts ten grasps per object, giving $14 \times 10 = 140$ trials per policy.

Figure 3.18 and table 3.7 show the per-object success counts for each policy.

All models demonstrate successful zero-shot sim-to-real transfer, with the mVNeRF grasp 1v policy achieving the highest overall success rate of 51% despite never

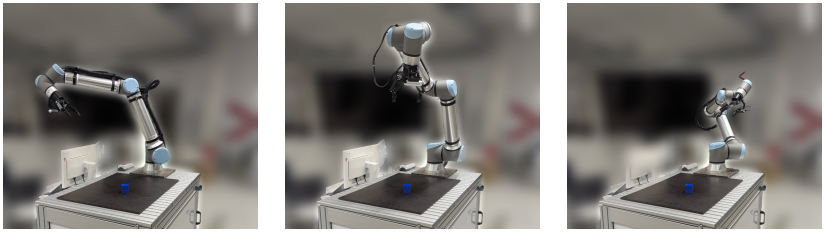


Figure 3.17: Real-world observation poses. The robot moves the camera to three predefined viewpoints to record the observations.

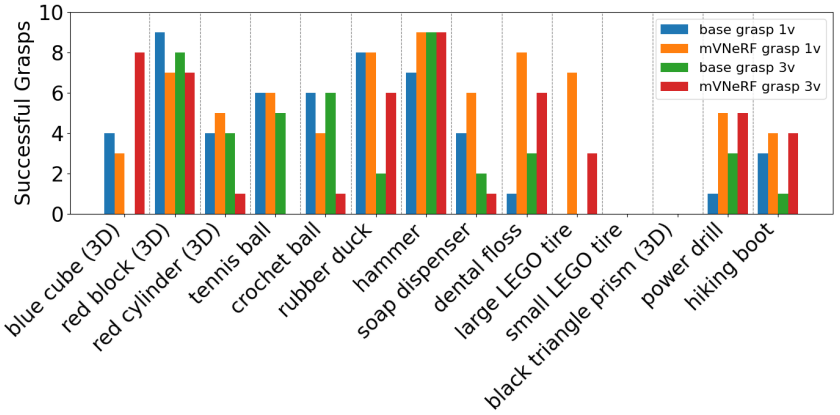


Figure 3.18: Grasp success histogram for the real-world experiments.

seeing real-world data during training. This performance is achieved through the geometric understanding acquired during novel view synthesis training on simulated scenes. Figures 3.19 and 3.20 illustrate typical successes and failures: the three left images show the observations with the five best grasp candidates (largest frame = highest value), and the right image shows the executed grasp.

The 3D printed blue cube and red block most closely match the simulated training objects. While the red block is grasped fairly consistently, every policy except the mVNeRF grasp 3v struggles with the cube (Figures 3.19 (a,b)). 1-view variants sometimes stop about $45°$ off the correct orientation on box-shaped items like the cube or the soap dispenser (Figure 3.20 (a)). Summing three single-image scores can flatten the landscape at saddle points, keeping the optimizer from rotating to the correct pose, the object then slips from between the fingers as they close.

Objects with moderate contrast against the dark background are located by the base models in many cases and almost always by the mVNeRF policies, even when the grasp fails. Overall, the mVNeRF policies struggle more with round objects – the red cylinder, tennis ball, and crochet ball – because both the training set and the SPD bias the network toward surfaces parallel to the gripper's fingers. During optimization, grasp candidates tend to slide off these curved surfaces and

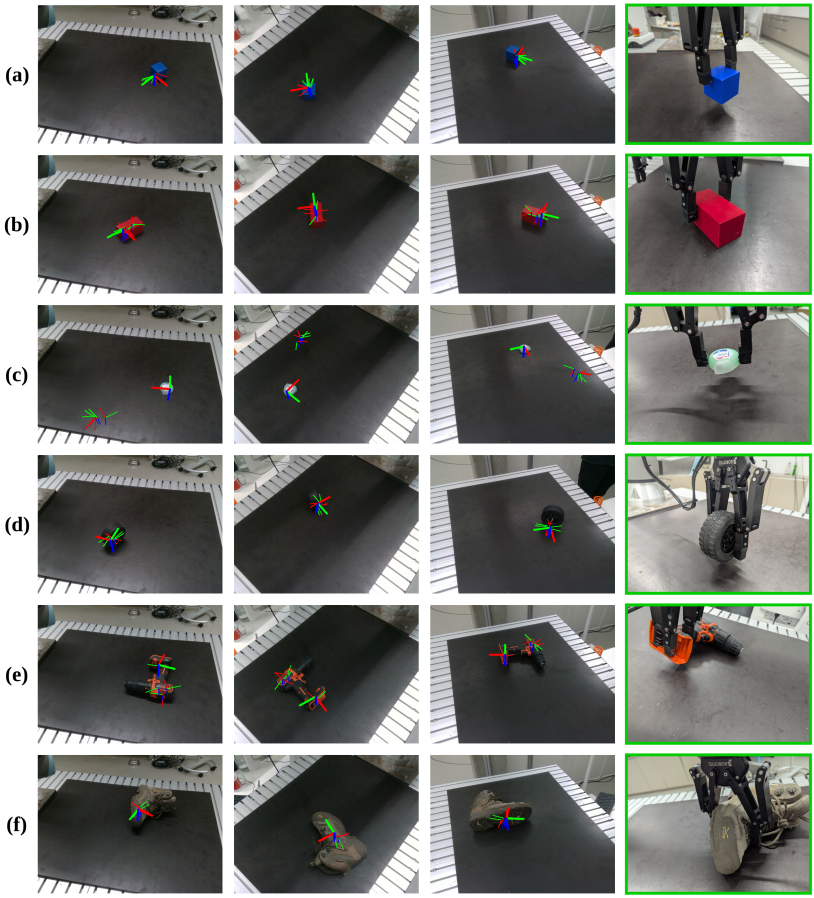


Figure 3.19: Real-world success cases.

drift off onto the flat workspace (Figure 3.21 (b)). The dental floss container, though small, has high background contrast, accordingly the mVNeRF policies pick it up (Figure 3.19 (c)), whereas the base grasp 3v policy often misses it and the base grasp 1v policy stops just above it.

A common failure mode is that the optimizer drifts to the workspace border and closes on empty space. This pose usually sits on a bright reflection patch

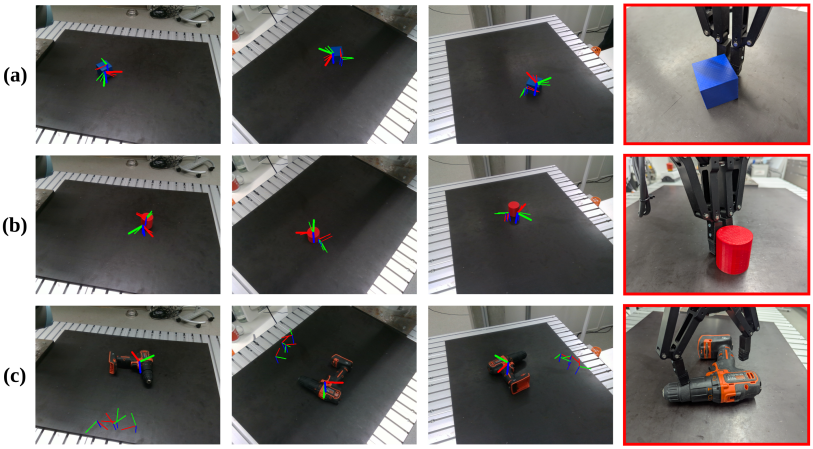


Figure 3.20: Real-world failure cases 1: In many cases the policy produces a grasp that is close to an object, but fails due to minor position errors, orientation errors, slipping, or collisions.

(Figure 3.21), the same spot reached when no object is present. Especially for dark objects that provide few visual cues, such as the LEGO tires and the black triangular prism, base models always end up at this pose. The mVNeRF policies partially share this bias, but still grasp the large LEGO tire (Figure 3.19 (d)), demonstrating the advantage of NeRF-based representations and their geometric prior in low-contrast scenes.

The power drill and hiking boot are larger than the other objects and combine dark textures with concave geometry. Base models again end up at the workspace edge. The mVNeRF policies locate these objects, yet the robot often collides with the object during the approach, so the lift fails even though the pose is close to correct (Figures 3.19 (e,f) and 3.20 (c)).

Due to the random initial grasp candidates, the policies are non-deterministic. With multiple objects present, they often return a multimodal set of high-scoring candidates that cover several items in different orientations (Figure 3.22).

Overall, the mVNeRF grasp 1v achieves the highest real-world success, while the 3-view version – although strongest in simulation – performs worse on the robot.

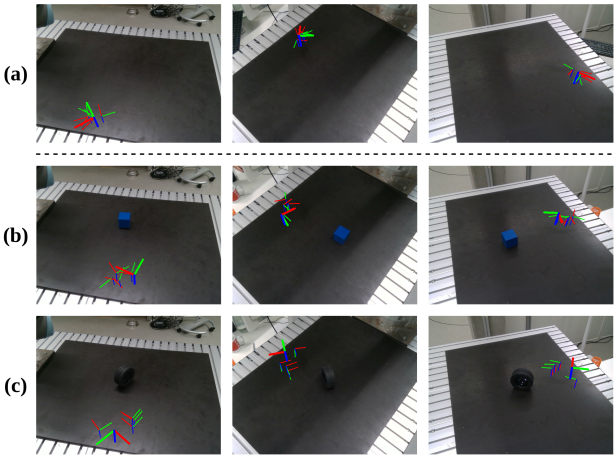


Figure 3.21: Real-world failure cases 2: The policy converges to a reflection artefact on the workspace floor, which is the same spot it converges to if there are no objects in the scene.

Figure 3.22: Real-world multimodal grasp behavior: The highest scoring grasp candidates in the same scene are diverse covering multiple objects in multiple orientations.

This outcome matches the rendering observations in Section 3.3.1: mVNeRF 3v images often miss parts of objects, whereas the 1-view model, though blurrier, retains most geometric consistency. These results demonstrate that the geometric representations learned through volumetric rendering generalize across the simulation-reality gap.

3.4 Summary

This chapter presented a NeRF-based pipeline for robotic grasping and evaluated it in simulation and on a real robot.

First, VisionNeRF was extended to handle multiple source images. The resulting **mVNeRF** fuses per-image visual features by mean pooling, which substantially **improves PSNR, SSIM, and LPIPS** in novel-view synthesis. Although the 3-view network achieves the highest image-quality metrics, the 1-view variant renders unseen objects more geometrically consistently.

To convert the pre-trained mVNeRF into a grasp evaluator, **SPD** maps each 6-DoF grasp candidate to a fixed set of 5-DoF NeRF queries spanning the gripper volume. Intermediate activations from the frozen mVNeRF are aggregated by a read-out network trained on grasp demonstrations. This transfer-learning setup yields a **grasp value model without re-training the scene representation**. Exploiting the model's differentiability, the implicit policy refines randomly initialized grasps via gradient ascent in $SE(3)$ and selects the highest scoring grasp.

Single-view and multi-view policies – with and without a pre-trained mVNeRF backbone – were trained on using a simple simulated grasping task and tested on four simulated scenarios and in the real world. All variants exhibit some generalization to novel objects, poses, and even **zero-shot sim-to-real transfer**. The mVNeRF grasp 1v policy delivers the overall best results, attaining the highest **real-world success rate of 51%**, mirroring the more geometrically consistent depth renderings of the 1-view variant (see Figures 3.12 and 3.13).

The next two chapters build on this foundation but serve different purposes. Chapter 4 is a primarily theoretical contribution: it strengthens training by exploiting full demonstration trajectories rather than just the final grasp pose. Chapter 5 showcases the framework's versatility by pairing it with two established ideas – federated training of NeRF backbones across multiple sites and language-conditioned grasping via CLIP embeddings.

4 Trajectory-Informed Grasp Value Learning

The NeRF-based implicit grasp policies introduced in Chapter 3 demonstrate successful zero-shot sim-to-real transfer, yet analysis of the optimization landscapes reveals several opportunities for improvement. The learned grasp value functions exhibit local extrema and irregular gradients that can trap the gradient-based optimization in suboptimal solutions. More fundamentally, the current training approach uses only final grasp poses for supervision, discarding the rich trajectory information present in demonstration data that could guide the optimization process more effectively.

This chapter addresses these limitations through two enhancements that preserve the core NeRF-based architecture while improving optimization reliability. First, a trajectory loss is introduced that leverages full demonstration trajectories to shape the gradient field, creating structured pathways through pose space that guide optimization toward successful grasps. Second, the policy's inference-time hyperparameters are optimized through Bayesian search to improve gradient-based pose refinement.

The chapter is structured as follows. Section 4.1 introduces the trajectory loss, describes its architectural implications, and details the Bayesian tuning of optimizer hyperparameters. Section 4.2 evaluates the effects of these modifications, and Section 4.3 concludes with a summary.

Note: This chapter builds upon previous work (Sóti et al. (2025)), presenting the latest implementation of the grasping framework along with some modifications. These modifications are summarized in Appendix A.2.

4.1 Method

The overall grasp pipeline remains unchanged: the same image-conditioned NeRF backbone, SPD, grasp value model, and implicit grasp policy formulation from Chapter 3 are retained.

In this chapter, for grasp value model training, the loss is augmented with a trajectory loss term. In Chapter 3, the policy uses gradients of the learned grasp value function to drive grasp candidates along trajectories in $SE(3)$ that ideally converge to successful poses. These gradients can be conceptualized as a TCP displacement field that indicates how to move from the current pose toward higher grasp values. While this interpretation is straightforward for the translational components, it is mathematically imprecise for rotational components: quaternion gradients do not represent proper angular displacements since quaternions compose through multiplication rather than addition. However, this approximation remains practically useful because the small optimization steps and adaptive learning rates ensure that the linear quaternion updates, followed by normalization, provide reasonable local approximations of rotational adjustments. Each recorded grasp demonstration already contains a reference TCP trajectory leading to a successful grasp. The trajectory loss encourages the learned gradient field to align with these reference trajectories, thereby shaping the optimization landscape to guide pose optimization along demonstrated TCP movements. The formulation of the trajectory loss presented in Section 4.1.1, and its architectural implications are discussed in Section 4.1.2.

Additionally, the policy's gradient-ascent optimizers (one for translation, one for rotation) are fine-tuned via Bayesian optimization. The process is detailed in Section 4.1.3.

4.1.1 Trajectory Loss

In Chapter 3, the optimization landscape of the implicit grasp policy is shaped by a loss function that only considers the executed grasp pose (Equation 3.23). In the following, this loss is referred to as the *pose loss* $\mathcal{L}_P$.

Incorporating trajectory information, results in the grasp value function Q, that satisfies the following property:

$$a_{t+1} = a_t + \nabla_a Q(o, a_t) \tag{4.1}$$

with a_t as the TCP pose at timestep t and a_{t+1} as the pose at a later timestep $t+1$ from a demonstrated trajectory.

The motivation for this approach is that pose-only supervision creates a challenging learning problem: the grasp value function must encode how to reach successful poses from arbitrary starting locations throughout the workspace. This results in a complex optimization landscape where the gradient field must implicitly learn all possible paths to success. Trajectory supervision addresses this by creating structured pathways through the pose space. Rather than learning arbitrary routes from every location, the gradient field is trained to guide optimization toward and along demonstrated successful trajectories. This creates a hierarchical structure where poses near demonstrated paths are directed toward these "highways" to success, significantly simplifying the learning problem and providing more reliable gradient flow during policy optimization.

The trajectory loss supervises the gradient field $\nabla_a Q$ to align with the demonstrated TCP displacements $a_{t+1} \ominus a_t$. For a single pose pair, this term is

$$\ell_T = -S_C(a_{t+1} \ominus a_t, \nabla_a Q(o, a_t)) \tag{4.2}$$

where $S_C(\cdot, \cdot)$ is the cosine similarity, computed separately for the translational and rotational components and then summed. The operator $\ominus$ denotes the element-wise difference in the chosen pose representation (translation vectors and quaternions). Although the element-wise subtraction $q_{t+1} \ominus q_t$ does not strictly respect

the quaternion manifold, it works well in practice because of the way the gradient and optimizer interact. Cosine similarity is chosen over magnitude-sensitive metrics (such as L2 distance) because it focuses supervision exclusively on gradient direction rather than magnitude. This design choice leverages the Adam optimizer's adaptive update mechanism, which automatically scales step sizes based on local curvature and gradient history. Since the policy optimization already handles magnitude adaptation through Adam's per-parameter learning rates and momentum terms, the trajectory loss need only ensure that gradients point in the correct direction. This separation of concerns – direction from trajectory supervision, magnitude from adaptive optimization – prevents conflicting signals during training and allows both mechanisms to operate in their respective strengths. Furthermore, as learning rates decay, the resulting small, incremental updates make the naive linear difference a valid local approximation of true pose changes in $SE(3)$.

Similar to the negative sampling used for the pose loss, the poses along a trajectory are augmented to compute the trajectory loss (Figure 4.1). For an observation o and current and future TCP poses a_t and a_{t+1}, a set of poses A^t is sampled in the proximity of a_t. The loss is then computed on $a_{t+1} \ominus a_r$ and $\nabla_a Q(o, a_r)$ for each sampled pose $a_r \in A^t$:

$$\mathcal{L}_T = \sum_{a_r \in A^t} \big[- S_C(a_{t+1} \ominus a_r,\ \nabla_a Q(o, a_r)) \big]. \tag{4.3}$$

This formulation encourages gradients at any nearby pose a_r to point toward the next demonstration pose a_{t+1}, under the assumption that moving in that direction continues to improve grasp success. In effect, it widens the basin of attraction around the demonstrated trajectory and provides a consistent corrective "push" toward successful grasps.

The combined loss for the grasp value model is

$$\mathcal{L} = \mathcal{L}_P + \mathcal{L}_T \,, \tag{4.4}$$

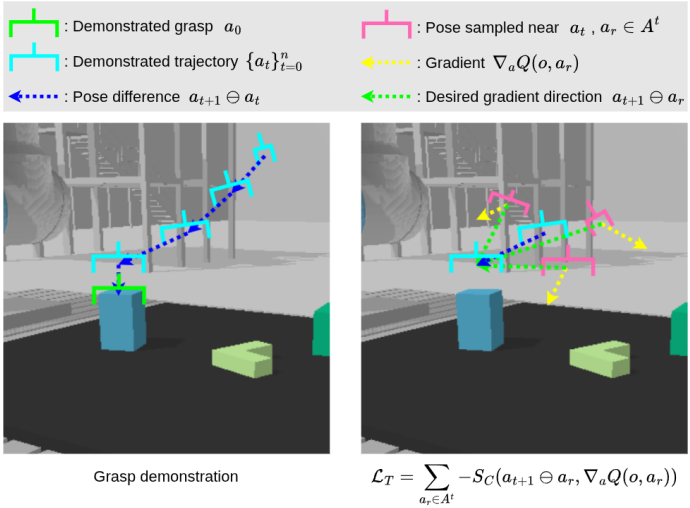


Figure 4.1: Trajectory Loss Illustration. A grasp demonstration provides a sequence of TCP poses (cyan) ending in the final grasp (green). The trajectory loss aligns the gradient (yellow arrows) at nearby poses (pink) to return to the demonstrated TCP motion (green arrows).

where $\mathcal{L}_P$ (pose loss) encourages high scores at demonstrated successful grasp poses, and $\mathcal{L}_T$ (trajectory loss) aligns the gradient landscape with proven TCP trajectories, enabling more effective policy optimization.

4.1.2 Architectural Implications

Given an observation o and candidate pose a, the trajectory loss depends explicitly on the gradient $\nabla_a Q(o, a)$, which must itself be differentiated during backpropagation to update the readout network's weights. Since the pretrained NeRF f_θ remains frozen, the grasp value function

$$Q(o, a) = Q_\psi\left(\{\hat{f}_\theta(p, o)\}_{s \in \mathcal{F}(a)}\right) \tag{4.5}$$

contains trainable parameters only in the readout network Q_ψ. $\mathcal{F}(a)$ denotes the SPD of a. By the chain rule,

$$\nabla_a Q(o,a) = \frac{\partial Q}{\partial a} = \sum_{s \in \mathcal{F}(a)} \frac{\partial Q}{\partial \hat{f}_\theta} \frac{\partial \hat{f}_\theta}{\partial s} \frac{\partial s}{\partial a} \,. \tag{4.6}$$

Considering that neither $\frac{\partial f_\theta}{\partial s}$ nor $\frac{\partial s}{\partial a}$ depend on the readout weights ψ, the gradient of $\mathcal{L}_T$ with respect to ψ – required for the weight update – is given by

$$\begin{aligned} \frac{\partial \mathcal{L}_T}{\partial \psi} &= \frac{\partial \mathcal{L}_T}{\partial \frac{\partial Q}{\partial a}} \frac{\partial \sum\limits_{s \in \mathcal{F}(a)} \frac{\partial Q}{\partial \hat{f}_\theta} \frac{\partial \hat{f}_\theta}{\partial s} \frac{\partial s}{\partial a}}{\partial \psi} \\ &= \frac{\partial \mathcal{L}_T}{\partial \frac{\partial Q}{\partial a}} \sum_{s \in \mathcal{F}(a)} \frac{\partial \frac{\partial Q}{\partial \hat{f}_\theta} \frac{\partial \hat{f}_\theta}{\partial s} \frac{\partial s}{\partial a}}{\partial \psi} \\ &= \frac{\partial \mathcal{L}_T}{\partial \frac{\partial Q}{\partial a}} \sum_{s \in \mathcal{F}(a)} \frac{\partial Q^2}{\partial \hat{f}_\theta \partial \psi} \frac{\partial \hat{f}_\theta}{\partial s} \frac{\partial s}{\partial a} \end{aligned} \tag{4.7}$$

The expression shows that mixed partial derivatives of Q_ψ with respect to its inputs and weights must be computed. While the pose-only loss does not require second-order gradients, the trajectory loss necessitates this architectural consideration. In the original paper that Chapter 3 is based on (Sóti et al. (2024)), ReLU activations were used for the pose-only models. However, for this dissertation, the models in Chapter 3 were already updated to use ELU activations in anticipation of the trajectory supervision extension presented in this chapter. All nonlinearities in Q_ψ are chosen to be ELU to ensure stable higher-order gradients and enable fair comparison between pose-only and trajectory-informed models. ELU was selected because it maintains similar activation patterns to ReLU, while providing continuous derivatives everywhere without the discontinuity at zero that characterizes ReLU. This continuity avoids instabilities in the required second-order derivatives for trajectory supervision. The frozen NeRF f_θ need only be differentiable in its input pose, so it can be reused without modification.

With this setup, the pose and trajectory losses backpropagate cleanly through $\mathcal{F}(a) \rightarrow \hat{f}_\theta \rightarrow Q_\psi$, enabling end-to-end updates of the grasp value readout and a gradient field aligned to demonstrated TCP motions.

4.1.3 Policy Hyperparameter Tuning

The implicit grasp policy itself remains unchanged; only its inference-time hyperparameters are optimized. During training, only the grasp value model is learned, while the policy's optimization procedure uses fixed hyperparameters. However, the policy's gradient ascent performance depends on these inference-time settings. Inference proceeds by parallel gradient ascent in $SE(3)$ using two Adam optimizers – one for translation and one for rotation – each governed by an initial learning rate and an exponential decay. The tunable parameters also include the total number of optimization steps.

Since these hyperparameters are not optimized during model training but significantly affect policy performance, a separate tuning procedure is needed to find optimal settings for each trained model. Bayesian optimization is employed to search this five-dimensional hyperparameter space efficiently. During the optimization, a Gaussian process surrogate is fit to the success-rate evaluations, and an acquisition function (e.g., Expected Improvement) balances exploration of uncertain regions against exploitation of promising settings. This approach requires far fewer evaluations than grid or random search, making it well suited for costly policy rollouts.

To tune the policy's hyperparameters, the search is performed over 60 trials on a small held-out *prism-simple* dataset containing 3 scenes with 12 objects overall. Each trial is run 6 times and the average success rate is recorded. The objective is to maximize the strict success rate of the policy. The search space is defined in Table 4.1

The tuned hyperparameters are those that achieve the highest strict-success rate. The number of initial grasp candidates remains fixed at 8000.

Table 4.1: Bayesian optimization search space for policy hyperparameters: number of optimization steps, and initial learning (lr) and exponential decay rates (γ) for both, translation and rotation policy optimizers.

Hyperparameter	Range	Distribution
num steps	8 - 32	integer uniform
initial lr_t	0.00 - 0.40	uniform
γ_t	0.00 - 1.00	uniform
initial lr_r	0.00 - 1.57	uniform
γ_r	0.00 - 1.00	uniform

4.2 Experiments and Results

Two dGrasp variants (mVNeRF dGrasp 1v and mVNeRF dGrasp 3v) are trained on the *prism-simple* demonstration set from Chapter 3, augmented with trajectory supervision. Demonstration trajectories begin at a random pose in the upper half of the workspace, proceed through a perturbed pre-grasp pose, and terminate at the final grasp pose. The TCP-movement is recorded at every 30 simulation steps (8 Hz) and the final pose is held for four additional timesteps.

Training follows the protocol of Chapter 3: the pretrained mVNeRF backbone remains frozen, and only the grasp value readout is updated for 3200 epochs with batch size 16 and learning rate 1×10^{-4}. At each step, a batch of 16 scenes provides the required camera measurements and one grasp demonstration (pose + trajectory). The pose loss is computed exactly as before. For the trajectory loss, a partial trajectory containing 6 poses is selected at random. For each of these 6 poses 32 neighbouring poses are drawn to form the supervision set (192 points per demonstration). Pose loss and trajectory loss are weighted equally.

Once both dGrasp grasp value models are trained, all six policy variants (base grasp 1v, base grasp 3v, mVNeRF grasp 1v, mVNeRF grasp 3v, mVNeRF dGrasp 1v, mVNeRF dGrasp 3v) are tuned, and evaluation proceeds exactly as in Section 3.3.3. Each policy is tested on the four simulated scenarios (*prism-simple*, *YCB-simple*, *prism-clutter*, *YCB-clutter*), 20 scenes each, with three fixed source

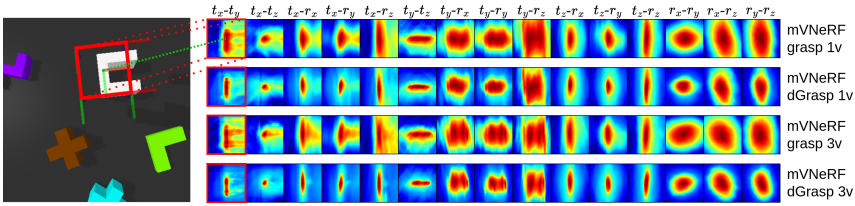


Figure 4.2: Grasp values landscapes around a successful grasp pose. The grasp values of the mVNeRF grasp and dGrasp models visualized in a neighborhood of a successful grasp pose. The patches correspond to perturbations of the grasp pose along the translational and rotational axes with the successful grasp pose in the center. Warm colors indicate high grasp values, while cool colors indicate low values.

views. 1-view policies fuse three images by summing per-view grasp values, like before (Eq. 3.28). Metrics are strict success rate (one attempt per object) and object-lifted rate with recovery (up to two attempts per object). In the real world, each policy attempts ten grasps per object on 14 items, the success rate is reported.

Tuning results appear in Section 4.2.2, simulation results in Section 4.2.3, and real-world results in Section 4.2.4.

4.2.1 Grasp Value Model – dGrasp

A qualitative assessment of the grasp values is provided the same way as in Section 3.3.2. The grasp values of the mVNeRF dGrasp 1v and mVNeRF dGrasp 3v models are visualized in a neighborhood of a successful grasp pose (Figure 4.2). The patches correspond to perturbations of the grasp pose along the translational and rotational axes with the successful grasp pose in the center. Warm colors indicate high grasp values, while cool colors indicate low values. For easier comparison, the mVNeRF grasp 1v and mVNeRF grasp 3v grasp values from the previous chapter are also included.

The dGrasp models show more pronounced peaks and an overall smoother landscape when compared to the mVNeRF grasp models. There are fewer local extrema. This smoother topology with sharper peaks around successful grasps creates more favorable conditions for gradient-based optimization: the pronounced

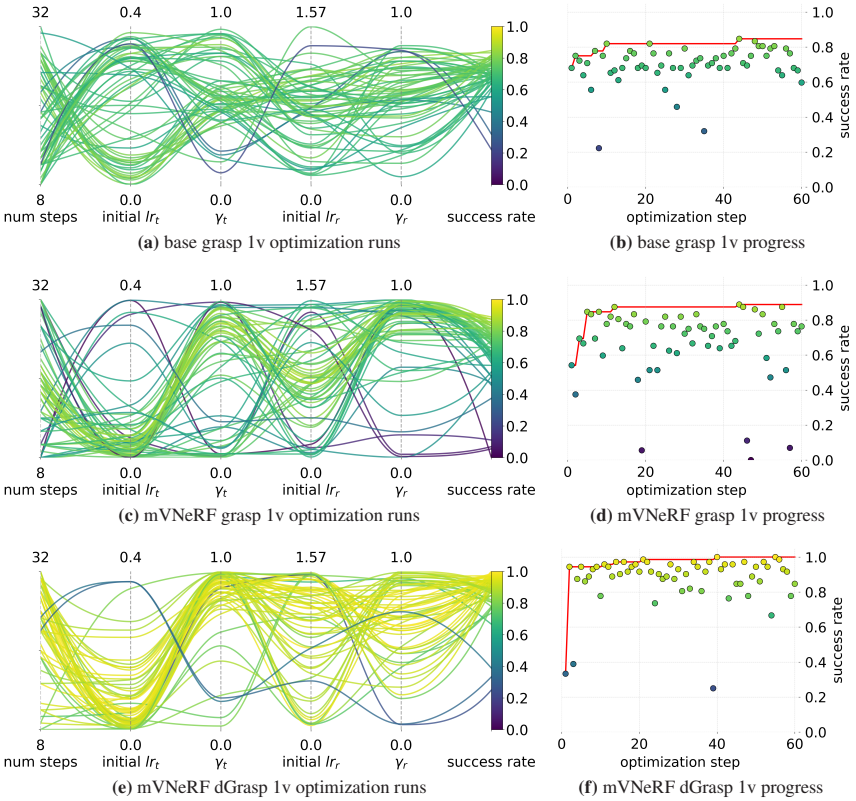
peaks provide stronger gradient signals to guide optimization toward successful poses, while the reduced number of local extrema decreases the likelihood of premature convergence to suboptimal solutions. The smoother landscape also ensures more consistent gradient flow, enabling the implicit policy's gradient ascent to navigate more reliably toward high-value regions.

The artefacts in the translational y-axis (t_y) remain similar: the 3-view model fragments the high-value region, whereas the 1-view model maintains a more uniform, consistently elevated band. The most improvement due to the trajectory loss can be observed in the translational x-axis (t_x), where the mVNeRF models show high value artefacts when moving between the U-shape's walls, while the dGrasp models value only the actual successful poses high and suppress the artefacts.

4.2.2 Policy Tuning Results

Figures 4.3 and 4.4 show the results of the hyperparameter search for the 1-view and 3-view policies, respectively. The search was conducted on a small held-out *prism-simple* validation set (3 scenes, 12 objects) to identify optimal hyperparameter configurations before evaluation on the main test datasets. The left column shows a parallel-coordinates plot of the 60 trials depicting the 5-dimensional hyperparameter space consisting of the number of optimization steps (num steps), the initial translational learning rate (lr_t), the translational decay (γ_t), the initial rotational learning rate (lr_r), and the rotational decay (γ_r), with color encoded average strict-success rate. The right column shows the optimization progress over the 60 trials.

> **Note:** The absolute success rates reported in this section are specific to the small validation set and serve to illustrate hyperparameter sensitivity patterns rather than overall model performance, which is evaluated in Sections 4.2.3 and 4.2.4.



(a) base grasp 1v optimization runs

(b) base grasp 1v progress

(c) mVNeRF grasp 1v optimization runs

(d) mVNeRF grasp 1v progress

(e) mVNeRF dGrasp 1v optimization runs

(f) mVNeRF dGrasp 1v progress

Figure 4.3: Hyperparameter search for 1-view policies: the left column shows the hyperparamter combinations and the achieved success rate in a parallel coordinate plot. The right column shows the progress of the success rate over the optimization steps.

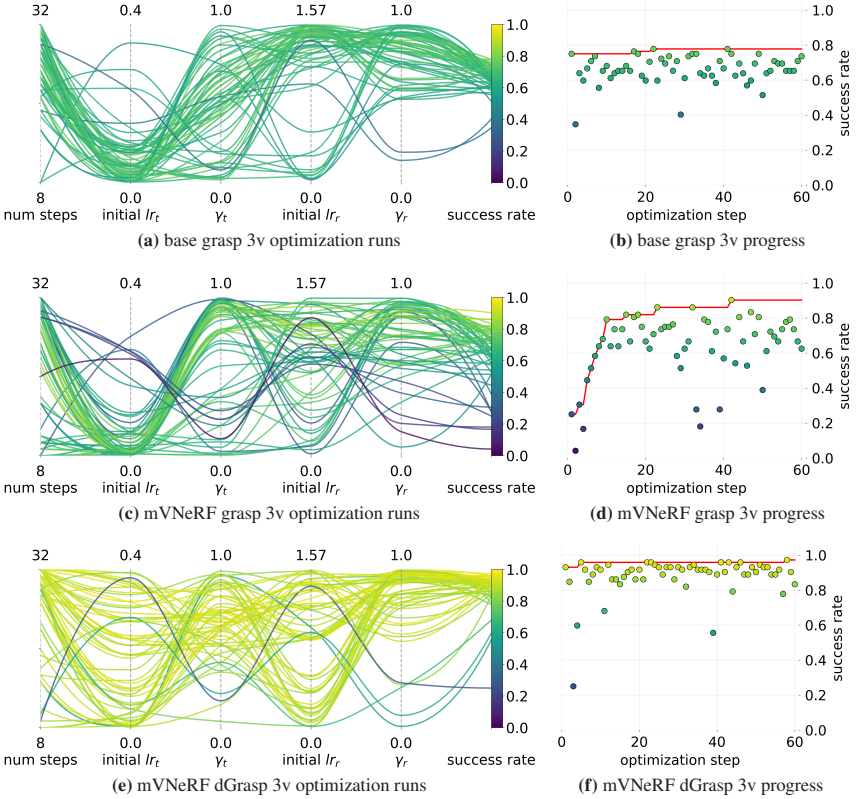
The **base grasp 1v policy** shows broad tolerance to hyperparameter settings, with successful configurations spanning nearly the entire search space (Figure 4.3a). Good performance occurs across the full range of 8-32 optimization steps, indicating no strong dependence on iteration count. Two clusters emerge for the initial translational learning rate: a low band in $[0.00, 0.08]$ and a high band in $[0.30, 0.40]$. Most top-performing runs use relatively permissive ranges: translational decay in $[0.40, 1.0]$, initial rotational learning rate in $[0.60, 1.20]$, and

rotational decay above $[0.60, 1.00]$. The optimization shows steady, gradual improvement from initial trials to peak performance, reaching maximum validation success of 0.85 at trial 44 (Figure 4.3b).

The **mVNeRF grasp 1v policy** concentrates its best configurations in a narrower region of the hyperparameter space, indicating increased sensitivity to parameter choices (Figure 4.3c). Optimal configurations predominantly use 15-22 optimization steps and require more precise settings: initial translational learning rate in $[0.00, 0.06]$, translational decay in $[0.75, 1.0]$, and rotational decay in $[0.85, 1.0]$. The initial rotational learning rate clusters around $[0.70, 0.90]$ but shows more scatter than other parameters. The optimization exhibits rapid early convergence, jumping from 0.54 to approximately 0.85 by trial 5, then showing incremental improvements to peak at 0.89 (Figure 4.3d).

The **mVNeRF dGrasp 1v policy** demonstrates exceptional robustness across most hyperparameter choices while achieving the highest validation performance (Figure 4.3e). Success spans the full range of 8-32 iterations with only slight preference for larger step counts. Initial learning rates for both translation and rotation show effective performance across nearly the entire search domain, indicating reduced sensitivity to these parameters. Decay factors concentrate in moderate bands—translational decay in $[0.70, 0.90]$ and rotational decay in $[0.60, 1.00]$—but remain fairly permissive. Remarkably, the optimization starts from an already high baseline of 0.91 at trial 1, quickly reaches 0.98 by trial 4, and achieves perfect validation performance (1.00) by trial 43, with almost no low-performing outliers (Figure 4.3f).

The **base grasp 3v policy** shows tighter hyperparameter preferences than its 1-view counterpart, indicating increased sensitivity (Figure 4.4a). Successful trials concentrate at higher step counts (21-32 optimization steps) and require specific parameter ranges: initial translation learning rate in $[0.00, 0.08]$, translation decay in $[0.40, 0.90]$, initial rotation learning rates in $[1.40, 1.60]$, and rotation decay in $[0.80, 0.95]$. The optimization shows modest improvement from 0.75 at trial 1 to a maximum of 0.78 at trial 22, indicating limited optimization potential (Figure 4.4b).



(a) base grasp 3v optimization runs

(b) base grasp 3v progress

(c) mVNeRF grasp 3v optimization runs

(d) mVNeRF grasp 3v progress

(e) mVNeRF dGrasp 3v optimization runs

(f) mVNeRF dGrasp 3v progress

Figure 4.4: Hyperparameter search for 3-view policies: the left column shows the hyperparameter combinations and the achieved success rate in a parallel coordinate plot. The right column shows the progress of the success rate over the optimization steps.

The **mVNeRF grasp 3v policy** exhibits even narrower parameter requirements (Figure 4.4c). Optimal configurations use 21-32 steps with very tight constraints: initial translation learning rates in $[0.00, 0.02]$, translation decay in $[0.88, 1.00]$, initial rotation learning rates in $[1.00, 1.40]$, and rotation decay in $[0.70, 0.95]$. The optimization displays high variability, starting from a low 0.25 at trial 1, climbing to 0.79 by trial 10, and eventually peaking at 0.90 at trial 43 (Figure 4.4d).

The **mVNeRF dGrasp 3v policy** restores robustness while maintaining high validation performance (Figure 4.4e). Although it favors 21-32 steps, successful trials span the full 8-32 range. Most importantly, initial learning rates cover the complete search ranges ($[0.00 - 0.40]$ for translation, $[0.00 - 1.57]$ for rotation), indicating reduced sensitivity. Translation decay concentrates in $[0.45, 0.90]$ and rotation decay in $[0.40, 1.00]$. The optimization demonstrates remarkable stability, starting from an exceptionally high 0.93 at trial 1 and reaching 0.97 by trial 57, with most configurations achieving above 0.90 performance (Figure 4.4f).

The hyperparameter sensitivity analysis reveals distinct patterns across model variants and clear benefits of trajectory supervision. Comparing optimization progressions on the validation set:

Peak Performance: Base models reach moderate peaks (0.85 for 1v, 0.78 for 3v), mVNeRF variants achieve higher levels (0.89 for 1v, 0.90 for 3v), while dGrasp models attain the highest validation performance (1.00 for 1v, 0.97 for 3v).

Convergence Patterns: Base 1v shows gradual improvement, mVNeRF 1v exhibits rapid early convergence, and dGrasp 1v starts from high baseline performance. The 3-view pattern is similar but with greater variability for mVNeRF models.

Hyperparameter Sensitivity: Base grasp 1v shows broad tolerance, base grasp 3v exhibits increased sensitivity with tighter viable parameter bands, and mVNeRF variants further concentrate their optimal parameters around specific intervals. In contrast, dGrasp policies achieve both strong performance and enhanced robustness, simultaneously widening the spectrum of effective hyperparameter choices.

These findings suggest that trajectory-informed training not only enhances performance but fundamentally reshapes the optimization landscape: aligning the learned gradient field with demonstrated TCP motions broadens basins of attraction and reduces sensitivity to optimization hyperparameters, facilitating more reliable gradient-based policy optimization across diverse parameter settings.

Table 4.2 lists the final, selected parameters for each policy.

Table 4.2: Default and tuned inference hyperparameters (32 steps).

Policy	num steps	lr_t	γ_t	lr_r	γ_r
default	32	0.01	0.93	0.98	0.95
base grasp 1v	27	0.020	0.47	1.08	0.76
mVNeRF grasp 1v	21	0.020	0.86	0.77	0.92
mVNeRF dGrasp 1v	29	0.025	0.90	1.50	0.62
base grasp 3v	26	0.060	0.44	1.39	0.87
mVNeRF grasp 3v	27	0.0028	0.95	1.28	0.93
mVNeRF dGrasp 3v	31	0.267	0.682	1.284	0.985

These final settings form the basis for all subsequent simulation and real-world evaluations in Sections 4.2.3 and 4.2.4.

4.2.3 Simulation Results

Evaluation follows the protocol of Section 3.3.3: each policy is run with 8000 initial grasp candidates and two Adam optimizers (translation and rotation) using the tuned hyperparameters. Four simulated scenarios (*prism-simple*, *YCB-simple*, *prism-clutter*, *YCB-clutter*) are tested, with 20 scenes per scenario and three fixed camera views. Each experiment is repeated six times, and both the mean *strict success rate* (one attempt per object) and the mean *object-lifted rate with recovery* (up to two attempts per object) are reported.

Table 4.3 compares the strict success rates of all six tuned policies. These results are evaluated on the main test scenarios (different from the small validation set used for hyperparameter tuning). Hyperparameter tuning yields mixed results compared to the untuned baselines from Chapter 3. Only the mVNeRF grasp 1v shows slight improvements in the *prism-simple* (from 0.83 to 0.89) and *YCB-clutter* (from 0.47 to 0.53) scenarios. MVNeRF dGrasp 1v – despite its broad robustness during validation-set tuning – does not surpass mVNeRF grasp 1v on the main test scenarios. By contrast, the 3-view dGrasp policy achieves the

highest overall performance (mean 0.72), with particularly strong results of 0.94 in the *prism-simple* setting.

Table 4.3: Strict success rate (final model at 3200 epochs).

Policy (views)	***prism-simple***	***YCB-simple***	***prism-clutter***	***YCB-clutter***	**mean**
base grasp 1v	0.75	0.50	0.73	0.52	0.63
mVNeRF grasp 1v	0.89	**0.59**	0.76	**0.53**	0.69
mVNeRF dGrasp 1v	0.84	0.50	0.79	0.42	0.64
base grasp 3v	0.58	0.47	0.57	0.49	0.53
mVNeRF grasp 3v	0.79	0.54	0.72	0.48	0.63
mVNeRF dGrasp 3v	**0.94**	0.55	**0.88**	0.52	**0.72**

A similar pattern emerges for the object-lifted rate with recovery (Table 4.4). The tuned policies produce almost identical results to their untuned counterparts, with minor improvements only for the mVNeRF grasp 1v policy. The 1-view base and mVNeRF policies achieve the best mean object-lifted rates of 0.86, while mVNeRF dGrasp 1v performs the worst with 0.76. The mVNeRF dGrasp 3v is only slightly behind the best policies with a mean of 0.85, while it notably achieves perfect lifts (1.00) in both prism scenarios. These perfect object-lifted rates across 113 objects, achieved consistently across all 6 experimental runs, demonstrate strong reliability compared to other models that achieved excellent but more variable performance (95-98 %) on the same scenarios, suggesting that trajectory supervision enhances consistency on geometric shapes similar to the training distribution.

Considering the post-tuning simulation outcomes shows that the choice of tuning distribution affects policy generalization. The tuned hyperparameters for mVNeRF dGrasp 1v indicate overfitting to the *prism-simple* tuning set: its strict success and object-lifted rate with recovery on *prism-simple* falls below expectations even though promising during tuning. By contrast, mVNeRF dGrasp 3v performs best on the prism benchmarks – especially *prism-simple* – suggesting that multi-view

Table 4.4: Object-lifted rate with recovery (final model at 3200 epochs).

Policy (views)	*prism-simple*	*YCB-simple*	*prism-clutter*	*YCB-clutter*	mean
base grasp 1v	0.95	0.73	0.97	0.77	**0.86**
mVNeRF grasp 1v	0.98	**0.77**	0.98	0.69	**0.86**
mVNeRF dGrasp 1v	0.95	0.60	0.95	0.54	0.76
base grasp 3v	0.88	**0.77**	0.89	**0.77**	0.83
mVNeRF grasp 3v	0.97	**0.77**	0.96	0.70	0.85
mVNeRF dGrasp 3v	**1.00**	0.71	**1.00**	0.71	0.85

tuning has primed the policy for prism-like shapes rather than arbitrary geometry. Thus, while trajectory loss offers benefits in multi-view settings, careful matching of the tuning distribution to the deployment scenes is essential to avoid scene-specific overfitting.

4.2.4 Real-World Results

Real-world performance is measured on 14 distinct objects, each subjected to ten grasp attempts per policy (see Section 3.3.4 for setup). Figure 4.5 and Table 4.5 show the results for all six tuned policies.

Hyperparameter tuning of both the base and mVNeRF 1-view policies produces fairly balanced, object-specific shifts in performance. In each case, four objects see improved success rates, six remain unchanged, and four decline. Both variants gain on the blue cube – whose geometry matches the tuning dataset – and the power drill, thanks to fewer colliding grasps. The red block and tennis ball maintain their performance. Declines occur for the red cylinder and hammer in both policies; the base policy additionally misses the rubber duck and soap dispenser more often, while the mVNeRF variant sees slight drops on the dental floss and large LEGO tire. The mVNeRF grasp 1v policy also benefits from tuning on the soap dispenser, where the base model does not. Both the small LEGO tire

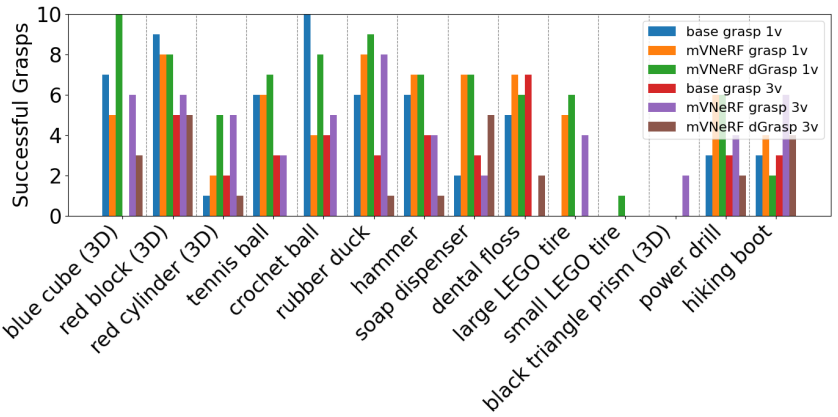


Figure 4.5: Grasp success bar chart for the real-world experiments.

and the black triangular prism remain ungrasped; however, the mVNeRF grasp 1v policy at least localizes both, failing only through near-misses or slippage.

In the 3-view setting, tuning has a larger effect on object-specific performance. The base policy achieves modest gains on four objects – rubber duck, soap dispenser, hiking boot, and the highest gain with dental floss ($+4$) – but suffers decreases on five items, including red block, red cylinder, tennis ball, crochet ball, and hammer (-5). By contrast, the tuned mVNeRF grasp 3v policy experiences the largest swings: it improves on seven objects, notably the red cylinder ($+4$) and crochet ball ($+4$), yet declines sharply on six, most significantly the blue cube (-5), dental floss (-6), and hammer (-5). It also grasps the black triangular prism twice – while none of the other models do – but entirely misses the dental floss despite the untuned policy's 6/10 success.

Both single-view base and mVNeRF policies remain more stable after tuning, with modest, balanced gains and losses compared to their multi-view counterparts. This aligns with the parallel-coordinates analysis: single-view policies inhabit broader basins of attraction, whereas their multi-view variants concentrate their optima – and pitfalls – into tighter regions, more aligned with the tuning data distribution.

Table 4.5: Detailed real-world grasp success: number of successes out of ten trials per object (best in bold); rightmost column shows overall success percentage.

Policy (views)	blue cube (3D)	red block (3D)	red cylinder (3D)	tennis ball	crochet ball	rubber duck	hammer	soap dispenser	dental floss	large LEGO tire	small LEGO tire	black tri. prism (3D)	power drill	hiking boot	Mean
base grasp 1v	7	**9**	1	6	**10**	6	6	2	5	0	0	0	3	3	41%
mVNeRF grasp 1v	5	8	2	6	4	8	7	**7**	**7**	5	0	0	6	4	49%
mVNeRF dGrasp 1v	**10**	8	**5**	**7**	8	**9**	7	**7**	6	**6**	**1**	0	**6**	2	**59%**
base grasp 3v	0	5	2	3	4	3	4	3	7	0	0	0	3	3	26%
mVNeRF grasp 3v	6	6	**5**	3	5	8	4	2	0	4	0	2	4	**6**	39%
mVNeRF dGrasp 3v	3	5	1	0	0	1	1	5	2	0	0	0	2	4	17%

Across the 14 real-world objects, the single-view mVNeRF dGrasp 1v policy achieves the highest mean success rate of 59%, even though it delivered the weakest results on the simulated tasks. This performance represents a notable achievement for zero-shot sim-to-real transfer without domain adaptation, intermediate perception steps, or simulation augmentation – conditions that are rare in the literature, as most approaches rely on such adaptations to bridge the reality gap. It outperforms the tuned mVNeRF grasp 1v on eight items, ties on four, and slightly underperforms on only two: dental floss and hiking boot. Like the tuned mVNeRF grasp 1v policy, it locates both the small LEGO tire and the black triangular prism, though it succeeds only once with the small LEGO tire. This demonstrates that trajectory supervision delivers broad gains for single-view policies, improving sim-to-real transfer on varied objects while preserving performance where mVNeRF grasp 1v was already strong.

By contrast, mVNeRF dGrasp 3v exceeds mVNeRF grasp 3v on only two objects (soap dispenser, dental floss), underperforms on eleven, and matches only on the small LEGO tire (which neither grasps at all). The classic failure mode – converging to the workspace edge – also reappears in the 3-view dGrasp policy. Although

its overall success rate drops, its successful grasps show tighter alignment to the objects when compared to most other policies.

The dominant failure cases remain unchanged from Section 3.3.4: convergence to the workspace edge, and failures through slippage or collision. A new error appears as $90°$ misalignment when grasping the hammer's shaft, whereas the base-1v error on dental floss, where the grasp is executed slightly above the object, no longer occurs.

As before, 3-view models perform worse overall than their 1-view counterparts. Trajectory loss clearly benefits the mVNeRF dGrasp 1v policy – producing the best real-world results – but further amplifies the performance gap, with mVNeRF dGrasp 3v the lowest success rate.

4.3 Summary

This chapter introduces a **trajectory loss** that augments the pose loss by supervising the gradient field of the grasp value model with recorded TCP trajectories, thereby embedding second-order information into training. In parallel, the implicit grasp policy's inference hyperparameters are optimized via **Bayesian search**.

In simulation, hyperparameter tuning alone yields only marginal improvements for 1-view policies, while the trajectory-informed mVNeRF dGrasp 3v model achieves the highest strict success rates across all scenarios. By contrast, real-world testing shows that the mVNeRF dGrasp 1v policy benefits substantially from trajectory supervision, achieving the highest **real-world success rate of 59%** – whereas mVNeRF dGrasp 3v's real-world success drops below its mVNeRF grasp 3v baseline.

These results indicate that trajectory supervision can reshape the optimization landscape to boost grasping performance in both simulation and reality. The smoother value landscapes with more pronounced peaks around successful grasps provide better gradient signals for the implicit policy's optimization process,

leading to improved convergence and generalization. However, the results also highlight that other factors can significantly influence performance, such as the alignment of the tuning distribution to the test data.

5 Complementary Extensions: Federated Learning and Language Conditioning

This chapter explores two complementary extensions that broaden the applicability of the NeRF-based implicit grasping framework.

The first extension addresses data distribution and privacy concerns through federated learning. In practical robotic deployments, different sites may have distinct environmental characteristics, object distributions, or privacy constraints that prevent centralized data collection. By incorporating federated learning into NeRF backbone training, this extension shows how multiple sites can collaboratively improve scene representations without sharing raw visual data, aggregating geometric understanding from diverse scenarios while maintaining data locality.

The second extension tackles the limitation of purely autonomous grasp selection by integrating semantic understanding. While the previous chapters demonstrate that NeRF-based policies can identify graspable objects, they lack the ability to selectively target specific objects based on user intent. This extension integrates language grounding through a pretrained CLIP encoder Radford et al. (2021), enabling the policy to accept natural-language prompts and bias grasp selection toward semantically specified objects. This bridges the gap between autonomous grasp planning and user-directed manipulation tasks.

Section 5.1 details the federated NeRF training procedure and its impact on grasp performance. Section 5.2 describes the CLIP-based conditioning architecture

and evaluates its effectiveness in enabling language-guided grasping. Finally, Section 5.3 briefly summarizes the chapter.

5.1 Federated NeRF Training

Federated learning is a decentralized training paradigm in which multiple clients collaboratively optimize a shared model by exchanging only parameter updates, rather than raw data. Each client holds its own local dataset – here, collections of RGB images and corresponding camera configurations from a particular simulated scenario – and performs training locally. Periodically, a central server aggregates these updates to produce a global model, which is then redistributed to each client for the next round. This mechanism preserves data locality, reduces bandwidth compared to transferring large image sets, and naturally accommodates heterogeneous scene distributions.

The motivation for federated NeRF training extends beyond data privacy considerations typical in federated learning literature. In robotic applications, different deployment sites may have distinct environmental characteristics, lighting conditions, or object distributions. Federated learning enables each site to contribute its local visual knowledge while building a shared representation that benefits from this diversity. This approach tests whether aggregating heterogeneous scene knowledge improves generalization to novel real-world conditions compared to training on homogeneous simulated data. Crucially, NeRFs require only calibrated camera poses as supervision, making them well-suited candidates for knowledge sharing between sites since ground truth data is readily available with calibrated cameras without requiring manual annotation.

This work applies federated learning to NeRF backbone training in order to study how splitting – or pooling – different simulated scenarios affects downstream implicit grasp policies. Section 5.1.1 describes the models' training configurations. Section 5.1.2 presents the experiment setup and results. Section 5.1.3 concludes with a brief summary.

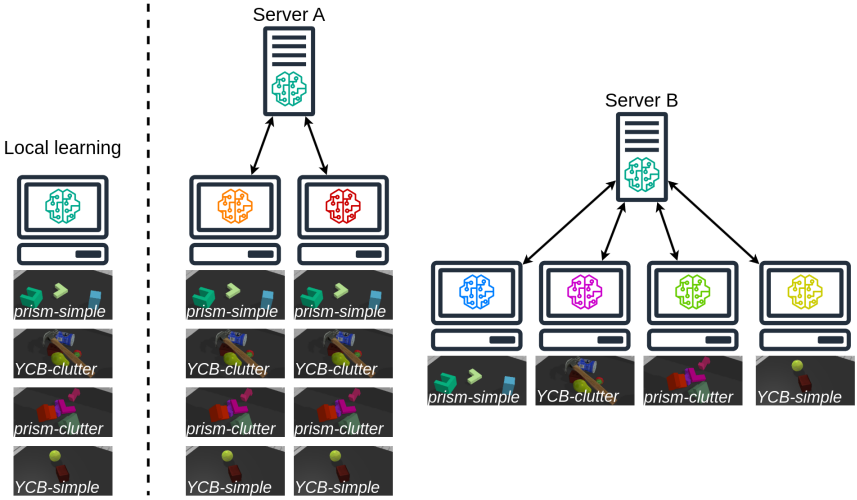


Figure 5.1: Federated Learning Setup: Local learning and both clients of Server A use data from all scenarios, while clients of Server B use data from distinct scenarios.

5.1.1 Federated Learning Configurations

For NeRF backbone training, 626 scenes are generated for each of the four simulated scenarios (2 504 scenes total), with 50 views per scene as in Chapter 3. Three 1-view mVNeRF models are trained:

- $\text{mVNeRF}_{\text{pool}}$ trains a single model locally on the combined dataset without federated learning.
- $\text{mVNeRF}_{\text{pool-fed}}$ is a federated setup, that duplicates the pooled dataset across two clients.
- $\text{mVNeRF}_{\text{fed}}$ assigns each of the four scenario datasets to its own client, with federated aggregation across all four.

Figure 5.1 illustrates these configurations.

In all configurations, model weights are initialized once on a central server and are broadcast to clients. Clients perform one epoch of local NeRF training (batch size

8, learning-rate schedule per Section 3.3.1) and return their updated parameters. The server averages these and configurations the new global weights. This repeats until 1600 global epochs are completed.

> **Note:** The $\text{mVNeRF}_{\text{pool}}$ and $\text{mVNeRF}_{\text{pool-fed}}$ runs each execute roughly 500k gradient-update steps per model, whereas in the $\text{mVNeRF}_{\text{fed}}$ setup each client performs about 125k steps on its own since its dataset is only a quarter of the total.

5.1.2 Experiments and Results

The models are evaluated on novel view synthesis and grasping. The experiments follow the protocol of the previous chapters, evaluating the mVNeRF models for novel view synthesis on the same dataset of all four simulated scenarios introduced in Section 3.3.1. The results are presented in Section 5.1.2.1. For grasping, a new 1-view grasp value model is trained using each mVNeRF backbone as a frozen feature extractor, with the same *prism-simple* dataset and identical hyperparameters from Chapter 3, employing both pose and trajectory loss. These grasp value models are then embedded in implicit policies with the tuned mVNeRF dGrasp 1v settings and evaluated in simulation (Section 5.1.2.2) and in the real world (Section 5.1.2.3).

5.1.2.1 Novel View Synthesis Results

To gauge how a more diverse training dataset, and federated learning, affects novel view synthesis with the mVNeRF architecture, first a qualitative assessment of the rendering quality is given, followed by a quantitative evaluation with the PSNR, SSIM, and LPIPS metrics.

Renderings from the *pooled* model (Figure 5.2) exhibit softer object edges in both RGB and depth compared to the original 1-view mVNeRF trained only on the *prism-simple* scenario. The pooled variant also produces fewer shadow artifacts

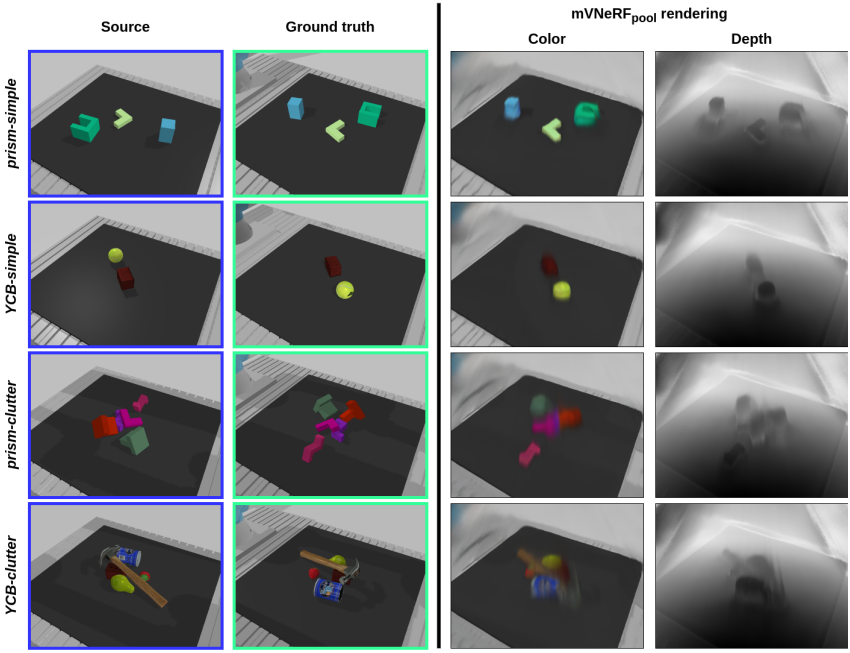


Figure 5.2: mVNeRF$_{pool}$ rendering results. The blue-bordered image serves as the source input to the NeRF along with its corresponding camera configuration (extrinsics and intrinsics). The model is used to render the scene from a different target camera configuration. The ground-truth image for that target configuration is highlighted with a green border. On the right are the model's renderings: RGB and depth.

in the depth maps and renders round objects – like the tennis ball – more faithfully, avoiding the cube-like distortion seen previously. Nevertheless, its overall fidelity declines: features on both prism and YCB objects appear blurrier than before.

The *pool-fed* model (Figure 5.3) similarly softens edges but captures round shapes even more accurately than the pooled baseline however, the shadows in the depth image are more pronounced. Notably, the magenta L-shape and hammer's shaft appear with better geometric fidelity than in both the original and pooled models, where both displayed an angular distortion (laying flat on the table). This suggests that federated aggregation can enhance certain geometric aspects, though overall sharpness remains reduced.

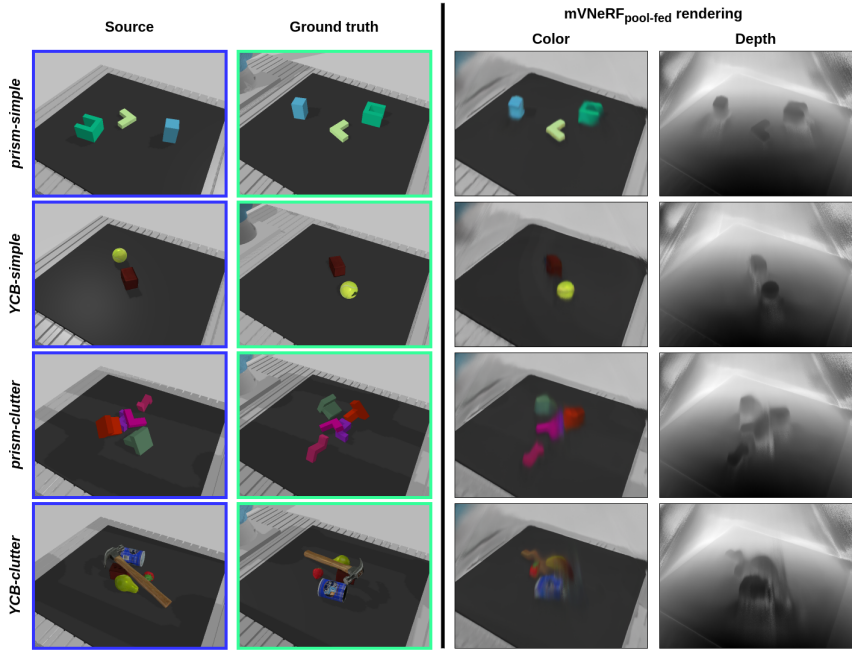


Figure 5.3: mVNeRF$_{\text{pool-fed}}$ rendering results. The blue-bordered image serves as the source input to the NeRF along with its corresponding camera configuration (extrinsics and intrinsics). The model is used to render the scene from a different target camera configuration. The ground-truth image for that target configuration is highlighted with a green border. On the right are the model's renderings: RGB and depth.

Among the models, the mVNeRF$_{\text{fed}}$ model yields the least favorable rendering results (Figure 5.4). While its renderings are noticeably blurrier than those of the other models, the overall geometry remains recognizable, demonstrating that the federated learning concept is indeed applicable to the mVNeRF backbone. The reduced rendering quality can be attributed to the training setup: although all models were trained for the same number of epochs, the mVNeRF$_{\text{fed}}$ model's training data was distributed among four clients. This division effectively reduces the amount of data each individual client-model processed.

Table 5.1 reports the peak signal-to-noise ratio (PSNR) for each federated training configuration. The mVNeRF$_{\text{pool}}$ model achieves the highest mean PSNR of 26.80

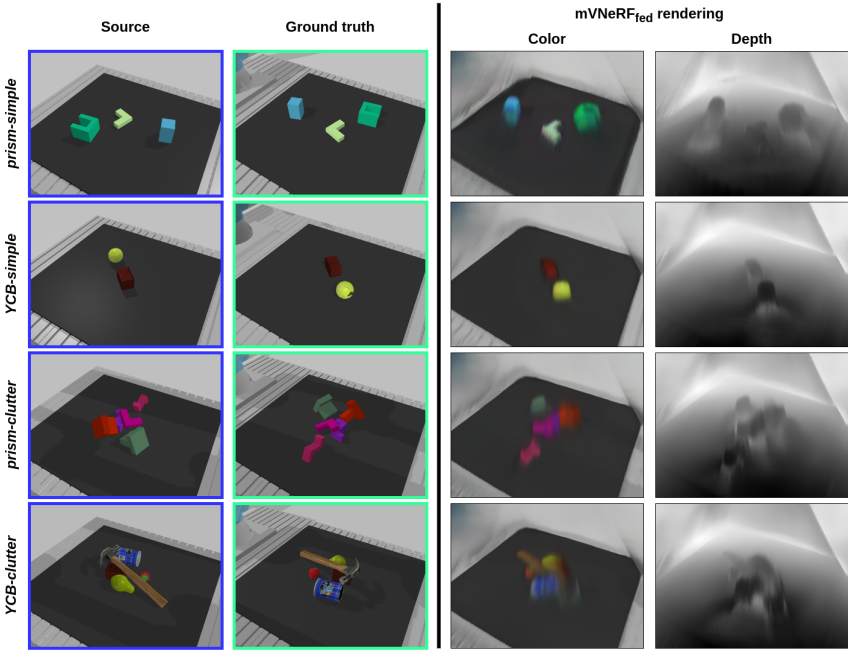


Figure 5.4: mVNeRF$_{fed}$ rendering results. The blue-bordered image serves as the source input to the NeRF along with its corresponding camera configuration (extrinsics and intrinsics). The model is used to render the scene from a different target camera configuration. The ground-truth image for that target configuration is highlighted with a green border. On the right are the model's renderings: RGB and depth.

dB, exceeding the original *prism-simple* 1-view mVNeRF (25.89 dB) and demonstrating the benefit of a more diverse pretraining corpus. The mVNeRF$_{\text{pool-fed}}$ variant follows closely at 26.58 dB and in fact outperforms all other 1-view variants on the *prism-simple* scenario (27.46 dB vs. 27.02 dB for the original) even thoug the original was trained solely on *prism-simple* data. In contrast, the fully partitioned mVNeRF$_{\text{fed}}$ model attains the lowest mean PSNR of 25.83 dB, reflecting its reduced per-client data budget. As an overall tendency, the mVNeRF$_{\text{pool}}$ model excels on the YCB scenes, while the mVNeRF$_{\text{pool-fed}}$ model retains stronger fidelity on the prism benchmarks.

Table 5.1: Federated Learning mVNeRF PSNR results (higher is better). Best values per scenario are in bold.

Model Variant	*prism-simple*	*YCB-simple*	*prism-clutter*	*YCB-clutter*	mean
$\text{mVNeRF}_{\text{pool}}$	26.30	**28.13**	25.86	**26.93**	**26.80**
$\text{mVNeRF}_{\text{pool-fed}}$	**27.46**	26.70	**26.73**	25.41	26.58
$\text{mVNeRF}_{\text{fed}}$	25.60	26.75	25.26	25.73	25.83

Table 5.2: Federated Learning mVNeRF SSIM results (higher is better). Best values per scenario are in bold.

Model Variant	*prism-simple*	*YCB-simple*	*prism-clutter*	*YCB-clutter*	mean
$\text{mVNeRF}_{\text{pool}}$	0.907	**0.936**	0.908	**0.919**	**0.918**
$\text{mVNeRF}_{\text{pool-fed}}$	**0.929**	0.912	**0.922**	0.885	0.912
$\text{mVNeRF}_{\text{fed}}$	0.898	0.915	0.896	0.897	0.901

Table 5.2 shows the structural similarity index (SSIM). Again, the $\text{mVNeRF}_{\text{pool}}$ model attains the highest overall SSIM (0.918), driven by top performance on both YCB scenarios (0.936 and 0.919). The $\text{mVNeRF}_{\text{pool-fed}}$ model achieves a close second mean of 0.912 and leads on the *prism-simple* and *prism-simple* scenarios, while the fully federated $\text{mVNeRF}_{\text{fed}}$ model records the lowest mean SSIM (0.901). These results mirror the PSNR trends: pooling all data yields the most consistent structural fidelity, whereas splitting unchanged pooled data across two clients scores higher across the prism scenarios, and dividing data across four clients degrades global coherence.

Finally, Table 5.3 reports the perceptual LPIPS metric (lower is better). The $\text{mVNeRF}_{\text{pool-fed}}$ model achieves the best mean LPIPS of 0.177, improving slightly over both the pooled (0.180) and original 1-view mVNeRF (0.180). It also records the lowest distortion on the *prism-simple* and *prism-simple* scenarios. The fully pooled $\text{mVNeRF}_{\text{pool}}$ model performs best on YCB scenes (0.149 and 0.176), while the fully federated $\text{mVNeRF}_{\text{fed}}$ model again lags behind with a mean LPIPS of

Table 5.3: Federated Learning mVNeRF LPIPS results (lower is better). Best values per scenario are in bold.

Model Variant	*prism-simple*	*YCB-simple*	*prism-clutter*	*YCB-clutter*	mean
$\text{mVNeRF}_{\text{pool}}$	0.201	**0.149**	0.194	**0.176**	0.180
$\text{mVNeRF}_{\text{pool-fed}}$	**0.162**	0.171	**0.166**	0.208	**0.177**
$\text{mVNeRF}_{\text{fed}}$	0.224	0.193	0.219	0.219	0.214

0.214. In summary, LPIPS corroborates the same pattern: pooling all data enhances general perceptual quality, federated pooling refines prism reconstructions, and extreme splitting reduces fidelity across the board.

These results show that training on a broader mix of scenes yields some gains in geometric fidelity, at the cost of sharpness. Federated learning also shows promise for the mVNeRF backbone, achieving comparable and sometimes better results than the pooled model.

5.1.2.2 Grasp Results – Simulation

Grasp performance is reported with the same *strict success rate* and *object-lifted rate with recovery* metrics used throughout this thesis, evaluated on the set of 20 scenes per simulated scenario used in the previous chapters.

In the *strict success rate* metric (Table 5.4), the policy with the $\text{mVNeRF}_{\text{pool}}$ backbone attains a mean of 0.62, only two points below the mVNeRF dGrasp 1v baseline of 0.64. The two-client split $\text{mVNeRF}_{\text{pool-fed}}$ based dGrasp policy even matches the mVNeRF dGrasp 1v on the *prism-simple* scenario (0.84 vs. 0.84) but, like the fully partitioned split $\text{mVNeRF}_{\text{fed}}$ dGrasp policy, suffers larger drops on the YCB scenes. This suggests that diversifying the NeRF pretraining data need not degrade performance on the grasp-training – and can sometimes preserve it entirely – yet partitioning that data across clients reduces overall *strict success rate* by a small margin.

Table 5.4: Federated Learning dGrasp-1v simulation results (strict success rate).

Policy	*prism-simple*	*YCB-simple*	*prism-clutter*	*YCB-clutter*	mean
$\text{mVNeRF}_{\text{pool}}$ dGrasp	0.82	**0.46**	**0.77**	**0.43**	**0.62**
$\text{mVNeRF}_{\text{pool-fed}}$ dGrasp	**0.84**	0.41	0.74	**0.43**	0.60
$\text{mVNeRF}_{\text{fed}}$ dGrasp	0.82	0.43	0.75	0.41	0.60

For the *object-lifted rate with recovery* metric (Table 5.5), both the $\text{mVNeRF}_{\text{pool}}$ and the $\text{mVNeRF}_{\text{fed}}$ dGrasp policy variants achieve a mean of 0.74 (mVNeRF dGrasp 1v 0.76), while the two-client split, $\text{mVNeRF}_{\text{pool-fed}}$ dGrasp, trails at 0.73. Crucially, all federated backbones outperformed the mVNeRF dGrasp 1v baseline on the cluttered YCB scenario (e.g. $\text{mVNeRF}_{\text{pool}}$ dGrasp: 0.62 vs. 0.54), despite grasp training using only *prism-simple* demonstrations. This indicates that more diverse visual pretraining – even when grasp data remain homogeneous – can enhance performance when recovery is allowed.

Table 5.5: Federated Learning dGrasp-1v simulation results (object-lifted rate with recovery).

Policy	*prism-simple*	*YCB-simple*	*prism-clutter*	*YCB-clutter*	mean
$\text{mVNeRF}_{\text{pool}}$ dGrasp	0.90	0.49	0.94	**0.62**	**0.74**
$\text{mVNeRF}_{\text{pool-fed}}$ dGrasp	**0.95**	0.46	**0.95**	0.55	0.73
$\text{mVNeRF}_{\text{fed}}$ dGrasp	**0.95**	**0.52**	0.92	0.55	**0.74**

Together, these results confirm that federated NeRF training supports downstream implicit grasping with only modest losses and even occasional gains.

5.1.2.3 Grasp Results – Real-World

Real-world evaluation follows the protocol of Section 3.3.4: each policy attempts ten grasps on each of 14 diverse objects, and success is recorded as a lifted object per trial. Figure 5.5 visualizes these outcomes as a bar chart, while Table 5.6 provides the detailed per-object counts and mean success rates.

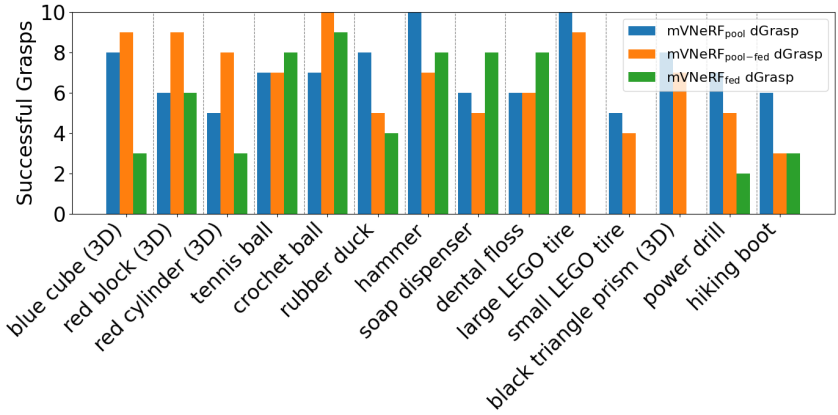


Figure 5.5: Grasp success bar chart for the federated learning experiments.

Table 5.6: Detailed real-world grasp success for federated learning policies: number of successes out of ten trials per object (best in bold). The rightmost column shows the overall success percentage.

Policy	blue cube (3D)	red block (3D)	red cylinder (3D)	tennis ball	crochet ball	rubber duck	hammer	soap dispenser	dental floss	large LEGO tire	small LEGO tire	black tri. prism (3D)	power drill	hiking boot	**Mean**
mVNeRF$_{pool}$ dGrasp	8	6	5	7	7	**8**	**10**	6	6	**10**	**5**	**8**	**7**	**6**	**71%**
mVNeRF$_{pool\text{-}fed}$ dGrasp	**9**	**9**	**8**	7	**10**	5	7	5	6	9	4	7	5	3	67%
mVNeRF$_{fed}$ dGrasp	3	6	3	**8**	9	4	8	**8**	**8**	0	0	0	2	3	44%

Both the mVNeRF$_{pool}$ and the mVNeRF$_{pool\text{-}fed}$ backbones yield appreciable gains in real-world grasping over the baseline mVNeRF dGrasp 1v policy (59 % mean). The mVNeRF$_{pool}$ backbone achieves a 71 % mean success rate, representing a 20% relative improvement. While formal statistical significance testing is challenging given the practical constraints of real-world robotics experiments (140 trials per policy across 14 diverse objects), the consistency of improvements

across multiple previously challenging objects suggests systematic rather than random performance gains: it grasps the large LEGO tire ($6 \rightarrow 10$), small LEGO tire ($1 \rightarrow 5$), black triangular prism ($0 \rightarrow 8$), hiking boot ($2 \rightarrow 6$), and hammer ($7 \rightarrow 10$) far more reliably. The mVNeRF$_{\text{pool-fed}}$ dGrasp variant follows at 67 %, also recovering many of these difficult cases (e.g. large LEGO tire $6 \rightarrow 9$, black prism $0 \rightarrow 7$, red cylinder $5 \rightarrow 8$) while trading off some performance on easier items like the rubber duck and soap dispenser. One salient change is the black triangular prism: exposure to a more heterogeneous, pooled dataset – with greater variation in object geometry (beyond right-angled prisms), viewpoints, textures, lighting, and clutter (including YCB scenes) – plausibly improved the backbone's ability to represent shapes outside the *prism-simple* distribution. The resulting broader visual prior provides the grasp value model with more reliable geometric cues around the prism, enabling stable contact proposals and aligning with the jump from 0 to 7-8 successes on this object.

In contrast, the fully federated mVNeRF$_{\text{fed}}$ dGrasp setup – where each client sees only one scenario – drops to 44 % mean success. Although it regains a handful of objects (e.g. dental floss $6 \rightarrow 8$, tennis ball $7 \rightarrow 8$, hammer $7 \rightarrow 8$), it catastrophically fails others (large and small LEGO tires both $6 \rightarrow 0$, black prism remains $0 \rightarrow 0$). These results suggest that exposing the NeRF backbone to a diverse, aggregated dataset – even via federated averaging – enhances the downstream grasp model's ability to generalize to varied, challenging real-world shapes. In contrast, extreme partitioning of that data limits the visual representation and harms grasp success in this setting.

5.1.3 Summary

This section investigated the application of federated learning to train mVNeRF backbones for implicit grasp policies. Three main configurations were tested: a centralized mVNeRF$_{\text{pool}}$ model trained on all four simulated scenarios, a mVNeRF$_{\text{pool-fed}}$ variant distributing the same pooled data across two clients, and a mVNeRF$_{\text{fed}}$ setup where four clients each trained on a distinct scenario.

Qualitatively and quantitatively, novel view synthesis results showed that both the $\text{mVNeRF}_{\text{pool}}$ and $\text{mVNeRF}_{\text{pool-fed}}$ models, benefiting from diverse data, with comparable performance to the original 1-view mVNeRF (trained only on *prism-simple* data), though with loss in sharpness. The $\text{mVNeRF}_{\text{fed}}$ model, with its partitioned data, showed the lowest rendering quality, highlighting the impact of reduced per-client data.

For downstream grasping, policies using the $\text{mVNeRF}_{\text{pool}}$ and $\text{mVNeRF}_{\text{pool-fed}}$ backbones demonstrated strong performance. In simulation, they nearly matched or sometimes exceeded the baseline mVNeRF dGrasp 1v policy (trained on *prism-simple* only). Most significantly, in real-world experiments, the $\text{mVNeRF}_{\text{pool}}$ and $\text{mVNeRF}_{\text{pool-fed}}$ backbones led to substantial grasping success rate improvements (71% and 67% respectively, up from 59%), especially on challenging objects. Conversely, the $\text{mVNeRF}_{\text{fed}}$ model with extreme data partitioning saw a considerable drop in real-world grasp performance (44%).

5.2 Vision-Language Conditioning via CLIP

The previous chapters optimize grasp candidates purely on geometric and photometric properties. The policy is indifferent to *which* object it ultimately lifts. To introduce semantic selectivity, this section augments the grasp value model with a *vision-language* pathway based on OpenAI's Contrastive Language-Image Pre-training (CLIP) encoder Radford et al. (2021). At test time the policy receives a free-form text prompt – e.g. "grasp the yellow block" or "pick up the blue L-shape" – in addition to calibrated camera images. The prompt is embedded by CLIP's text encoder, fused with NeRF-derived visual features, and propagated through a modified read-out network. Gradient ascent in $SE(3)$ then favours candidate poses whose fused representation aligns with the prompt, effectively focusing on the target object.

The remainder of the section is organised as follows: Section 5.2.1 introduces the multimodal architecture, Section 5.2.2 details the experiment setup, Section 5.2.3 reports the results, and finally Section 5.2.4 concludes with a summary.

5.2.1 Multimodal Architecture

CLIP itself is a multimodal model that can be used to embed text and images into a common embedding space. Its typical usage is to correlate these embeddings to find matching image-text pairs. Additionally, a pre-trained CLIP is often integrated into various other models to provide visual and linguistic grounding for other applications. There are two distinct CLIP image encoder implementations, a ResNet50 and a ViT-B/32. The ViT-B/32 is used in this work.

CLIP's image encoder returns a single global embedding vector of the whole image and therefore loses spatial information. To retain locality, the input image is cropped into 7×7 overlapping patches. The patch size is chosen to potentially contain whole objects while maintaining spatial resolution through overlap, enabling CLIP to provide semantic guidance about object locations rather than fine-grained spatial details. Each patch is encoded by the image encoder. The resulting 768-D tokens are re-assembled into a 7×7 grid and bilinearly up-sampled to full resolution (Figure 5.6). The operation delivers a dense tensor C that assigns a CLIP feature vector to every pixel.

Similar to the NeRF-based processing, CLIP spatial features are extracted from the image by projecting the TCP pose into the image and then extracting the CLIP feature vector at this pixel. In contrast to the NeRF-based processing, it is not the support poses obtained from SPD that are used. This should be sufficient, since the objective of the language conditioning is to roughly guide the TCP towards the correct object, while the finer positioning is handled by the geometric features from the NeRF. Also, it reduces the number of parameters to be trained.

The processing of the natural language prompt is straightforward. It is tokenized and embedded by CLIP's unaltered text encoder into a 768-D vector.

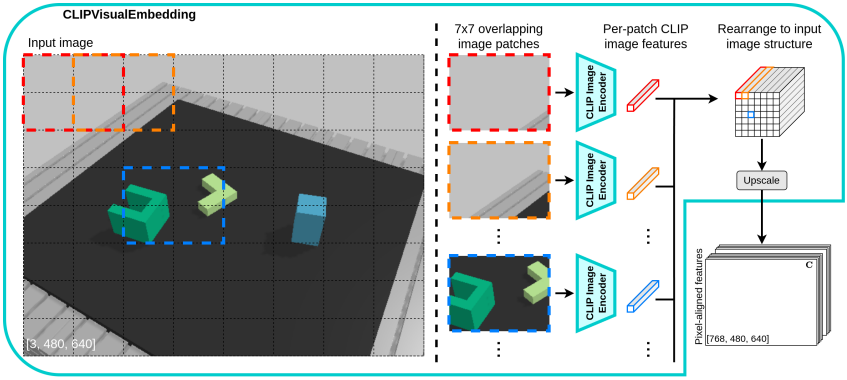


Figure 5.6: CLIP Visual Embedding. Each 640×480 RGB image is tiled into 7×7 overlapping patches. Every patch is passed through the frozen ViT-B/32 image encoder of CLIP, yielding a 768-D token. Tokens are re-arranged to form a coarse spatial grid and up-sampled to the original resolution, producing pixel-aligned CLIP features $C \in \mathbb{R}^{768 \times 480 \times 640}$.

Aligning with typical CLIP applications that measure image-text correlation, the text and image embeddings are combined via the Hadamard product (element-wise multiplication). This preserves more information than a scalar product while remaining computationally simple for the network to process, as the readout network can easily learn to aggregate the element-wise correlations if needed. With the unaltered embeddings and this combined embedding there are 3 additional 768-D vectors to integrate into the readout network.

The readout network combines these additional 768-D vectors and the NeRF features of all the support poses. First, the NeRF features are processed in the same way as in the non-CLIP grasp value readout via the support pose embedding module and all 3 CLIP vectors are projected down to 512-D. Then they are concatenated to the support pose embeddings and passed through 3 ResNet blocks to produce a scalar grasp value as shown in Figure 5.7.

with this, the grasp value function is defined as:

$$Q(o,a) = Q_\psi^{\text{CLIP}}\left(\left\{\hat{f}_\theta(s,o)\right\}_{s\in\mathcal{F}(a)}, \text{CLIP}_{image}(I), \text{CLIP}_{text}(p)\right) = q. \tag{5.1}$$

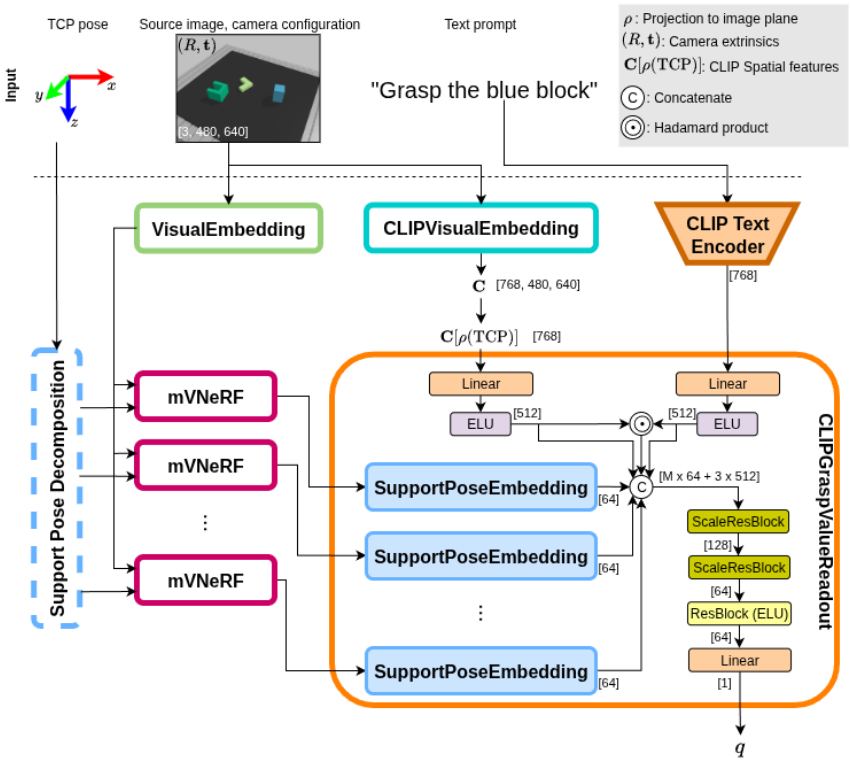


Figure 5.7: CLIP Grasp Value Readout. It fuses the mVNeRF and CLIP based information, including the prompt to produce the grasp value. The fist ScaleResBlock's parameters are $srb_i = M \times 64 + 3 \times 64$, $srb_h = 512$ and $srb_o = 128$ and the second ScaleResBlock's parameters are $srb_i = 128$, $srb_h = 128$ and $srb_o = 64$.

where o is the camera measurement, a is the grasp candidate, Q_ψ^{CLIP} is the CLIP grasp value readout network, $\hat{f}_\theta$ is the NeRF feature extractor (see Eq. 3.18), $\mathcal{F}(a)$ is the set of support poses for the grasp candidate a, I is the image from the camera measurement, p is the prompt, and q is the grasp value. CLIP_{image} and CLIP_{text} are the pretrained CLIP image and text encoders.

5.2.2 Experimental Setup

The CLIP grasp value model is trained on the *prism-simple* demonstrations (Section 3.3.2), using the same pose- and trajectory-loss objectives and hyperparameters. For every demonstrated grasp a *prompt* is auto-generated in the following pattern:

$$\text{verb article } \big[\text{color}\big] \ \big[\text{object name}\big].$$

Verbs are *grasp*, *pick up*, or *take*. The article can be *the*, *a* or *an*, and is only included for completeness of the sentence, it is removed during tokenization. An object name is included in 80 % of prompts. In the remaining cases the placeholder word *object* is used. When the object name is given, the color is specified in 80 % of the cases. When only the placeholder world *object* is used, the color is always specified. Each training data sample therefore contains a calibrated image, a demonstrated grasp pose, grasp trajectory and a corresponding prompt.

The training configuration is the same as in the previous chapters. Both pose and trajectory loss are used. For policy inference, the tuned hyperparameters from the mVNeRF dGrasp 1v model (Chapter 4) are employed, as this configuration most closely matches this CLIP based setup (single input view with both pose and trajectory supervision).

For evaluation, 60 scenes with two to five objects from the *prism-simple* scenario are placed in the robot's workspace and the observations are rendered from the three fixed viewpoints used throughout this work. For every scene the objects are queried in a random order. After each successful grasp and lift the object is removed and the next prompt is evaluated. Metrics record physical grasp success as well as whether the lifted object matches the prompted *color*, *shape*, and *both*. For computing the matches, grasp success is not considered. Instead, the object closest to the attempted grasp is compared to the prompted object. A *random* baseline that picks an object uniformly at random (and is always counted

as physically successful) is used to contextualise the language-matching scores. The experiment is repeated 6 times.

5.2.3 Results

Table 5.7 summarises the policy's accuracy averaged over all 6 runs and the random baseline. The policy lifts objects reliably (93%) and matches the prompt markedly better than chance: +14% for color, +22% for shape, +21% when both attributes must agree.

Table 5.7: Language-conditioned grasp results. "Success" denotes physical lifts. The remaining columns measure attribute matches irrespective of lift outcome.

	Success	**Color**	**Shape**	**Both**
CLIP policy	0.933	0.614	0.772	0.578
Random	-	0.472	0.547	0.372

Figure 5.8 plots per-run accuracy against the random baseline. Performance varies by < 8% and consistently outperforms the random baseline.

Confusion matrices in Figure 5.9 show the types of errors the policy makes. The diagonal shows the correct matches, the off-diagonal shows the errors. For shapes, the CLIP policy has a significantly more pronounced diagonal. The confusion matrix is not symmetric and there are no conspicuous errors, the most confused items are *block* $\rightarrow$ *cross*, *U-shape* $\rightarrow$ *L-shape* and *block ring* $\rightarrow$ *T-shape*. The color confusion matrix of the CLIP policy also shows more consistent results than the random baseline. Again, the results are not symmetrical. It struggles the most with the color *gray*. Interestingly, *yellow* and *orange* are matched with the highest rate, which are also the colors that are most often picked when another color is specified, showing an overall bias towards these bright colors. The reason for the random policy not having a more uniform distribution is that only a subset of the

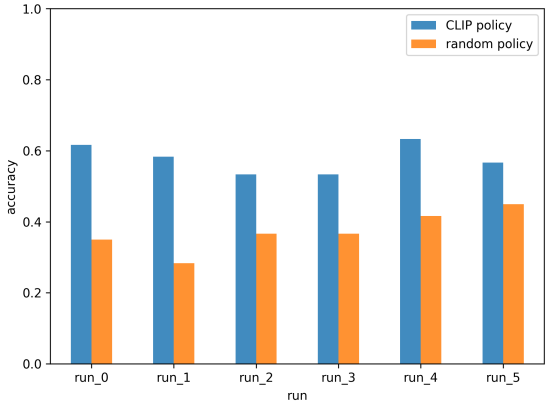


Figure 5.8: Per-run prompt-match accuracy. The blue line shows the accuracy of the CLIP-conditioned policy, the orange line shows the accuracy of the random baseline. Accuracy is measured as the percentage of correctly selected objects.

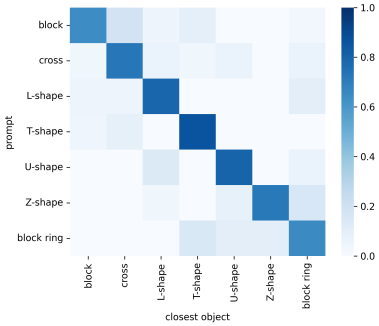
objects is present in each trial, but the object specified in the prompt is always present.

5.2.4 Summary

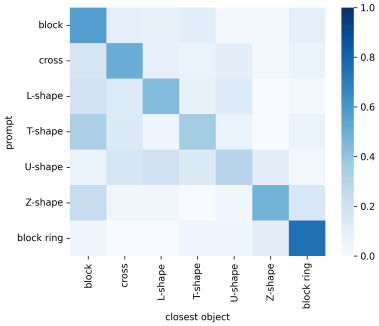
This section introduces vision-language conditioning to the NeRF-based grasp policy by integrating CLIP embeddings. The approach extends the grasp value model to accept natural language prompts specifying target objects (e.g., "grasp the yellow block"), enabling semantic selectivity while preserving the implicit policy's optimization in $SE(3)$.

The multimodal architecture extracts CLIP image and text features and fuses them with NeRF-derived representations in an extended grasp value readout network. The system is trained on auto-generated prompts for simple prismatic objects and evaluated on its ability to match prompted color and shape attributes.

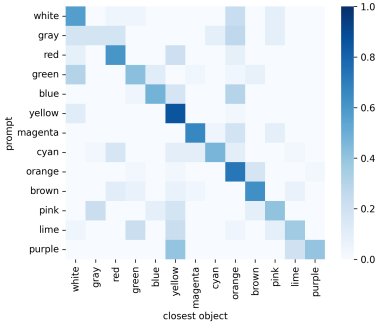
Experimental results demonstrate that the CLIP-conditioned policy achieves reliable physical grasping (93.3% success rate) while significantly outperforming



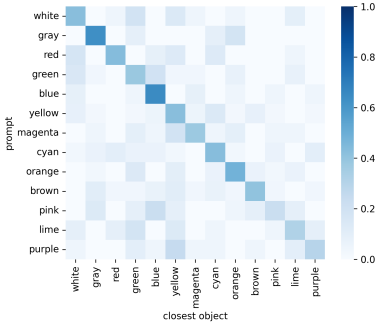
(a) Shape – CLIP policy



(b) Shape – random picker



(c) Color – CLIP policy



(d) Color – random picker

Figure 5.9: Confusion matrices for prompted attribute matching. Darker cells correspond to a higher match rate for the corresponding attributes.

random selection in attribute matching: +14.2% for color identification, +22.5% for shape recognition, and +20.6% when both attributes must agree. However, the 57.8% accuracy for correctly following complete language instructions ("both" attributes) falls well short of practical deployment requirements. For real-world applications, users would expect robots to correctly interpret and execute language commands. The current results demonstrate proof-of-concept for the architectural approach but highlight the need for substantial improvements through larger-scale training data, more sophisticated vision-language fusion mechanisms, or pre-training on more diverse object vocabularies.

5.3 Chapter Summary

This chapter explored two extensions that broaden the applicability of the NeRF-based implicit grasping framework: federated learning for distributed NeRF training and CLIP integration for language-conditioned grasping. Both extensions maintain the core architecture while addressing distinct deployment challenges.

The federated learning approach enables collaborative NeRF training across multiple robotic sites without data sharing, with the pooled federated model achieving improved real-world performance (71% vs 59% baseline). The CLIP integration adds vision-language conditioning for natural language object specification, maintaining reliable physical grasping (93.3%) while providing moderate semantic targeting capabilities (57.8% for complete attribute matching).

Both extensions preserve the framework's differentiable, continuous optimization properties and can be successfully combined, validating the modular design. The consistent performance improvements demonstrate the robustness and extensibility of NeRF-based representations for robotic grasping.

6 Related Work

This chapter surveys the research landscape relevant to the core contributions of this dissertation: NeRF-based implicit grasp policies, trajectory-informed learning, federated NeRF training, and vision-language conditioning for robotic manipulation. The discussion is organized to highlight how existing approaches relate to and differ from the methods presented in this work.

The chapter begins with an overview of robotic grasping approaches, establishing the broader context for learning-based manipulation. It then examines implicit and diffusion-based policy learning, including trajectory-informed methods that leverage full demonstration trajectories for improved optimization landscapes. The discussion continues with neural scene representations in robotics, covering NeRFs, image-conditioned variants, and federated training approaches. Finally, the chapter explores vision-language conditioning methods, including recent advances in combining language understanding with 3D scene representations, before positioning the contributions of this work within the broader research landscape.

6.1 Robotic Grasping Approaches

Data-driven policies in robotic grasping represent a widely researched area with diverse methodological approaches, as comprehensively detailed in surveys by Bohg et al. (2013), Kleeberger et al. (2020), Newbury et al. (2023). These methods can be broadly categorized into two main paradigms: traditional model-based approaches that rely on explicit object representations, and end-to-end learning methods that directly map perception to actions.

6.1.1 Model-Based Approaches

Traditional approaches to robotic grasping often couple object detection or segmentation with 6D pose estimation and grasp pose prediction (Zhu et al. (2014), Zeng et al. (2017), Xiang et al. (2017), Wang et al. (2019), Deng et al. (2020)). These methods typically require explicit object models and assume objects have nominal orientations, making them unsuitable for handling unknown objects. While effective for known objects in structured environments, their reliance on pre-defined models limits their applicability in unstructured settings.

Methods based on keypoint detection or dense descriptors can generalize to object categories and even learn to manipulate deformable objects (Florence et al. (2018), Manuelli et al. (2019), Kulkarni et al. (2019), Nagabandi et al. (2020), Liu et al. (2020)). However, these approaches often require substantial amounts of object-specific labeled data, making their universal application challenging. The need for extensive annotation limits their scalability to new object categories and environments.

6.1.2 End-to-End Learning Approaches

End-to-end learning models have shown superior performance in unstructured environments such as logistics and household settings, as they directly map perception to actions without intermediate representations (Gualtieri et al. (2018), Wu et al. (2019), Zakka et al. (2020), Hundt et al. (2020), Berscheid et al. (2020), Devin et al. (2020), Khansari et al. (2020), Song et al. (2020), Zeng et al. (2020)). These approaches can be further categorized based on their learning paradigm: learning from demonstration, reinforcement learning, and supervised learning from large-scale datasets.

6.1.2.1 Learning from Demonstration

Learning from demonstration approaches enable robots to acquire manipulation skills by observing expert demonstrations. A single recurrent visuomotor policy trained from raw images by Rahmatizadeh et al. (2018) can accomplish a suite of pick-and-place and other complex tasks on a low-cost robotic arm. Florence et al. (2019) introduced a pixel-wise correspondence objective that markedly improves the generalization and sample efficiency of visuomotor controllers, and Zhang et al. (2018b) demonstrated that minutes-long VR-teleoperated demonstrations suffice to train end-to-end RGB-D policies for complex manipulation tasks.

A particularly influential architecture in this category is the Transporter Network Zeng et al. (2020), which has gained popularity due to its straightforward design and sample efficiency. Transporter Networks address pick-and-place tasks through a two-module architecture: an attention module that predicts grasp positions and a transport module that determines placement poses by correlating visual features of the picked object with target locations.

The attention module maps observations to probability distributions over pick positions, while the transport module learns to correlate visual features of the picked object with visual features of the target pose. The architecture relies heavily on orthographic projections of point clouds to ensure spatial consistency, enabling extensive data augmentation. The transport module processes rotated copies of the picked object's features to handle different orientations, using convolution operations to compute correlations across the workspace.

Transporter Networks have inspired numerous extensions and modifications. CLIPort Shridhar et al. (2022) combines Transporter Networks with language-conditioned policies for task specification. Equivariant Transporter Networks Huang et al. (2022) incorporate E(2) steerable equivariant convolutional layers, making the architecture completely rotationally equivariant. This design ensures that input transformations result in corresponding output transformations, improving sample efficiency and reducing computational requirements for rotation

handling. Sóti et al. (2023b) further improves pick-and-place precision by introducing a training and inference method that enables iteratively refining rotation estimates.

The attention module of the Transporter Network has also greatly inspired the design of the mVNeRF-based grasp value model presented in this work. Conceptually, both behave as grasp value functions that score candidate grasps, however, whereas the Transporter attention discretizes the workspace into a 2D grid and restricts itself to top-down grasps, our model operates directly in the continuous $SE(3)$ space, allowing evaluation of arbitrary 6-DoF grasp poses.

6.1.2.2 Reinforcement Learning

Reinforcement learning methods for grasping have shown promise in learning complex manipulation behaviors through environmental interaction (Levine et al. (2018), Song et al. (2020), Berscheid et al. (2021)). These approaches can discover novel grasping strategies and adapt to changing environments. However, they typically require extensive data collection and environmental interactions, making them less practical for real-world deployment where sample efficiency is crucial.

6.1.2.3 Large-Scale Supervised Learning

Many recent supervised learning approaches leverage large-scale, labeled datasets to train grasping models (Mahler et al. (2019), Eppner et al. (2020), Chen et al. (2022)). These methods can achieve impressive performance when sufficient training data is available.

The Robotics Transformer (RT-1, Brohan et al. (2022)) represents a significant advancement in scaling robotic learning through transformer architectures. RT-1 demonstrates that transferring knowledge from large, diverse, task-agnostic datasets can enable robotic models to solve specific downstream tasks with high

performance. The approach emphasizes the importance of open-ended task-agnostic training combined with high-capacity architectures that can absorb diverse robotic data. Through extensive evaluation on real robots performing real-world tasks, RT-1 shows how generalization capabilities scale as a function of data size, model size, and data diversity, establishing promising scalable model properties for robotics.

6.2 Implicit and Diffusion-Based Policy Learning

The concept of implicit policies represents a significant departure from traditional explicit policy learning, where actions are directly predicted from observations. Instead, implicit policies learn to evaluate actions through energy or value functions, with optimal actions determined through optimization procedures. Recently, diffusion-based approaches have emerged as a complementary paradigm that learns to generate actions through iterative denoising processes. Table 6.1 provides an overview of the differences and similarities between the different policy families discussed in this section and this work.

6.2.1 Implicit Behavioral Cloning

Florence et al. (2022) provide a comprehensive investigation of implicit models for behavior cloning (IBC) across various robotic tasks. They define an implicit policy as the argmin of a continuous energy function learned from demonstrations:

$$\pi(o) = \underset{a \in \mathcal{A}}{\operatorname{argmin}} \, E(o, a) \tag{6.1}$$

where $E(o, a)$ assigns lower energies to optimal actions and higher energies to suboptimal ones.

Aspect	Implicit Behavioral Cloning	Diffusion Policies	SE(3)-DiffusionFields	NeRF-based implicit policy (this work)
Action space	Continuous action space	Action sequences	6-DoF pose + approach	6-DoF grasp pose
Scene representation	N/A	N/A	N/A	Pre-trained mVNeRF
Training data	Demonstrations	Demonstrations	Known object pose + geometry (SDF)	Demonstrations + calibrated cameras
Inference setup	RGB images; same camera setup as training	RGB images; same camera setup as training	Known object pose + geometry (SDF)	Calibrated cameras (multi-view RGB)
Training signal	Negative sampling	Denoising objective	Denoising objective	Negative sampling + gradient supervision
Inference	Gradient- or sampling-based	Iterative denoising	Iterative denoising	Adam-based gradient ascent
Multimodality	Yes	Yes	Yes	Yes
Generalization	Same task, out of distribution setup, no sim-to-real	Recovery without demonstration, no sim-to-real	Sim-to-real, limited by pose and geometry requirement	Strong across shapes/clutter, zero-shot sim-to-real

Table 6.1: Comparison of policy families relevant to this dissertation.

The energy function is trained using a negative-sampling approach, where demonstrated actions serve as low energy examples and randomly sampled actions serve as high energy examples. This naturally accommodates multimodal and potentially discontinuous action distributions commonly encountered in robotic manipulation. IBC optimization strategies include derivative-free sampling methods, gradient-based approaches such as Langevin dynamics, and coordinate-descent-based auto-regressive optimizers.

The key advantages of implicit policies are their flexibility in representing complex, multimodal action distributions and their natural integration of action constraints through the energy landscape. However, the performance of implicit policies can be sensitive to optimization choices and the quality of action samples during training.

In this dissertation, the implicit policy introduced in Chapter 3 follows a closely related formulation but defines a differentiable value function $Q(o, a)$ instead of an energy function, leading to maximization at inference (argmax rather than argmin). Optimization is performed with Adam-based gradient ascent over $SE(3)$ instead of Langevin dynamics or derivative-free sampling. Furthermore, the trajectory-informed extension in Chapter 4 augments the negative-sampling based learning setup by supervising the value-function gradients using full demonstration trajectories, providing a proxy signal that shapes the optimization landscape for gradient-based inference.

6.2.2 Diffusion Policies

Chi et al. (2023) introduce diffusion policies as an alternative to energy-based implicit models. Rather than learning an energy function, diffusion policies use a denoising neural network that captures the gradient of the reverse diffusion process. The model is trained by minimizing the difference between diffused actions (using multiple noise levels) and synthetic denoised actions. This approach also enables the integration and generation of trajectories of robotic movements, rather than just final actions.

Given a sequence of observations, the policy uses the learned gradient field to iteratively denoise randomly sampled action sequences. This approach naturally incorporates temporal information and can generate smooth, coherent action sequences. The diffusion framework provides several advantages: unlike energy-based models, diffusion policies avoid the negative sampling problem that can

cause training instability, the denoising process naturally encourages smooth action sequences, and finally, the framework can represent complex, multimodal action distributions.

The relationship between diffusion policies and implicit models is noteworthy: while diffusion policies learn the gradient field directly to update action sequences from noisy ones, implicit models capture a value/energy function and use the actual gradients of these functions to update action sequences from arbitrary states toward optimal actions.

The trajectory-informed model in Chapter 4 similarly leverages gradients to guide robot motion but differs from diffusion policies in three respects. First, it estimates a differentiable value function and uses its true gradients during training and inference. Second, training combines negative-sampling with gradient supervision, in contrast to diffusion policies however only a sinlge level of noise is used. Third, inference uses Adam to select a single high-value grasp pose rather than synthesizing trajectories. While diffusion policies have been demonstrated on real robots with real-world training data, zero-shot sim-to-real transfer has not been reported to the author's knowledge.

6.2.3 SE(3)-DiffusionFields

SE(3)-DiffusionFields Urain et al. (2023) extends diffusion-based approaches specifically to $SE(3)$ pose spaces for robotic grasping and motion planning. This method learns smooth cost functions over $SE(3)$ through denoising score matching, with three main contributions: learning continuous, differentiable cost functions over $SE(3)$ that naturally handle the manifold structure of rotation groups; representing grasp pose distributions as learned cost functions rather than discrete samples; and providing a unified framework for simultaneously resolving grasp selection and motion optimization while considering collision constraints.

The mathematical foundation relies on Lie group theory, where $SE(3)$ poses are represented in the corresponding Lie algebra space for noise addition and gradient

computation. The exponential and logarithmic maps between the Lie group and Lie algebra enable proper handling of the manifold structure.

In contrast to diffusion policies, where the gradient field is learned directly, SE(3)-DiffusionFields learns to match the gradients of the model to the denoised actions, similar to the trajectory loss from Chapter 4.

A key advantage of SE(3)-DiffusionFields is its ability to jointly optimize grasp poses and approach trajectories while considering collision constraints. This differs from approaches that treat grasp selection and motion planning as separate problems. However, the method requires known object poses and geometry (through Signed Distance Fields), which limits its applicability to scenarios with unknown objects.

While SE(3)-DiffusionFields and the trajectory-informed model in Chapter 4 share the use of gradients to guide robot motion, the method in this dissertation additionally applies a negative-sampling based loss that directly supervises the value function's outputs, not only its gradients. Moreover, unlike SE(3)-DiffusionFields, the method in this dissertation does not require object poses or signed distance fields and operates from calibrated multi-view RGB observations alone.

6.3 Neural Fields in Robotics

Neural scene representations have emerged as powerful tools for robotic perception and manipulation, offering advantages over traditional geometric representations through their ability to capture complex visual and geometric properties in learned feature spaces. This enables better generalization across different scenes and conditions, which is a key strength of deep learning and a major reason for its adoption in robotic applications.

6.3.1 NeRFs in Robotic Applications

Several works have explored the application of NeRFs to robotic manipulation tasks. These applications primarily leverage NeRFs for augmenting observations or as feature extractors for explicit policies.

Dex-NeRF (Ichnowski et al. (2021)) uses a NeRF-based model to render high-quality depth images of a scene, which are then fed to DexNet (Mahler et al. (2017)) to compute robust grasp poses. This approach focuses on improving depth rendering quality for existing grasp planning methods rather than learning grasps directly from NeRF representations.

Kerr et al. (2022) follow a similar approach with Evo-NeRF, but instead of focusing on improving depth rendering, the grasp planner network is trained to perform well on NeRF-rendered depth maps. This method utilizes a different NeRF implementation to significantly improve training times compared to Dex-NeRF.

A multi-view NeRF-based approach is utilized in GraspNeRF (Dai et al. (2023)) to estimate a truncated signed distance field in voxels to predict successful grasps. This method more directly integrates NeRF representations into the grasp planning process, though it still operates on discretized voxel representations rather than continuous optimization.

NEAT (Blukis et al. (2023)) demonstrates one-shot learning for manipulation tasks using NeRF-based representations but focuses on explicit policy learning rather than implicit optimization.

However, these applications differ fundamentally from the implicit optimization-based framework presented in this work, as they typically use NeRFs as feature extractors for explicit policies rather than as the foundation for implicit value functions.

6.3.2 Neural Motion Fields

Neural Motion Fields (Chen et al. (2022)) introduce a novel object representation that encodes both object point clouds and relative task trajectories as an implicit value function parameterized by a neural network. Unlike traditional robotic pick-and-place pipelines that consist of separate stages (grasp pose detection, inverse kinematics, collision-free planning, and execution), Neural Motion Fields provide an object-centric representation that models a continuous distribution over $SE(3)$ space.

The key innovation lies in encoding grasp trajectories directly into the neural field representation, enabling reactive grasping through sampling-based model predictive control (MPC) that optimizes the learned value function. This approach addresses the challenge of grasping dynamic objects in constrained environments, where traditional static grasp planning methods often fail.

The method differs from the work presented in this dissertation in several ways: it focuses on dynamic grasping scenarios rather than static objects, uses sampling-based optimization rather than gradient-based methods, and processes segmented point clouds as input rather than multi-view RGB images. However, both approaches share the fundamental insight that neural fields can effectively encode manipulation-relevant information for robotic tasks.

6.3.3 Neural Grasp Distance Fields

Neural Grasp Distance Fields (NGDF, Weng et al. (2023)) formulate grasp learning as a neural field where the input is a 6D pose of a robot end effector and the output is a distance to a continuous manifold of valid grasps for an object. This distance-based representation contrasts with approaches that predict discrete candidate grasps, as the distance can be directly interpreted as a cost function.

Unlike discrete grasp proposals, NGDF provides a continuous field that can be smoothly optimized. The distance cost can be incorporated directly into trajectory

optimizers alongside other costs such as smoothness and collision avoidance. During optimization, grasp targets can smoothly vary as the learned field is continuous, allowing for more robust planning.

NGDF demonstrates strong performance in joint grasp and motion planning, outperforming baselines by 63% execution success while generalizing to unseen query poses and object shapes. The approach shares similarities with the implicit value functions presented in this work, though NGDF focuses specifically on distance-based representations rather than general value functions, and emphasizes joint grasp-motion planning rather than pure grasp selection.

6.3.4 Federated NeRF Training

Recent work has explored distributed training approaches for NeRFs to address privacy and computational constraints. Holden et al. (2023) present the first federated learning algorithm specifically for NeRF training, splitting training effort across multiple compute nodes without pooling images centrally. They introduce low-rank decomposition techniques for NeRF layers to reduce bandwidth consumption during model parameter transmission. Their approach supports cooperative scene modeling using multiple agents while preserving data privacy.

FedNeRF (Zhang and Shao (2024)) proposes a federated learning approach for NeRFs that enables training across multiple data owners while preserving data privacy. The method addresses scenarios where training images are distributed across different locations due to strict regulations and privacy concerns.

DecentNeRFs (Tasneem et al. (2024)) introduces a decentralized approach to NeRF training from crowdsourced images. The method requires approximately 10^4 times less server computing compared to centralized approaches by having users send 3D representations rather than raw data. The approach decomposes users' 3D views into personal and global NeRFs, using optimally weighted aggregation of global components while maintaining privacy through secure aggregation.

6.4 Vision-Language Conditioning

The integration of natural language understanding with visual perception has become increasingly important in robotics, enabling more intuitive human-robot interaction and semantic scene understanding. Recent advances in vision-language models like CLIP have opened new possibilities for grounding language in visual representations.

6.4.1 CLIP and Vision-Language Models

Contrastive Language-Image Pre-training (CLIP, Radford et al. (2021)) represents a breakthrough in vision-language understanding by learning joint embeddings of images and text through contrastive learning on large-scale internet data. CLIP consists of separate image and text encoders that map inputs to a shared embedding space, enabling zero-shot classification and cross-modal retrieval.

The key innovation of CLIP lies in its training objective, which maximizes the cosine similarity between correct image-text pairs while minimizing similarity for incorrect pairs. This contrastive approach enables the model to learn rich semantic representations that capture both visual appearance and linguistic concepts.

CLIP has found widespread application in robotics for tasks requiring semantic understanding, including object detection, scene understanding, and instruction following. Its zero-shot capabilities make it particularly valuable for open-vocabulary scenarios where the set of possible objects or concepts is not known a priori.

RT-2 (Zitkovich et al. (2023)) extends vision-language conditioning to end-to-end robotic control by incorporating vision-language models trained on Internet-scale data directly into robotic policies. RT-2 introduces vision-language-action models (VLAs) that co-fine-tune state-of-the-art vision-language models on both robotic trajectory data and Internet-scale vision-language tasks. The key innovation lies in expressing robotic actions as text tokens, allowing them to be incorporated directly

into the training set alongside natural language tokens. This approach enables RT-2 to achieve emergent capabilities from Internet-scale training, including improved generalization to novel objects, interpretation of commands not present in robot training data, and rudimentary semantic reasoning such as identifying objects by relative properties or functional requirements.

6.4.2 Language Embedded Radiance Fields (LERF)

Kerr et al. (2023) introduce Language Embedded Radiance Fields (LERF), a method for grounding language embeddings from CLIP into NeRF representations. LERF enables open-ended language queries in 3D by learning a dense, multi-scale language field inside NeRF through volume rendering of CLIP embeddings.

LERF's approach centers on learning dense language embeddings at multiple scales within the NeRF representation, enabling hierarchical language queries across the volume. Rather than just rendering color and density as in traditional NeRFs, LERF volume renders CLIP embeddings along training rays, providing multi-view consistency for the language field. This extension of the volume rendering equation allows the method to maintain spatial coherence of semantic information across different viewpoints.

The method employs a multi-level pyramid encoding of ViT image embeddings to capture features at different scales, departing from traditional CLIP applications that use overlapping image patches. This pyramid approach enables more effective capture of both fine-grained and coarse semantic information within the scene. Additionally, LERF incorporates DINO Caron et al. (2021) features alongside CLIP embeddings to improve geometric understanding and object segmentation capabilities, creating a richer multimodal representation.

The training process involves optimizing both the standard NeRF objectives (color and density) and the language embedding consistency across multiple views. The language field is supervised by ensuring that rendered CLIP embeddings along

rays match the corresponding image regions' CLIP features. After optimization, LERF can extract 3D relevancy maps for arbitrary language prompts, enabling pixel-aligned, zero-shot queries without requiring region proposals or masks.

6.5 Positioning of This Work

This work introduces technical and methodological contributions that expand upon and unify existing approaches to neural scene representations and implicit policy learning.

6.5.1 Technical Distinctions

NeRF-Based Implicit Policies: This work represents the first comprehensive exploration of using pre-trained NeRFs as the foundation for implicit grasp policies. While previous work (Ichnowski et al. (2021), Kerr et al. (2022), Dai et al. (2023)) has used NeRFs for robotic applications, these approaches treat NeRFs as feature extractors for explicit policies. In contrast, this work leverages NeRF's learned geometric representations to inform value functions that guide gradient-based optimization in SE(3), combining the geometric understanding of NeRFs with the flexibility of implicit policies.

Trajectory-Aware Learning: The trajectory-informed learning extends implicit behavioral cloning (Florence et al. (2022)) by incorporating demonstrated trajectories into the learning process. Unlike diffusion policies (Chi et al. (2023)) that learn to denoise action sequences, this approach directly supervises the gradients of the value function using demonstrated movements, addressing the optimization landscape quality that is crucial for gradient-based inference.

Federated NeRF Training: While federated learning has been explored for NeRFs Holden et al. (2023), Zhang and Shao (2024), this work specifically addresses its application to robotics scenarios, demonstrating how federated training

can improve NeRF-based representations by leveraging diverse scene distributions across multiple environments while maintaining data privacy.

Multimodal Integration: The integration of CLIP-based vision-language conditioning with NeRF-derived implicit policies creates a multimodal architecture that preserves the continuous optimization framework and fuses geometric, visual and semantic information.

Data Efficiency and Generalization: This work demonstrates that effective NeRF-based manipulation policies can be achieved with relatively limited training data and can generalize across multiple dimensions: from simple geometric shapes to complex YCB objects, from isolated arrangements to cluttered scenes, and from simulation to real-world deployment through zero-shot transfer.

Camera-Robot Decoupling: The proposed approach requires only calibrated cameras at inference and does not assume the same camera-to-robot configuration as during training. As long as intrinsic and extrinsic calibrations are available at runtime, the camera rig can differ from the training setup, effectively decoupling perception from the manipulator.

6.5.2 Methodological Contributions

Support Pose Decomposition (SPD): A method for extending 5-DoF NeRF queries to 6-DoF grasp poses, enabling the use of NeRF representations for grasp pose optimization.

Multi-view VisionNeRF (mVNeRF): An extension of single-view image-conditioned NeRFs Lin et al. (2023) to handle multiple calibrated cameras, improving scene representation quality and robustness.

Transfer Learning Framework: A complete pipeline that combines SPD with a trainable grasp value readout to leverage a frozen, pre-trained mVNeRF for grasping, demonstrating effective transfer of geometric representations from novel

view synthesis to robotic manipulation. Crucially, this transfer enables **zero-shot sim-to-real transfer** without extensive domain randomization, adaptation or augmentation, addressing a common bottleneck in robotic learning.

These contributions collectively address key ideas not entirely explored in the literature: the use of neural scene representations in implicit policies, utilization of trajectory information in imitation learning, and multimodal implicit policies that maintain continuous optimization capabilities.

7 Discussion

This chapter summarizes the thesis's key findings and discusses the performance and implications of NeRF-based implicit grasp policies. The discussion is organized into three sections: an analysis of the experimental results across all contributions (Section 7.1), an examination of the current limitations and challenges (Section 7.2), and an outlook on future research directions (Section 7.3).

7.1 Experimental Results Discussion

The experimental evaluation across Chapters 3, 4, and 5 demonstrates the effectiveness of NeRF-based implicit grasp policies while revealing important insights about their behavior and generalization capabilities.

A key advantage of the proposed approach is its minimal dependency on auxiliary perception systems. Unlike many existing methods for robotic manipulation, particularly those addressing sim-to-real transfer, this work requires only calibrated RGB camera images as input. Most competing approaches depend on additional mechanisms such as object detection, semantic segmentation, explicit object pose estimation, or known object models to bridge the simulation-reality gap or to achieve robust grasping performance. The approach's reliance on geometric consistency rather than object-specific knowledge enables it to generalize to previously unseen objects without requiring additional training or model updates. This stands in contrast to methods that require explicit object databases, pre-computed grasp libraries, or category-specific models.

7.1.1 Transfer Learning from Novel View Synthesis

The fundamental NeRF-based implicit grasp policy introduced in Chapter 3 achieved a 51% success rate in real-world experiments, representing a significant result in zero-shot sim-to-real transfer for robotic grasping. This performance is particularly noteworthy given that the system was trained exclusively on simple prismatic objects in simulation yet successfully generalized to diverse real-world objects including everyday items with complex geometries.

The comparison between single-view and multi-view mVNeRF variants revealed an unexpected finding: while the 3-view model achieved superior novel view synthesis metrics (PSNR: 29.94 vs 25.89, SSIM: 0.96 vs 0.91), the single-view model demonstrated better real-world grasping performance. This counterintuitive result highlights the importance of geometric consistency over photometric fidelity for grasping tasks. The single-view model's blurrier but more geometrically coherent representations proved more suitable for extracting meaningful grasp-relevant features.

The advantage of pre-trained NeRF backbones over models trained from scratch was consistently demonstrated across all scenarios. NeRF-based policies outperformed baseline models in both simulation (strict success rate: 0.70 vs 0.64) and real-world experiments (51 % vs 38 % mean success rate), confirming that geometric pre-training through novel view synthesis provides valuable inductive biases for grasping tasks.

7.1.2 Trajectory-Aware Learning Impact

The trajectory-informed learning methodology introduced in Chapter 4 provided measurable improvements in optimization landscape quality. The dGrasp models exhibited smoother value landscapes with more pronounced peaks around successful grasp poses, reducing local extrema that could trap gradient-based optimization. Furthermore, Bayesian optimization of policy hyperparameters revealed that trajectory-informed models (dGrasp) not only achieved higher peak

success rates during tuning but also exhibited greater robustness to hyperparameter variations, suggesting a more well-shaped optimization landscape.

Despite no substantial gains from hyperparameter tuning or trajectory supervision in the simulated tasks, real-world experiments showed that trajectory supervision improved success from 51 % (Grasp-1v) to 59 % (dGrasp-1v).

The trajectory loss addresses a key limitation of implicit policies: the quality of the optimization landscape. By aligning learned gradients with demonstrated trajectories, the resulting value functions provide better guidance for gradient-based optimization during inference, leading to more reliable convergence to successful grasp poses.

7.1.3 Complementary Extensions

The federated learning experiments provided several insights into distributed training of NeRF-based manipulation policies. The pooled federated model, which aggregated data from diverse scenarios, achieved a 71 % real-world success rate, outperforming even the best centrally trained dGrasp model. This highlights the benefit of broader data exposure for generalization. The fully federated setup, which involved extreme data partitioning, yielded a lower success rate (44%). It is plausible that the reduced number of effective weight updates per client compared to centralized or less partitioned federated training regimes contributed to this performance. Nevertheless, this finding suggests that practical deployments of federated training should carefully consider strategies to ensure both adequate data diversity and sufficient training steps across clients, rather than relying on strict, potentially data-limiting partitioning.

The CLIP-based vision-language conditioning, while achieving only modest improvements over random selection (+14% for color, +22 % for shape), demonstrated the feasibility of integrating semantic understanding with geometric reasoning. The 93.3 % physical grasping success rate confirmed that multimodal conditioning does not compromise the core grasping capabilities.

7.1.4 Why Zero-Shot Sim-to-Real Works

The observed sim-to-real transfer can be attributed to design choices that embed physical structure into both representation and policy. The NeRF backbone uses differentiable volume rendering with calibrated cameras, so known intrinsics and extrinsics ground the representation in metric 3D rather than requiring the network to infer camera geometry. The implicit policy operates directly in pose space and, through projection-based feature sampling, explicitly couples candidate TCP poses with the observed images. Together, these choices reduce the learning burden and align optimization with real-world geometry and imaging physics, enabling transfer to unseen objects without object models or intermediate perception. Notably, the base model without NeRF pretraining already performs competitively, which supports the premise that the projection-based feature query mechanism itself provides meaningful geometric signal that the value readout can exploit.

7.1.5 Challenging Scenarios and Robustness

No explicitly curated *challenge sets* were constructed. However, the zero-shot sim-to-real setting itself is inherently challenging, and real-world experiments were conducted over multiple days with naturally varying lighting. The consistent success rates across days indicate robustness to moderate illumination and scene changes, despite the overall controlled laboratory setup.

7.2 Limitations

While the proposed NeRF-based implicit grasp policy demonstrates promising results, several fundamental limitations constrain its current applicability and performance.

Analysis of failure modes across experiments revealed consistent patterns in the system's generalization capabilities. The policy demonstrated strong performance on objects with clear geometric structure and moderate visual contrast, such as the 3D-printed blocks and everyday items like the hammer and dental floss container.

Systematic failures occurred with specific object categories:

- **Curved objects**: The tennis ball, crochet ball, and red cylinder consistently challenged the system due to the prismatic training bias and SPD strategy.
- **Low-contrast objects**: Dark objects like the LEGO tires and black triangular prism often led to convergence on workspace reflections rather than the objects themselves.
- **Complex geometries**: Objects with concave features or irregular shapes, such as the power drill and hiking boot, sometimes resulted in collision-prone grasp poses.

These patterns suggest that the current approach's limitations are primarily biases due to design choices rather than fundamental constraints, pointing toward specific areas for improvement in future work. The following analysis examines four key limitation categories: training data constraints that introduce geometric and scenario biases, technical dependencies including camera calibration requirements, methodological limitations in the optimization and execution approach, and evaluation constraints that limit the scope of demonstrated capabilities. Each of these areas represents opportunities for targeted improvements rather than insurmountable barriers.

7.2.1 Training Data Constraints

The most significant limitation stems from the restricted training distribution. All models were trained exclusively on the simple-prism scenario, which introduces a strong structural bias toward objects with parallel, opposing surfaces. This bias manifests in several ways:

Support Pose Decomposition Bias: The SPD strategy inherently favors flat surfaces that align with the gripper's parallel jaws. When confronted with curved or highly concave shapes, the optimizer often "slides" off the object onto the workspace, as observed with spherical objects like the tennis ball and crochet ball.

Limited Object Diversity: A significant limitation is the homogeneity of the training data, which consists of successful grasps and lifts involving only simple, easy-to-grasp objects. This constrains the grasp-specific training to a narrow object class, limiting the system's ability to generalize to objects with fundamentally different geometric properties or to learn grasps that account for complex physical constraints, such as an object's center of mass. Consequently, while the NeRF backbone itself benefits from diverse visual pretraining, the overall policy is not explicitly trained to predict physically robust grasps for arbitrary objects. For broader applicability and more complex real-world scenarios, a more comprehensive training distribution, similar in scale and diversity to datasets like ACRONYM (Eppner et al. (2020), featuring 17.7 million grasps across 8,872 objects), would be necessary to cover a wider range of physically viable grasps.

Scenario Specificity: Training on a single scenario (simple-prism) with controlled lighting and backgrounds does not capture the full complexity of real-world environments, potentially limiting robustness to varying conditions.

Despite these constraints, grasping was defined for a parallel-jaw gripper and trained on prismatic objects assembled from rectangular cuboids, yet the policy generalized to a variety of real-world objects. This suggests that the projection-based formulation together with geometric priors from novel view synthesis captures transferable grasp concepts beyond the narrow training distribution.

7.2.2 Technical Dependencies

Several technical requirements impose practical constraints on deployment:

Camera Calibration Sensitivity: The method assumes precisely calibrated and synchronized RGB cameras. Small errors in extrinsics shift the projected support poses on the image plane, misaligning the latent features queried from the NeRF and degrading both grasp-value estimation and gradient guidance.

Computational Requirements: Gradient-based inference is computationally intensive compared to direct prediction methods. Each grasp proposal requires dozens of forward and backward passes through the frozen NeRF and readout network. While inference completes within a few seconds, latency scales linearly with the number of candidate seeds and may become prohibitive for high-speed manipulation or resource-constrained platforms. In practice, runtime also scales with the number of optimization steps of the implicit policy (typical tuned settings: about 21–31 steps across policies), and the number of source views used for feature aggregation (three views in the experiments), in addition to hardware.

Training Runtime: The mVNeRF backbone follows a 1600-epoch schedule with 500k training steps, and the grasp-value readout is trained for 3200 epochs. While wall-clock time depends on hardware and implementation, on a single GPU the NeRF training typically took a couple of days, whereas the grasp-value readout trained in a few hours. Policy tuning via Bayesian optimization was time-consuming, as each candidate configuration required full inference on a tuning dataset.

NeRF Architecture Limitations: The current NeRF architecture, while effective for the demonstrated scenarios, may not represent the optimal choice for robotic manipulation. Simply scaling up the training data or model size do not seem to improve the performance significantly. The volumetric rendering loss provides useful geometric priors, but the architecture's limitations and sensitivity to camera configuration and calibration suggests room for improvement.

7.2.3 Methodological Limitations

Several aspects of the current approach limit its broader applicability:

Open-Loop Operation: The policy operates in an open-loop manner without real-time feedback during execution. This restricts adaptability during task execution and prevents recovery from minor errors or environmental changes.

Single Gripper Configuration: The approach has been evaluated exclusively with a two-finger parallel jaw gripper under the assumption that no obstacles obstruct the approach path. Generalizing the SPD to multi-finger hands, suction cups, would require modifications to the decomposition strategy.

Limited Trajectory Complexity: The demonstrated trajectories used for trajectory-informed training follow relatively simple paths from random starting poses to final grasp configurations. More complex manipulation scenarios requiring sophisticated approach strategies or obstacle avoidance are not addressed.

Simplistic Optimization Procedure: The gradient-based policy optimization employs a basic scheme with initial learning rates and exponential decay over a fixed number of steps. This simple approach lacks sophisticated convergence criteria, adaptive step sizing, or early termination mechanisms that could improve efficiency and robustness. More advanced optimization methods such as adaptive learning rate schedules, convergence detection based on gradient norms or value function changes, or second-order optimization methods could potentially achieve better performance with fewer iterations and more reliable convergence.

7.2.4 Evaluation Constraints

The experimental evaluation, while comprehensive within its scope, has several limitations:

Object Set Limitations: The real-world evaluation used 14 objects, which, while diverse, represents a limited sample of the full space of graspable objects. The selection may not capture all relevant failure modes or edge cases.

Metric Limitations: The binary success/failure metrics, while practical, do not capture nuanced aspects of grasp quality such as stability, force distribution, or robustness to perturbations.

Environmental Constraints: All experiments were conducted in controlled laboratory environments with only limited variation in lighting. Performance in more challenging real-world conditions remains to be demonstrated.

7.3 Future Work

The limitations identified above, combined with the promising results achieved, suggest several important directions for future research. These directions span architectural improvements, methodological extensions, and broader applications of the core principles.

7.3.1 Architectural Improvements

Advanced NeRF Architectures: While the current VisionNeRF-based approach demonstrates effectiveness, newer NeRF architectures offer potential improvements. Instant-NGP Müller et al. (2022) and similar approaches could dramatically reduce training and inference times. More importantly, architectures designed specifically for such applications could reduce sensitivity to camera calibration and configuration while maintaining geometric reasoning capabilities.

Robust Feature Extraction: Developing NeRF variants that are inherently more robust to environmental variations would significantly improve practical applicability. This might involve incorporating uncertainty estimation, multi-scale feature extraction, or alternative geometric representations that are less sensitive to lighting variations.

Cross-Modal NeRF Training: Extending NeRF training with additional sensing modalities (e.g., proximity or range sensors, depth, tactile) could regularize

geometry and improve robustness. Calibrated auxiliary signals can be aligned with the camera model to supervise density/occupancy and surface cues directly, reducing sensitivity to illumination and texture.

Temporal Integration: Extending the framework to handle temporal sequences could enable dynamic scene understanding and temporal consistency in grasp planning. This would be particularly valuable for scenarios involving moving objects or changing environmental conditions.

7.3.2 Training and Data Improvements

Diverse Training Distributions: Addressing the training data constraints requires systematic expansion of the training distribution. This includes:

- More diverse objects and grasps (e.g. ACRONYM)
- Varied lighting conditions and backgrounds
- More complex environments with obstacles, clutter and occlusions
- Different gripper configurations and approach constraints

Self-Supervised Learning: Developing methods for self-supervised data collection could help address the data scarcity problem. Robots could autonomously explore grasping strategies and learn from both successes and failures, gradually expanding their capabilities without extensive human supervision.

Sim-to-Real Transfer: Improving sim-to-real transfer through domain adaptation techniques, more realistic simulation environments, or hybrid training approaches that combine simulated and real data could enhance real-world performance.

7.3.3 Methodological Extensions

Closed-Loop Control: Integrating real-time feedback during grasp execution would enable adaptive behavior and error recovery. This could involve visual servoing based on NeRF features, force feedback integration, or online replanning based on observed execution outcomes.

Multi-Modal Integration: Expanding beyond vision-language conditioning to incorporate other modalities such as tactile feedback, force sensing, or audio cues could enhance the system's understanding of object properties and manipulation requirements.

Hierarchical Planning: Developing hierarchical approaches that combine high-level task planning with low-level grasp execution could enable more complex manipulation scenarios. This might involve integrating the current approach with task and motion planning frameworks.

Collision-Aware Planning: The SPD strategy could be extended beyond the gripper to cover the entire robot body. Combined with NeRF's geometric representation, this could enable collision checking throughout the robot's kinematic chain. Furthermore, this geometric awareness could be integrated with trajectory optimization methods like CHOMP (Covariant Hamiltonian Optimization for Motion Planning, Zucker et al. (2013)) and STOMP (Stochastic Trajectory Optimization for Motion Planning, Kalakrishnan et al. (2011)) to directly generate collision-free paths. This would create a unified framework where the same NeRF representation used for grasp planning also informs safe motion execution.

Explicit NeRF-Based Quality Checks: Leverage the NeRF density within the gripper's closing volume around the TCP to detect object boundaries and penetration, yielding an analytic grasp-quality signal that can gate or modulate the learned value. Such checks could reduce failure modes due to collisions or thin structures.

Other End-Effectors and SPDs: Generalize the Support Pose Decomposition to multi-finger hands or suction grippers and evaluate whether the learned value

readout transfers zero-shot when only the geometric decomposition changes, potentially avoiding retraining.

Reinforcement Learning Integration: The grasp value model developed in this work represents a natural foundation for reinforcement learning approaches to manipulation. The learned value function, which already evaluates action quality given visual observations, could serve as an effective prior for Q-value functions in RL algorithms. This could significantly accelerate convergence by providing informed initial value estimates, reducing the sample complexity typically associated with learning manipulation policies from scratch. Additionally, the rich NeRF-based scene representation could serve as a powerful state representation for RL, potentially enabling more efficient exploration and better generalization across manipulation tasks.

7.3.4 Broader Applications

Beyond Grasping: The core principles of using NeRF-based representations for implicit policy learning could extend to other manipulation tasks such as:

- Object placement and assembly
- Tool use and manipulation
- Deformable object handling
- Multi-robot coordination

Foundation Models for Robotics: The demonstrated effectiveness of geometric pretraining suggests potential for developing foundation models specifically for robotic manipulation. Such models could capture general principles of physical interaction and transfer across diverse manipulation tasks.

Real-World Deployment: Transitioning from laboratory demonstrations to real-world deployment requires addressing robustness, safety, and reliability concerns. This includes developing failure detection and recovery mechanisms, ensuring safe

operation in human environments, and maintaining performance across varying conditions.

7.3.5 Theoretical Developments

Optimization Theory: Deeper theoretical understanding of the optimization landscapes learned by implicit policies could guide architectural choices and training procedures. This includes analyzing convergence properties, local minima structure, and the relationship between trajectory supervision and optimization performance.

Generalization Bounds: Developing theoretical frameworks for understanding when and why NeRF-based policies generalize could guide training data selection and architectural design decisions.

Multi-Task Learning: Investigating how NeRF-based representations can be shared across multiple manipulation tasks could lead to more efficient learning and better generalization.

7.3.6 Practical Considerations

Computational Efficiency: Developing more efficient inference methods, possibly through model compression, knowledge distillation, or specialized hardware acceleration, would make the approach more practical for real-time applications.

Human-Robot Interaction: Enhancing the vision-language conditioning capabilities could enable more natural human-robot interaction, allowing users to specify manipulation goals through natural language or demonstration.

Safety and Reliability: Developing formal verification methods and safety guarantees for implicit policies would be crucial for deployment in safety-critical applications.

7.4 Broader Impact and Significance

The work presented in this dissertation contributes to broader trends in robotics and machine learning:

Neural Scene Representations in Robotics: This work demonstrates the potential of neural scene representations beyond computer vision applications, showing how geometric understanding learned through novel view synthesis can transfer to manipulation tasks.

Implicit Policy Learning: The successful application of implicit policies to robotic manipulation, particularly with gradient-based optimization in $SE(3)$, opens new avenues for policy representation and learning.

Sim-to-Real Transfer: The demonstrated zero-shot sim-to-real transfer capabilities suggest that geometric pretraining may be a key component in bridging the simulation-reality gap for robotic manipulation.

Multimodal Learning: The integration of vision-language conditioning with geometric reasoning provides a template for combining semantic understanding with precise spatial manipulation capabilities.

The convergence of these contributions suggests a promising direction for future robotics research, where neural scene representations serve as a foundation for learning complex manipulation behaviors that can generalize across diverse real-world scenarios.

8 Conclusion

This dissertation addresses the fundamental research question: "Can neural scene representations trained for novel view synthesis serve as foundations for robotic grasping?" The answer is definitively yes. A novel framework has been introduced that uses pre-trained NeRF representations as frozen backbones for implicit grasp policies, achieving 71% success rate in real-world experiments (Table 5.6) through zero-shot sim-to-real transfer when the NeRF backbone is pretrained on diverse simulated scenarios – requiring only calibrated RGB cameras without object models, segmentation, pose estimation, or domain adaptation techniques typically employed by other methods.

The work establishes three main scientific and technical contributions. First, a novel NeRF-based implicit grasping framework that adapts image-conditioned NeRFs for robotic grasping through multi-view extensions, Support Pose Decomposition, and a grasp value readout that leverages intermediate NeRF activations, enabling continuous optimization in SE(3) rather than discrete candidate search. Second, trajectory-informed learning that enhances grasp value learning by incorporating full demonstration trajectories, creating smoother optimization landscapes that improve convergence and generalization. Third, framework extensions that demonstrate versatility through federated learning across scenarios and vision-language conditioning for task specification. See Figure 8.1 for a summary of the contributions. Experimental validation demonstrates remarkable generalization capabilities, with the framework achieving substantial success rates in real-world grasping experiments despite being trained exclusively on simple simulated objects.

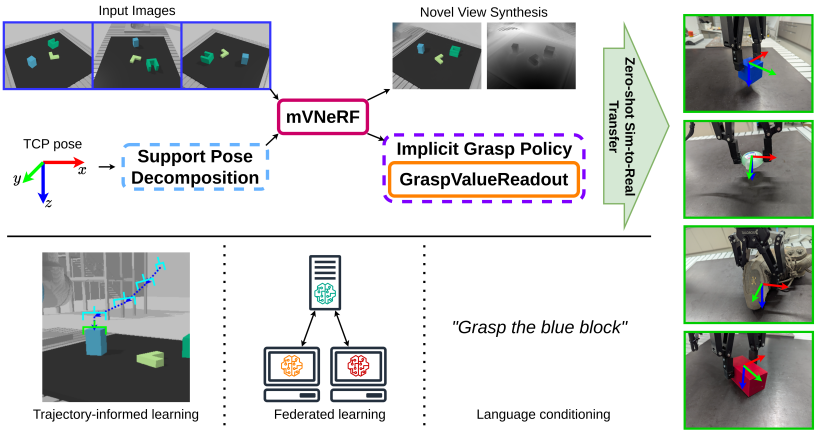


Figure 8.1: Summary of contributions. Calibrated multi-view RGB observations are encoded by an image-conditioned mVNeRF that serves as a frozen geometric backbone for grasping. SPD converts 6-DoF TCP poses into NeRF queries and the *GraspValueReadout* maps intermediate activations to a scalar value, enabling gradient-based optimization in $SE(3)$. The framework is enhanced by trajectory-informed learning, and federated learning and language conditioning are investigated as extensions. In real-world experiments, the grasping policy achieves 71% grasp success via zero-shot sim-to-real transfer.

The success stems from the volumetric rendering loss that enforces consistency with the fundamental physics of light transport and camera projection. This loss function embeds knowledge from computer graphics about how cameras, light, and 3D geometry interact, resulting in learned representations that inherently respect geometric constraints. By demonstrating that this physics-informed geometric understanding transfers to manipulation tasks, the work establishes a new paradigm for connecting computer vision advances with robotic manipulation and opens pathways for leveraging similar principles across neural scene representations for robotics applications. The implicit policy formulation provides a principled approach to exploiting the differentiability of modern neural networks.

This work has implications for both practical robotics and machine learning research. By enabling robots to adapt to new objects without retraining, the approach

offers potential value for logistics, manufacturing, healthcare, and domestic applications. The framework extensions further expand its applicability: vision-language conditioning establishes a foundation for natural language interaction, while federated learning enables collaborative improvement across robot fleets without data sharing. From a scientific perspective, the work demonstrates how intermediate representations learned for one task can transfer to related tasks, supporting broader trends toward foundation models in machine learning. The combination of geometric understanding, differentiable optimization, and multimodal conditioning provides a foundation for developing more capable and generalizable approaches for robotic manipulation in unstructured and changing environments. While the language conditioning extensions face current limitations that constrain immediate practical deployment, the core framework's promising results and clear directions for improvement suggest that NeRF-like approaches will play an increasingly important role in robotic manipulation, potentially transforming how robots perceive, understand, and interact with the world around them.

Acronyms

CNN Convolutional Neural Network. 7–9, 13, 14, 20, 83

FCN Fully Convolutional Network. 8, 13, 84

mVNeRF multi-view VisionNeRF. 20, 21, 24, 26, 28, 29, 31, 36, 84

NeRF Neural Radiance Field. 3–12, 14, 15, 17, 19–21, 26–28, 83

SPD Support Pose Decomposition. 28

TCP Tool Center Point. 18, 19, 26–28, 31, 33

ViT Vision Transformer. 7, 8, 12–14, 20, 21

A Implementation Differences from Published Papers

A.1 6-DoF Grasp Pose Evaluation and Optimization via Transfer Learning from NeRFs

After the initial publication of the concept in ICRA 2024, the framework and methodologies have been further developed and refined. The implementation described in Chapter 3 differs from the original ICRA paper Sóti et al. (2024) in several key aspects outlined below.

The most significant changes relate to how training data is structured and processed. While the original work used positionally encoded direction vectors, this dissertation employs raw direction vectors. The amount of visual information per scene was increased from 16 views per scene to 50 views per scene for grasp training. Additionally, the demonstration structure was modified from one demonstration per scene to one demonstration per object. Additionally, the training data generation differs in terms of scene randomization. In the original paper, lighting conditions and object colors were fixed throughout both training and testing (e.g., the L-shaped object was always red). In this dissertation, both lighting position and object colors are randomized during data generation and testing.

Regarding the training procedure, the loss function was changed from cross entropy loss to KL divergence loss. The training duration was also extended from 400 epochs to 3200 epochs.

The activation functions in the grasp value readout network were changed from ReLU to ELU. This change was made to ensure stable higher-order gradients, anticipating the trajectory supervision extension in Chapter 4, which requires second-order derivatives.

Finally, the optimization strategy for grasp pose refinement was modified from sequential optimization, which optimized translation and rotation separately, to synchronized optimization that optimizes both simultaneously.

A.2 dGrasp: NeRF-Informed implicit grasp policies with supervised optimization slopes

The implementation described in Chapter 4 differs from the RAS journal paper Sóti et al. (2025) in several aspects.

The training data and training configuration for grasp training follow the same approach as described in the previous section for the ICRA paper. This includes 16 views and one demonstration per scene with fixed lighting and object colors, cross entropy loss, and training for 400 epochs.

Regarding rotation representation, the RAS paper tested both 6D rotation representation and quaternions, with quaternions proving to be better when using the trajectory loss. This dissertation uses only quaternions for orientation.

The hyperparameter optimization procedure differs in the number of optimization steps used during Bayesian optimization. The RAS paper uses 100 optimization steps for hyperparameter tuning, while this dissertation uses 60 steps. Additionally, the initial learning rate ranges differ: the RAS paper uses $[0, 1]$ for both translation and rotation, while this dissertation uses $[0, 0.4]$ for translation and $[0, 1.57]$ for rotation.

B Visual Embedding – Architectures

B.1 ConvEncoder

The convolutional encoder is a key component of the visual embedding network used in mVNeRF (Chapter 3, Figure 3.2). This lightweight CNN extracts local features from input images, providing spatially-aligned representations that complement the global features extracted by the Vision Transformer. The architecture is detailed in Figure B.1.

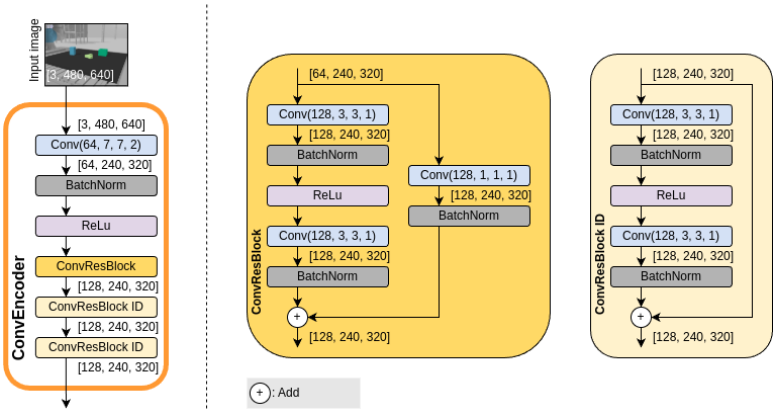


Figure B.1: Convolutional encoder architecture. A lightweight CNN extracts local features from the input image. Detailed description is provided in the appendix.

B.2 ViTEncoder

This section describes how a pre-trained Vision Transformer is used to extract pixel-aligned multi-level features from input images. While the input images have a resolution of 640x480, which is suitable for the convolutional encoder, the pre-trained ViT was originally trained on 224x224 images. Since the positional encodings are learned specifically for this resolution, the input images are resized to 224x224 before processing by the ViT. The transformer then generates patch embeddings that are converted into multi-level feature maps at different resolutions, as shown in Figure B.2.

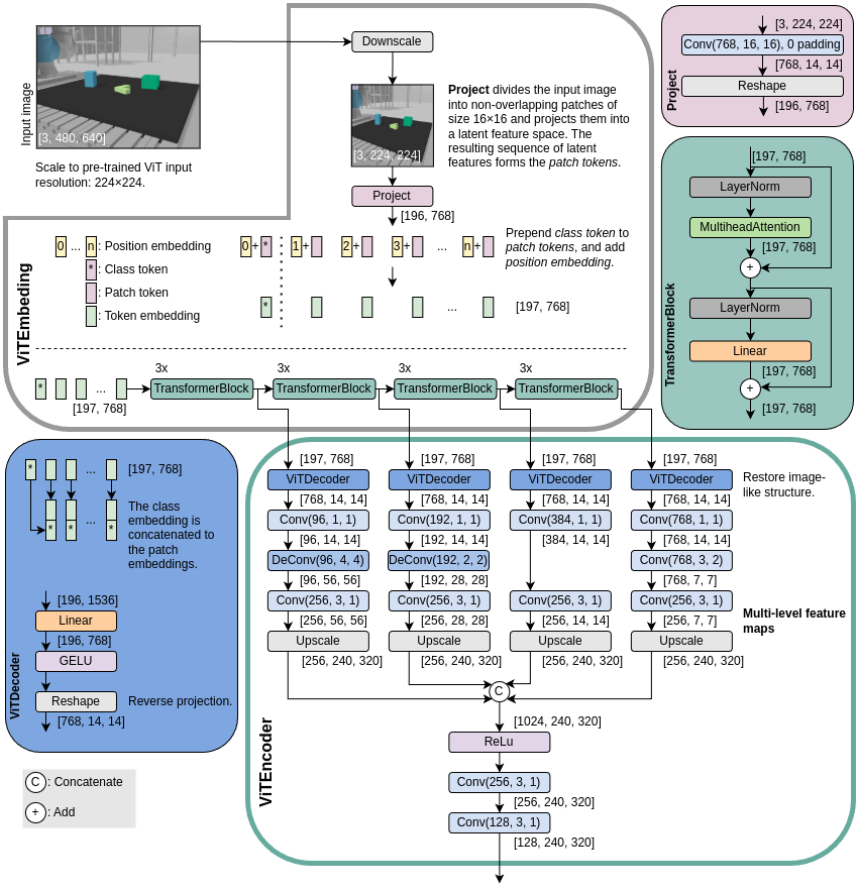


Figure B.2: Vision Transformer encoder architecture. The ViT processes image patches to extract global features, which are then transformed into multi-level feature maps. Detailed description is provided in the appendix.

List of Figures

List of Tables

List of Publications

Journal articles

Gergely Sóti, Xi Huang, Christian Wurll, and Björn Hein. dgrasp: Nerf-informed implicit grasp policies with supervised optimization slopes. *Robotics and Autonomous Systems*, 186:104921, 2025.

Conference contributions

Gergely Sóti, Björn Hein, and Christian Wurll. Gradient based grasp pose optimization on a nerf that approximates grasp success. In *International Conference on Intelligent Autonomous Systems*, pages 303–318. Springer, 2023a.

Gergely Sóti, Xi Huang, Christian Wurll, and Björn Hein. Train what you know–precise pick-and-place with transporter networks. In *2023 IEEE International Conference on Robotics and Automation (ICRA)*, pages 5814–5820. IEEE, 2023b.

Gergely Sóti, Xi Huang, Christian Wurll, and Björn Hein. 6-dof grasp pose evaluation and optimization via transfer learning from nerfs. In *2024 IEEE International Conference on Robotics and Automation (ICRA)*, pages 9495–9501. IEEE, 2024.

Bibliography

Michal Adamkiewicz, Timothy Chen, Adam Caccavale, Rachel Gardner, Preston Culbertson, Jeannette Bohg, and Mac Schwager. Vision-only robot navigation in a neural radiance world. *IEEE Robotics and Automation Letters*, 7(2): 4606–4613, 2022.

Yahav Avigal, Lars Berscheid, Tamim Asfour, Torsten Kröger, and Ken Goldberg. Speedfolding: Learning efficient bimanual folding of garments. In *2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 1–8. IEEE, 2022.

Lars Berscheid, Pascal Meißner, and Torsten Kröger. Self-supervised learning for precise pick-and-place without object model. *IEEE Robotics and Automation Letters*, 5(3):4828–4835, 2020.

Lars Berscheid, Christian Friedrich, and Torsten Kröger. Robot learning of 6 dof grasping using model-based adaptive primitives. In *2021 IEEE International Conference on Robotics and Automation (ICRA)*, pages 4474–4480. IEEE, 2021.

Valts Blukis, Taeyeop Lee, Jonathan Tremblay, Bowen Wen, In So Kweon, Kuk-Jin Yoon, Dieter Fox, and Stan Birchfield. One-shot neural fields for 3d object understanding, 2023.

Jeannette Bohg, Antonio Morales, Tamim Asfour, and Danica Kragic. Data-driven grasp synthesis—a survey. *IEEE Transactions on robotics*, 30(2):289–309, 2013.

Anthony Brohan, Noah Brown, Justice Carbajal, Yevgen Chebotar, Joseph Dabis, Chelsea Finn, Keerthana Gopalakrishnan, Karol Hausman, Alex Herzog, Jasmine Hsu, et al. Rt-1: Robotics transformer for real-world control at scale. *arXiv preprint arXiv:2212.06817*, 2022.

Berk Calli, Arjun Singh, Aaron Walsman, Siddhartha Srinivasa, Pieter Abbeel, and Aaron M Dollar. The ycb object and model set: Towards common benchmarks for manipulation research. In *2015 international conference on advanced robotics (ICAR)*, pages 510–517. IEEE, 2015.

Mathilde Caron, Hugo Touvron, Ishan Misra, Hervé Jégou, Julien Mairal, Piotr Bojanowski, and Armand Joulin. Emerging properties in self-supervised vision transformers. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 9650–9660, 2021.

Jiazhong Cen, Zanwei Zhou, Jiemin Fang, Wei Shen, Lingxi Xie, Xiaopeng Zhang, and Qi Tian. Segment anything in 3d with nerfs. *arXiv preprint arXiv:2304.12308*, 2023.

Yun-Chun Chen, Adithyavairavan Murali, Balakumar Sundaralingam, Wei Yang, Animesh Garg, and Dieter Fox. Neural motion fields: Encoding grasp trajectories as implicit value functions. *arXiv preprint arXiv:2206.14854*, 2022.

Cheng Chi, Siyuan Feng, Yilun Du, Zhenjia Xu, Eric Cousineau, Benjamin Burchfiel, and Shuran Song. Diffusion policy: Visuomotor policy learning via action diffusion. *arXiv preprint arXiv:2303.04137*, 2023.

Julian Chibane, Gerard Pons-Moll, et al. Neural unsigned distance fields for implicit function learning. *Advances in Neural Information Processing Systems*, 33:21638–21652, 2020.

Qiyu Dai, Yan Zhu, Yiran Geng, Ciyu Ruan, Jiazhao Zhang, and He Wang. Graspnerf: Multiview-based 6-dof grasp detection for transparent and specular objects using generalizable nerf. *arXiv preprint arXiv:2210.06575*, 2022.

Qiyu Dai, Yan Zhu, Yiran Geng, Ciyu Ruan, Jiazhao Zhang, and He Wang. Graspnerf: multiview-based 6-dof grasp detection for transparent and specular

objects using generalizable nerf. In *2023 IEEE International Conference on Robotics and Automation (ICRA)*, pages 1757–1763. IEEE, 2023.

Xinke Deng, Yu Xiang, Arsalan Mousavian, Clemens Eppner, Timothy Bretl, and Dieter Fox. Self-supervised 6d object pose estimation for robot manipulation. In *2020 IEEE International Conference on Robotics and Automation (ICRA)*, pages 3665–3671. IEEE, 2020.

Coline Devin, Payam Rowghanian, Chris Vigorito, Will Richards, and Khashayar Rohanimanesh. Self-supervised goal-conditioned pick and place. *arXiv preprint arXiv:2008.11466*, 2020.

Zhikai Dong, Sicheng Liu, Tao Zhou, Hui Cheng, Long Zeng, Xingyao Yu, and Houde Liu. Ppr-net: point-wise pose regression network for instance segmentation and 6d pose estimation in bin-picking scenarios. In *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 1773–1780. IEEE, 2019.

Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, et al. An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929*, 2020.

Yilun Du, Shuang Li, Joshua Tenenbaum, and Igor Mordatch. Improved contrastive divergence training of energy based models. *arXiv preprint arXiv:2012.01316*, 2020.

Clemens Eppner, Arsalan Mousavian, and Dieter Fox. ACRONYM: A large-scale grasp dataset based on simulation. In *2021 IEEE Int. Conf. on Robotics and Automation, ICRA*, 2020.

Bin Fang, Shidong Jia, Di Guo, Muhua Xu, Shuhuan Wen, and Fuchun Sun. Survey of imitation learning for robotic manipulation. *International Journal of Intelligent Robotics and Applications*, 3:362–369, 2019.

Roger Fletcher. *Practical methods of optimization*. John Wiley & Sons, 2000.

Pete Florence, Corey Lynch, Andy Zeng, Oscar A Ramirez, Ayzaan Wahid, Laura Downs, Adrian Wong, Johnny Lee, Igor Mordatch, and Jonathan Tompson. Implicit behavioral cloning. In *Conference on robot learning*, pages 158–168. PMLR, 2022.

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

Peter R Florence, Lucas Manuelli, and Russ Tedrake. Dense object nets: Learning dense visual object descriptors by and for robotic manipulation. *arXiv preprint arXiv:1806.08756*, 2018.

Marcus Gualtieri, Andreas ten Pas, and Robert Platt. Pick and place without geometric object models. In *2018 IEEE international conference on robotics and automation (ICRA)*, pages 7433–7440. IEEE, 2018.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

Lachlan Holden, Feras Dayoub, David Harvey, and Tat-Jun Chin. Federated neural radiance fields. *arXiv preprint arXiv:2305.01163*, 2023.

Dániel Horváth, Kristóf Bocsi, Gábor Erdős, and Zoltán Istenes. Sim2real grasp pose estimation for adaptive robotic applications. *arXiv preprint arXiv:2211.01048*, 2022.

Haojie Huang, Dian Wang, Robin Walter, and Robert Platt. Equivariant transporter network. *arXiv preprint arXiv:2202.09400*, 2022.

Andrew Hundt, Benjamin Killeen, Nicholas Greene, Hongtao Wu, Heeyeon Kwon, Chris Paxton, and Gregory D Hager. "good robot!": Efficient reinforcement learning for multi-step visual tasks with sim to real transfer. *IEEE Robotics and Automation Letters*, 5(4):6724–6731, 2020.

Ahmed Hussein, Mohamed Medhat Gaber, Eyad Elyan, and Chrisina Jayne. Imitation learning: A survey of learning methods. *ACM Computing Surveys (CSUR)*, 50(2):1–35, 2017.

Jeffrey Ichnowski, Yahav Avigal, Justin Kerr, and Ken Goldberg. Dex-nerf: Using a neural radiance field to grasp transparent objects. *arXiv preprint arXiv:2110.14217*, 2021.

Mrinal Kalakrishnan, Sachin Chitta, Evangelos Theodorou, Peter Pastor, and Stefan Schaal. Stomp: Stochastic trajectory optimization for motion planning. In *2011 IEEE international conference on robotics and automation*, pages 4569–4574. IEEE, 2011.

Justin Kerr, Letian Fu, Huang Huang, Yahav Avigal, Matthew Tancik, Jeffrey Ichnowski, Angjoo Kanazawa, and Ken Goldberg. Evo-nerf: Evolving nerf for sequential robot grasping of transparent objects. In *6th Annual Conference on Robot Learning*.

Justin Kerr, Letian Fu, Huang Huang, Yahav Avigal, Matthew Tancik, Jeffrey Ichnowski, Angjoo Kanazawa, and Ken Goldberg. Evo-nerf: Evolving nerf for sequential robot grasping of transparent objects. In *6th Annual Conference on Robot Learning*, 2022.

Justin Kerr, Chung Min Kim, Ken Goldberg, Angjoo Kanazawa, and Matthew Tancik. Lerf: Language embedded radiance fields. *arXiv preprint arXiv:2303.09553*, 2023.

Mohi Khansari, Daniel Kappler, Jianlan Luo, Jeff Bingham, and Mrinal Kalakrishnan. Action image representation: Learning scalable deep grasping policies with zero real world data. In *2020 IEEE International Conference on Robotics and Automation (ICRA)*, pages 3597–3603. IEEE, 2020.

Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

Kilian Kleeberger and Marco F Huber. Single shot 6d object pose estimation. In *2020 IEEE International Conference on Robotics and Automation (ICRA)*, pages 6239–6245. IEEE, 2020.

Kilian Kleeberger, Richard Bormann, Werner Kraus, and Marco F Huber. A survey on learning-based robotic grasping. *Current Robotics Reports*, 1:239–249, 2020.

Jens Kober, J Andrew Bagnell, and Jan Peters. Reinforcement learning in robotics: A survey. *The International Journal of Robotics Research*, 32(11):1238–1274, 2013.

Hiroaki Kotera. A scene-referred color transfer for pleasant imaging on display. In *IEEE International Conference on Image Processing 2005*, volume 2, pages II–5. IEEE, 2005.

Tejas D Kulkarni, Ankush Gupta, Catalin Ionescu, Sebastian Borgeaud, Malcolm Reynolds, Andrew Zisserman, and Volodymyr Mnih. Unsupervised learning of object keypoints for perception and control. *Advances in neural information processing systems*, 32, 2019.

Sergey Levine, Peter Pastor, Alex Krizhevsky, Julian Ibarz, and Deirdre Quillen. Learning hand-eye coordination for robotic grasping with deep learning and large-scale data collection. *The International journal of robotics research*, 37 (4-5):421–436, 2018.

Michael H Lim, Andy Zeng, Brian Ichter, Maryam Bandari, Erwin Coumans, Claire Tomlin, Stefan Schaal, and Aleksandra Faust. Multi-task learning with sequence-conditioned transporter networks. In *2022 International Conference on Robotics and Automation (ICRA)*, pages 2489–2496. IEEE, 2022.

Kai-En Lin, Yen-Chen Lin, Wei-Sheng Lai, Tsung-Yi Lin, Yi-Chang Shih, and Ravi Ramamoorthi. Vision transformer for nerf-based view synthesis from a single input image. In *Proceedings of the IEEE/CVF winter conference on applications of computer vision*, pages 806–815, 2023.

Xingyu Liu, Rico Jonschkowski, Anelia Angelova, and Kurt Konolige. Keypose: Multi-view 3d labeling and keypoint estimation for transparent objects. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 11602–11610, 2020.

Jeffrey Mahler, Jacky Liang, Sherdil Niyaz, Michael Laskey, Richard Doan, Xinyu Liu, Juan Aparicio Ojea, and Ken Goldberg. Dex-net 2.0: Deep learning to plan robust grasps with synthetic point clouds and analytic grasp metrics. *arXiv preprint arXiv:1703.09312*, 2017.

Jeffrey Mahler, Matthew Matl, Vishal Satish, Michael Danielczuk, Bill DeRose, Stephen McKinley, and Ken Goldberg. Learning ambidextrous robot grasping policies. *Science Robotics*, 4(26):eaau4984, 2019.

Lucas Manuelli, Wei Gao, Peter Florence, and Russ Tedrake. kpam: Keypoint affordances for category-level robotic manipulation. In *The International Symposium of Robotics Research*, pages 132–157. Springer, 2019.

Ricardo Martin-Brualla, Noha Radwan, Mehdi SM Sajjadi, Jonathan T Barron, Alexey Dosovitskiy, and Daniel Duckworth. Nerf in the wild: Neural radiance fields for unconstrained photo collections. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 7210–7219, 2021.

Lars Mescheder, Michael Oechsle, Michael Niemeyer, Sebastian Nowozin, and Andreas Geiger. Occupancy networks: Learning 3d reconstruction in function space. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 4460–4470, 2019.

Ben Mildenhall, Pratul P. Srinivasan, Matthew Tancik, Jonathan T. Barron, Ravi Ramamoorthi, and Ren Ng. Nerf: Representing scenes as neural radiance fields for view synthesis. In *ECCV*, 2020.

Ben Mildenhall, Pratul P Srinivasan, Matthew Tancik, Jonathan T Barron, Ravi Ramamoorthi, and Ren Ng. Nerf: Representing scenes as neural radiance fields for view synthesis. *Communications of the ACM*, 65(1):99–106, 2021.

Thomas Müller, Alex Evans, Christoph Schied, and Alexander Keller. Instant neural graphics primitives with a multiresolution hash encoding. *ACM transactions on graphics (TOG)*, 41(4):1–15, 2022.

Anusha Nagabandi, Kurt Konolige, Sergey Levine, and Vikash Kumar. Deep dynamics models for learning dexterous manipulation. In *Conference on Robot Learning*, pages 1101–1112. PMLR, 2020.

Rhys Newbury, Morris Gu, Lachlan Chumbley, Arsalan Mousavian, Clemens Eppner, Jürgen Leitner, Jeannette Bohg, Antonio Morales, Tamim Asfour, Danica Kragic, et al. Deep learning approaches to grasp synthesis: A review. *IEEE Transactions on Robotics*, 2023.

Hai Nguyen and Hung La. Review of deep reinforcement learning for robot manipulation. In *2019 Third IEEE International Conference on Robotic Computing (IRC)*, pages 590–595. IEEE, 2019.

F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12: 2825–2830, 2011.

Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, et al. Learning transferable visual models from natural language supervision. In *International conference on machine learning*, pages 8748–8763. PmLR, 2021.

Rouhollah Rahmatizadeh, Pooya Abolghasemi, Ladislau Bölöni, and Sergey Levine. Vision-based multi-task manipulation for inexpensive robots using end-to-end learning from demonstration. In *2018 IEEE international conference on robotics and automation (ICRA)*, pages 3758–3765. IEEE, 2018.

Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein,

et al. Imagenet large scale visual recognition challenge. *International journal of computer vision*, 115:211–252, 2015.

Daniel Seita, Pete Florence, Jonathan Tompson, Erwin Coumans, Vikas Sindhwani, Ken Goldberg, and Andy Zeng. Learning to rearrange deformable cables, fabrics, and bags with goal-conditioned transporter networks. In *2021 IEEE International Conference on Robotics and Automation (ICRA)*, pages 4568–4575. IEEE, 2021.

Mohit Shridhar, Lucas Manuelli, and Dieter Fox. Cliport: What and where pathways for robotic manipulation. In *Conference on Robot Learning*, pages 894–906. PMLR, 2022.

Shuran Song, Andy Zeng, Johnny Lee, and Thomas Funkhouser. Grasping in the wild: Learning 6dof closed-loop grasping from low-cost demonstrations. *IEEE Robotics and Automation Letters*, 5(3):4978–4985, 2020.

Zaid Tasneem, Akshat Dave, Abhishek Singh, Kushagra Tiwary, Praneeth Vepakomma, Ashok Veeraraghavan, and Ramesh Raskar. Decentnerfs: Decentralized neural radiance fields from crowdsourced images. In *European Conference on Computer Vision*, pages 144–161. Springer, 2024.

Josh Tobin, Rachel Fong, Alex Ray, Jonas Schneider, Wojciech Zaremba, and Pieter Abbeel. Domain randomization for transferring deep neural networks from simulation to the real world. In *2017 IEEE/RSJ international conference on intelligent robots and systems (IROS)*, pages 23–30. IEEE, 2017.

Julen Urain, Niklas Funk, Jan Peters, and Georgia Chalvatzaki. Se (3)-diffusionfields: Learning smooth cost functions for joint grasp and motion optimization through diffusion. In *2023 IEEE International Conference on Robotics and Automation (ICRA)*, pages 5923–5930. IEEE, 2023.

He Wang, Srinath Sridhar, Jingwei Huang, Julien Valentin, Shuran Song, and Leonidas J Guibas. 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*, pages 2642–2651, 2019.

Peng Wang, Lingjie Liu, Yuan Liu, Christian Theobalt, Taku Komura, and Wenping Wang. Neus: Learning neural implicit surfaces by volume rendering for multi-view reconstruction. *arXiv preprint arXiv:2106.10689*, 2021.

Zhou Wang, Alan C Bovik, Hamid R Sheikh, and Eero P Simoncelli. Image quality assessment: from error visibility to structural similarity. *IEEE transactions on image processing*, 13(4):600–612, 2004.

Moritz Weisenböhler, Björn Hein, and Christian Wurll. On scene engineering and domain randomization: Synthetic data for industrial item picking. In *International Conference on Intelligent Autonomous Systems*, pages 643–660. Springer, 2022.

Thomas Weng, David Held, Franziska Meier, and Mustafa Mukadam. Neural grasp distance fields for robot manipulation. In *2023 IEEE International Conference on Robotics and Automation (ICRA)*, pages 1814–1821. IEEE, 2023.

Ross Wightman. Pytorch image models. `https://github.com/rwightman/pytorch-image-models`, 2019.

Yilin Wu, Wilson Yan, Thanard Kurutach, Lerrel Pinto, and Pieter Abbeel. Learning to manipulate deformable objects without demonstrations. *arXiv preprint arXiv:1910.13439*, 2019.

Yu Xiang, Tanner Schmidt, Venkatraman Narayanan, and Dieter Fox. Posecnn: A convolutional neural network for 6d object pose estimation in cluttered scenes. *arXiv preprint arXiv:1711.00199*, 2017.

Lin Yen-Chen, Pete Florence, Jonathan T Barron, Alberto Rodriguez, Phillip Isola, and Tsung-Yi Lin. inerf: Inverting neural radiance fields for pose estimation. In *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 1323–1330. IEEE, 2021.

Lin Yen-Chen, Pete Florence, Jonathan T Barron, Tsung-Yi Lin, Alberto Rodriguez, and Phillip Isola. Nerf-supervision: Learning dense object descriptors from neural radiance fields. In *2022 International Conference on Robotics and Automation (ICRA)*, pages 6496–6503. IEEE, 2022.

Alex Yu, Ruilong Li, Matthew Tancik, Hao Li, Ren Ng, and Angjoo Kanazawa. Plenoctrees for real-time rendering of neural radiance fields. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 5752–5761, 2021a.

Alex Yu, Vickie Ye, Matthew Tancik, and Angjoo Kanazawa. pixelnerf: Neural radiance fields from one or few images. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 4578–4587, 2021b.

Kevin Zakka, Andy Zeng, Johnny Lee, and Shuran Song. Form2fit: Learning shape priors for generalizable assembly from disassembly. In *2020 IEEE International Conference on Robotics and Automation (ICRA)*, pages 9404–9410. IEEE, 2020.

Matthew D Zeiler, Dilip Krishnan, Graham W Taylor, and Rob Fergus. Deconvolutional networks. In *2010 IEEE Computer Society Conference on computer vision and pattern recognition*, pages 2528–2535. IEEE, 2010.

Andy Zeng, Kuan-Ting Yu, Shuran Song, Daniel Suo, Ed Walker, Alberto Rodriguez, and Jianxiong Xiao. Multi-view self-supervised deep learning for 6d pose estimation in the amazon picking challenge. In *2017 IEEE international conference on robotics and automation (ICRA)*, pages 1386–1383. IEEE, 2017.

Andy Zeng, Pete Florence, Jonathan Tompson, Stefan Welker, Jonathan Chien, Maria Attarian, Travis Armstrong, Ivan Krasin, Dan Duong, Vikas Sindhwani, et al. Transporter networks: Rearranging the visual world for robotic manipulation. *arXiv preprint arXiv:2010.14406*, 2020.

Andy Zeng, Pete Florence, Jonathan Tompson, Stefan Welker, Jonathan Chien, Maria Attarian, Travis Armstrong, Ivan Krasin, Dan Duong, Vikas Sindhwani, et al. Transporter networks: Rearranging the visual world for robotic manipulation. In *Conference on Robot Learning*, pages 726–747. PMLR, 2021.

Andy Zeng, Shuran Song, Kuan-Ting Yu, Elliott Donlon, Francois R Hogan, Maria Bauza, Daolin Ma, Orion Taylor, Melody Liu, Eudald Romo, et al. Robotic

pick-and-place of novel objects in clutter with multi-affordance grasping and cross-domain image matching. *The International Journal of Robotics Research*, 41(7):690–705, 2022.

Richard Zhang, Phillip Isola, Alexei A Efros, Eli Shechtman, and Oliver Wang. The unreasonable effectiveness of deep features as a perceptual metric. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 586–595, 2018a.

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

Yintian Zhang and Ziyu Shao. Federated neural radiance field for distributed intelligence. *arXiv preprint arXiv:2406.10474*, 2024.

Yi Zhou, Connelly Barnes, Jingwan Lu, Jimei Yang, and Hao Li. On the continuity of rotation representations in neural networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5745–5753, 2019.

Menglong Zhu, Konstantinos G Derpanis, Yinfei Yang, Samarth Brahmbhatt, Mabel Zhang, Cody Phillips, Matthieu Lecce, and Kostas Daniilidis. Single image 3d object detection and pose estimation for grasping. In *2014 IEEE International Conference on Robotics and Automation (ICRA)*, pages 3936–3943. IEEE, 2014.

Brianna Zitkovich, Tianhe Yu, Sichun Xu, Peng Xu, Ted Xiao, Fei Xia, Jialin Wu, Paul Wohlhart, Stefan Welker, Ayzaan Wahid, et al. Rt-2: Vision-language-action models transfer web knowledge to robotic control. In *Conference on Robot Learning*, pages 2165–2183. PMLR, 2023.

Matt Zucker, Nathan Ratliff, Anca D Dragan, Mihail Pivtoraiko, Matthew Klingensmith, Christopher M Dellin, J Andrew Bagnell, and Siddhartha S Srinivasa. Chomp: Covariant hamiltonian optimization for motion planning. *The International journal of robotics research*, 32(9-10):1164–1193, 2013.