



UNIVERSITÀ  
DI TRENTO

Department of  
Information Engineering and Computer Science

---

Doctoral Programme in  
Information Engineering and Computer Science

# NEURAL REPRESENTATION FOR ROBOTICS

Matteo Bortolon

Advisor

Dr. Fabio Poiesi  
Fondazione Bruno Kessler

Co-Advisor

Dr. Alessio Del Bue  
Fondazione Istituto Italiano di Tecnologia

---

April 2025





# Abstract

Modern robotic systems often exhibit limited flexibility, constraining their range of applications. Consequently, they are restricted to particular tasks, where even minor modifications, such as changes in object dimensions, can require considerable reconfiguration effort. A major bottleneck in achieving adaptable robotic systems is the perception component. This component provides critical information about the objects to manipulate, primarily through vision cameras. Accurate perception is especially crucial in assembly tasks, where robots must precisely localize objects for manipulation. The localization process is traditionally divided into three interdependent subtasks: geometry reconstruction, pose estimation, and grasp prediction.

Geometry reconstruction has conventionally relied on depth cameras. However, these cameras are susceptible to numerous limitations, including occlusions, reflections, challenging lighting conditions, and difficulties with small objects—all of which lead to inaccurate or incomplete geometric representations [1]. Neural representations have recently demonstrated exceptional capabilities in Novel View Synthesis (NVS), effectively handling many scenarios where conventional depth cameras struggle. Moreover, unlike traditional methods requiring pre-existing CAD models, neural representations can be generated directly from object images, facilitating the development of more adaptable and flexible robotic systems. While applying NVS techniques to improve robotic manipulation appears promising, their practical integration into functional robotic systems remains challenging and computationally demanding.

This thesis establishes a framework that addresses fundamental challenges limiting the application of neural representation techniques to robotics. First, we develop a method that reduces the number of views required for accurate neural object representation by combining traditional view morphing with neural rendering, thereby enhancing practicality for real-world robotic applications. Second, we introduce two complementary 6D pose estimation methods utilizing NVS representations that enable robots to accurately localize objects while preserving the advantages of neural representations. These methods facilitate direct pose validation by comparing query images and synthesized viewpoints. Finally, we enhance grasping precision for anthropomorphic robotic hands by incorporating novel view synthesis during grasp planning, allowing the system to assess potential grasps before execution. Extensive experimental evaluations substantiate the efficacy of our proposed methods across diverse localization and manipulation scenarios, demonstrating their potential to advance robotic perception capabilities.

**Keywords** [Robotics, Novel View Synthesis, Neural Rendering, 6D Pose Estimation, Grasping]

---

# Contents

|                                                                      |           |
|----------------------------------------------------------------------|-----------|
| <b>Abstract</b>                                                      | <b>i</b>  |
| <b>1 Introduction</b>                                                | <b>1</b>  |
| 1.1 The need for robots in real environments . . . . .               | 1         |
| 1.2 Aim of study . . . . .                                           | 2         |
| 1.2.1 Novel View Synthesis framework . . . . .                       | 2         |
| 1.2.2 6DoF pose estimation . . . . .                                 | 4         |
| 1.2.3 Grasp pose prediction . . . . .                                | 5         |
| 1.3 Deep Learning context and data-driven approaches . . . . .       | 6         |
| 1.4 Novel View Synthesis . . . . .                                   | 7         |
| 1.5 Robotic platforms . . . . .                                      | 10        |
| 1.6 Contribution . . . . .                                           | 11        |
| 1.6.1 List of publications . . . . .                                 | 12        |
| 1.7 Organization of the dissertation . . . . .                       | 13        |
| <b>2 Related works</b>                                               | <b>17</b> |
| 2.1 From Structure-from-Motion to NVS . . . . .                      | 17        |
| 2.1.1 Multi-View Stereo . . . . .                                    | 18        |
| 2.1.2 Neural Network for NVS . . . . .                               | 20        |
| 2.1.3 Neural Rendering . . . . .                                     | 21        |
| 2.2 Sparse View Novel View Synthesis . . . . .                       | 26        |
| 2.2.1 NeRF-based Sparse View Novel View Synthesis . . . . .          | 27        |
| 2.2.2 3DGS-based Sparse View Novel View Synthesis . . . . .          | 30        |
| 2.3 6DoF Pose Estimation using NVS . . . . .                         | 31        |
| 2.3.1 Early works on NeRF and Pose Refinement . . . . .              | 31        |
| 2.3.2 NeRF-based 6DoF Pose Estimation . . . . .                      | 32        |
| 2.3.3 3DGS-based 6DoF Pose Estimation . . . . .                      | 34        |
| 2.4 Robotic Grasping Prediction . . . . .                            | 35        |
| 2.4.1 Robotic end-effectors . . . . .                                | 36        |
| 2.4.2 Novel View Synthesis for Robotic Grasping Prediction . . . . . | 38        |
| 2.4.3 Dexterous Grasping . . . . .                                   | 42        |
| 2.4.4 Chapter conclusion . . . . .                                   | 43        |

|          |                                                                  |            |
|----------|------------------------------------------------------------------|------------|
| <b>3</b> | <b>Sparse View for Novel View Synthesis</b>                      | <b>45</b>  |
| 3.1      | View Morphing for Few-Viewpoint Neural Radiance Fields . . . . . | 45         |
| 3.1.1    | Approach . . . . .                                               | 47         |
| 3.1.2    | Results . . . . .                                                | 51         |
| 3.2      | Chapter conclusions . . . . .                                    | 56         |
| <b>4</b> | <b>6DoF Pose Estimation for Novel View Synthesis</b>             | <b>61</b>  |
| 4.1      | 6DoF using Neural Radiance Fields . . . . .                      | 62         |
| 4.1.1    | Approach . . . . .                                               | 65         |
| 4.1.2    | Results . . . . .                                                | 68         |
| 4.2      | 6DoF using 3D Gaussian Splatting . . . . .                       | 73         |
| 4.2.1    | Approach . . . . .                                               | 75         |
| 4.2.2    | Results . . . . .                                                | 79         |
| 4.3      | Chapter conclusions . . . . .                                    | 86         |
| <b>5</b> | <b>Robotic grasping</b>                                          | <b>89</b>  |
| 5.1      | Novel View Synthesis for dexterous grasping . . . . .            | 89         |
| 5.1.1    | Approach . . . . .                                               | 91         |
| 5.1.2    | GRASPLAT dataset . . . . .                                       | 95         |
| 5.1.3    | Results . . . . .                                                | 95         |
| 5.2      | Chapter conclusions . . . . .                                    | 99         |
| <b>6</b> | <b>Conclusions</b>                                               | <b>101</b> |
| 6.1      | Future Work . . . . .                                            | 102        |
|          | <b>Bibliography</b>                                              | <b>105</b> |

# List of Tables

|     |                                                                                                                                                                                                                                                                                                                                                                                                                                    |    |
|-----|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|----|
| 1.1 | Author contributions for the core publications of this thesis . . . . .                                                                                                                                                                                                                                                                                                                                                            | 14 |
| 3.1 | Results on the NeRF realistic synthetic 360° dataset. . . . .                                                                                                                                                                                                                                                                                                                                                                      | 52 |
| 3.2 | Ablation study results. Keys: # views: represent the number of views and the relative subset. avg. dist.: average distance between view pairs. . . . .                                                                                                                                                                                                                                                                             | 52 |
| 3.3 | SSIM ( $\uparrow$ ) metric per-scene results on the NeRF realistic synthetic 360° dataset. Best-performing results are highlighted in <b>bold</b> and second-best results are <u>underlined</u> . . . . .                                                                                                                                                                                                                          | 53 |
| 3.4 | LPIPS ( $\downarrow$ ) metric per-scene results on the NeRF realistic synthetic 360° dataset. Best-performing results are highlighted in <b>bold</b> and second-best results are <u>underlined</u> . . . . .                                                                                                                                                                                                                       | 57 |
| 4.1 | Evaluation of 6DoF pose estimation on two datasets: (a) Synthetic NeRF [2] and (b) Tanks & Temples [3]. We report the results in terms of Mean Angular Error (MAE) and Mean Translation Error (MTE). The lower MAE and MTE, the better. Best-performing results are highlighted in <b>bold</b> and second-best results are <u>underlined</u> . . . . .                                                                             | 70 |
| 4.2 | Evaluation of 6DoF pose estimation on the Mip-NeRF 360° [4] dataset. We report results in terms of Mean Angular Error (MAE) and Mean Translation Error (MTE) in terms of degrees and units, $u$ , respectively. Where $1u$ is equal to the object’s largest dimension. For both metrics lower is better. Best-performing results are highlighted in <b>bold</b> and green, while second best values are <u>underline</u> . . . . . | 80 |
| 4.3 | Evaluation of 6DoF pose estimation on the Tanks&Temples [3] dataset. We show the same metrics and analysis as in Table 4.2. . . . .                                                                                                                                                                                                                                                                                                | 81 |
| 4.4 | Ablation study on the number of rays selected for pose estimation using Mip-NeRF 360° [5]. <u>Underline</u> indicates the default values used. . . . .                                                                                                                                                                                                                                                                             | 83 |
| 4.5 | Ablation study on the number of rays cast from each ellipsoid using Mip-NeRF 360° [5]. <u>Underline</u> indicates the default values used. . . . .                                                                                                                                                                                                                                                                                 | 83 |
| 4.6 | Ablation study on the MLP channel feature size using Mip-NeRF 360° [5]. <u>Underline</u> indicates the default values used. . . . .                                                                                                                                                                                                                                                                                                | 84 |
| 4.7 | Ablation study on 6DGS robustness to noise in the Gaussian centers. Results show performance under different noise levels added to the ellipsoid centers on the Bonsai and Bicycle scenes. . . . .                                                                                                                                                                                                                                 | 84 |

|     |                                                                                                                                                                                                                                                                                               |    |
|-----|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|----|
| 4.8 | Ablation study on 6DGS performance with reduced viewpoints. Results demonstrate pose estimation accuracy when training with fewer camera views on the Bonsai and Bicycle scenes. . . . .                                                                                                      | 84 |
| 4.9 | Average computation time to estimate the pose of an image. Comparison between our method and state-of-the-art approaches across the two datasets. Best-performing results are highlighted in <b>bold</b> and second-best results are <u>underlined</u> . Time is reported in seconds. . . . . | 85 |
| 5.1 | We report the grasp accuracy, mean joint error, relative rotation error (RRE), translation error, where applicable, along with inference time. Best-performing results are highlighted in <b>bold</b> and second-best results are <u>underlined</u> . . . . .                                 | 97 |
| 5.2 | Ablation study on GRASPLAT’s robustness to input image noise. Results show mean joint error and percentage increase compared to the noise-free baseline. . . . .                                                                                                                              | 97 |
| 5.3 | Ablation study on GRASPLAT’s robustness to occlusions. We randomly occlude some part of the input image with patches of maximum 33 pixels, until a certain occlusion area is reached. Results show mean joint error and percentage increase compared to the occlusion-free baseline. . . .    | 99 |

# List of Figures

|     |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                    |    |
|-----|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|----|
| 1.1 | Conceptual illustration of Novel View Synthesis (NVS). Given a set of input images captured from different viewpoints (left), NVS algorithms reconstruct a 3D representation of the scene that can be rendered from arbitrary new viewpoints not present in the original capture (right). This process typically involves establishing correspondences between images, inferring scene geometry and appearance, and rendering techniques to generate photorealistic views from previously unseen perspectives. . . | 8  |
| 2.1 | Conceptual illustration of View Morphing [6], which enables the synthesis of novel views by interpolating between two reference images. This technique establishes correspondences between images and creates geometrically correct intermediate views. . . . .                                                                                                                                                                                                                                                    | 18 |
| 2.2 | Conceptual illustration of ray marching in Neural Radiance Fields (NeRF). Points are sampled along each ray, and the network predicts color and density at each point, which are then composed to render the final pixel color. Image source: [2] . . . . .                                                                                                                                                                                                                                                        | 22 |
| 2.3 | Visual representation of the Gaussian splatting process, showing how 3D Gaussians are projected onto the image plane. This approach, initially proposed in Elliptical Weighted Average (EWA) splatting and later adopted by 3D Gaussian Splatting, efficiently renders complex scenes by projecting 3D Gaussian primitives to 2D. Image source: [7] . . . . .                                                                                                                                                      | 25 |
| 2.4 | Comparison between generalizable sparse NeRF and per-scene sparse NeRF pipelines. A single generalizable network enables novel view synthesis across multiple test scenes without requiring per-scene fine-tuning, whereas per-scene models demand separate training for each individual scene. While per-scene models may lack generalization capabilities, their performance limitations are compensated by sophisticated training procedures that effectively constrain the optimization process. . . . .       | 27 |
| 2.5 | Examples of different robotic end-effectors: (a) Robotiq 2F-85 gripper, (b) Robotiq EPick vacuum gripper, (c) Schmalz needle gripper SNGi-AE, (d) Octopus-inspired soft gripper, and (e) Shadow Dexterous Hand. . .                                                                                                                                                                                                                                                                                                | 36 |

|     |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                 |    |
|-----|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|----|
| 2.6 | Examples of the same object (a mug) grasped with two different gripper types: (a) parallel and (b) anthropomorphic (or dexterous). While both successfully grasp the mug, the parallel gripper’s approach appears unnatural and limits functionality—common actions like pouring could potentially damage its circuits. In contrast, the anthropomorphic gripper’s grasp enables nearly all typical mug interactions, including pouring. Additionally, attempting to grasp the handle with a parallel gripper would likely result in an unstable grip, as it cannot apply sufficient force to prevent the mug from slipping. Image sources: Tom Buehler/MIT CSAIL (parallel gripper) and [8] (anthropomorphic gripper). . . . . | 37 |
| 3.1 | Given a set of known views (ground truth), View Morphing-NeRF (VM-NeRF) generates image transitions between views (morph) that can be effectively used to train a NeRF model in the case of few-shot view synthesis. Results are of a higher quality when VM-NeRF is used. . . .                                                                                                                                                                                                                                                                                                                                                                                                                                                | 46 |
| 3.2 | Block diagram of NeRF-based View Morphing (VM-NeRF). From the left, we (1) predict the depth with NeRF, (2) rectify the input images and predicted depths, and (3) compute the image morphing of a view randomly positioned between the view pair. $\alpha$ determines the new view position and it is sampled from a Gaussian distribution. . . . .                                                                                                                                                                                                                                                                                                                                                                            | 48 |
| 3.3 | Cases where the algorithm in [9] fails to produce rectification results that are consistent with the original camera configuration. . . . .                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                     | 49 |
| 3.4 | Comparisons on test-set views of scenes of NeRF realistic synthetic 360°. Unlike AugNeRF [10], VM-NeRF is an effective method that can be used for few-shot view synthesis problems. Unlike DietNeRF [11], VM-NeRF enables NeRF to learn scenes with a higher definition. VM-NeRF helps to produce fewer artifacts than RegNeRF during rendering [12]. We report the PSNR that we measured for each method and for each rendered image. AugNeRF unsuccessfully learns Chair and Lego (white and black outputs). . . . .                                                                                                                                                                                                         | 54 |
| 3.5 | Comparisons on test-set views of scenes of NeRF realistic synthetic 360°, trained on 8-views. Unlike AugNeRF [10], VM-NeRF is an effective method that can be used for few-shot view synthesis problems. Unlike DietNeRF [11], VM-NeRF enables NeRF to learn scenes with a higher definition. VM-NeRF helps to preserve high-frequency details and geometric consistency. We report the PSNR that we measured for each method and for each rendered image. . . . .                                                                                                                                                                                                                                                              | 55 |
| 3.6 | Comparisons on test-set views of scenes of NeRF realistic synthetic 360°, trained on 4-views. Unlike other techniques VM-NeRF help to increase the probability of building a consistent implicit representation with very sparse view. We report the PSNR that we measured for each method and for each rendered image. . . . .                                                                                                                                                                                                                                                                                                                                                                                                 | 58 |



|     |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                           |    |
|-----|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|----|
| 3.7 | Comparisons on test-set views of scenes of NeRF realistic synthetic 360°, trained on 4-views. Unlike other techniques VM-NeRF help to increase the probability of building a consistent implicit representation with very sparse view. We report the PSNR that we measured for each method and for each rendered image. . . . .                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                           | 59 |
| 4.1 | From (i) a given image with an unknown pose and (ii) a NeRF model, we recover the pose by first (iii) sampling surface points using Metropolis-Hasting algorithm and (iv) casting rays from them in isocell distribution. We then (iv/v) correlate rays with the image to identify relevant rays using attention and (vii) recover the unknown 6DoF camera pose. . . .                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                    | 63 |
| 4.3 | Illustration of the Isocell ray generation method over a circular domain of a unit disk and unit sphere. The generated points indicate the ray positions within the equally spaced circle cells (we will denote them as “cell centres” for simplicity). . . . .                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                           | 66 |
| 4.4 | Example of generated rays using our approach. The zoomed region highlights the ray cast operation (in this case 27 rays per isocell). . . . .                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                             | 67 |
| 4.5 | Example of the test-time pose estimation on the Chair object of Synthetic NeRF. (a) Top $N$ (red) <i>vs.</i> top $N$ ground-truth (green) rays. (b) Corresponding score distribution of top $N$ and ground-truth rays. . . .                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                              | 69 |
| 4.7 | Our 6DGS method introduces a novel approach to 6DoF pose estimation, departing from conventional analysis-by-synthesis methodologies. Standard NeRF-based methods ( <b>left</b> ) employ an iterative process, rendering candidate poses and comparing them with the target image before updating the pose, which often results in slow performance and limited precision. In contrast, 6DGS ( <b>right</b> ) estimates the camera pose by selecting a bundle of rays projected from the ellipsoid surface (a <i>radiant Ellicell</i> ) and learning an attention map to output ray/image pixel correspondences (based on DINOv2). The optimal bundle of rays should intersect the optical center of the camera and then are used to estimate the camera rotation in closed-form. Our 6GDS method offers significantly improved accuracy and speed, enabling the recovery of the pose within a one-shot estimate. . . . . | 74 |
| 4.8 | The figure illustrates the pipeline of our 6DGS methodology. The image is encoded using a visual backbone ( <b>a</b> ). Concurrently, rays are uniformly projected from the center of the 3DGS ellipsoids ( <b>b</b> ), and their corresponding color is estimated. Subsequently, an attention map mechanism is employed to compare the encoded ray and image features ( <b>c</b> ). Following this comparison, the $N_{top}$ matches are selected via attenuation, and the camera location is estimated ( <b>d</b> ) as the solution of a weighted Least Squares problem, resulting in a distinct 6DoF pose for the image. . . .                                                                                                                                                                                                                                                                                         | 75 |

- 4.9 The illustration depicts the three primary stages involved in the radiant Ellicell generation. Firstly, **(a)** delineates the formulation of components required to compute the geometric information for each cell. Secondly, **(b)** shows the resulting Ellicell grid positioned on the surface of the ellipsoid along with their respective center points. Finally, **(c)** demonstrates the generation of rays originating from the center point of the ellipsoid going through the Ellicell center. . . . . 76
- 4.10 The illustration presents qualitative results from Tanks & Temple (upper row) and Mip-NeRF 360° (lower row) datasets. Each scene showcases the target images (bottom left) along with their corresponding Novel View Synthesis (NVS) outputs (bottom right), derived from the camera poses estimated by 6DGS (located on the top). Furthermore, the estimated camera poses from the comparative baselines are visualized, with distinct colors as indicated in the image legend. The NVS of each scene is rendered based on the provided 3DGS model. Please check the supplementary material for more qualitative results. . . . . 82
- 4.11 Additional scenes from the Mip-NeRF 360° dataset. For each scene, we show a visualization of the camera poses in regards to the model (top) for 6DGS as well as the baselines, which are visualized with different colors as indicated in the image legend. In addition, for each scene, we showcase the target image (bottom left) along with their corresponding 3DGS Novel View Synthesis (NVS) output (bottom right) of the estimated camera pose by 6DGS. . . . . 87
- 4.12 Additional scenes from the Tanks&Temple dataset. For each scene, we show a visualization of the camera poses in regards to the model (top) for 6DGS as well as the baselines, which are visualized with different colors as indicated in the image legend. In addition, for each scene, we showcase the target image (bottom left) along with their corresponding Novel View Synthesis (NVS) output (bottom right) of the estimated camera pose by 6DGS. . . . . 88
- 5.1 GRASPLAT predicts 3D hand poses from a single RGB image using a frozen, denoised DINOv2 backbone. The extracted feature map is processed by a spatial transformer to estimate hand parameters, which are then refined by an MLP. During training, a feedback mechanism based on 3D Gaussian Splatting (3DGS) refines predictions by comparing rendered and real images. To accelerate training, 3DGS models are precomputed before training and then cached. The rendered scene includes both the object and the articulated hand, posed according to the predicted parameters. . . . . 91

|     |                                                                                                                                                                                                                                                                                                                                                                                    |     |
|-----|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-----|
| 5.2 | Comparison of methods on the GRASPLAT (top) and YCB (bottom) datasets. Each row represents a different object. For each of the compared methods, as well as for GRASPLAT, we present the results in the simulated environment. Additionally, for our method, we also show qualitative image synthesis of our 3DGS model during training against the relative ground truth. . . . . | 98  |
| 5.3 | Comparison of loss trends between GRASPLAT and an LLM, in this case Llama [13], note the similarity in the convergence patterns. . . .                                                                                                                                                                                                                                             | 100 |



# Chapter 1

## Introduction

### 1.1 The need for robots in real environments

Fully autonomous robots have long been considered a pinnacle achievement in the field of robotics [14–17]. Their primary objective is to assist humans in performing tedious and repetitive tasks, spanning both industrial and domestic environments [17,18]. While industrial robots are already deployed in manufacturing settings, programming these systems remains a complex and specialized task requiring significant expertise [19]. This programming complexity and requirement for specialized expertise substantially limits their application to highly structured and controlled environments, such as production lines, where tasks are predictable and exhibit minimal variation [18,19].

To illustrate this limitation, consider the assembly of eyeglass frames, a process that exemplifies the challenges in flexible manufacturing [20–22]. Manufacturers typically produce multiple models of eyeglass frames, often in small batches of no more than 1000 units per model due to economic constraints and market demands [23]. Although these models may differ substantially in design and shape, their assembly processes remain largely similar across variants [20,23]. Human workers can seamlessly switch between assembling different models or adapt to new designs with minimal training, whereas traditional industrial robots require extensive reprogramming for each model variation, making their deployment economically impractical in such flexible production settings [24]. Consequently, automating assembly in flexible production lines necessitates the development of robotic systems capable of high-level reasoning and adaptability to handle product variations without constant reprogramming [25].

This need for adaptive robotics extends beyond industrial settings and is equally critical in domestic environments, where tasks vary significantly and environments are inherently unstructured [17,26]. Everyday household activities such as cleaning, cooking, and providing assistance to senior citizens could greatly benefit from autonomous robots capable of understanding and adapting to diverse scenarios [27]. In these complex domestic contexts, humanoid robots are particularly promising candidates for widespread adoption, as their anthropomorphic design naturally facilitates seamless interaction with environments specifically built for human use and efficient manipulation [26]. Furthermore, their intuitive human-like appearance can foster a deeper sense of familiarity

and emotional comfort, making human-robot interaction substantially more intuitive and accessible, particularly for individuals less accustomed to modern technology, such as many senior citizens with limited tech exposure [28].

Industrial and domestic applications both require robots to possess sophisticated object manipulation capabilities, which constitute a core challenge in robotics research [17, 18]. Object manipulation is inherently complex, as it requires understanding the physical properties of the object, the environment around it, and the specific requirements of the task at hand [29, 30]. While some of these tasks, such as pick&place require less precision, others, such as assembly or portioning, demand a high level of dexterity and precision [18, 30, 31]. The first fundamental step in object manipulation is grasping, an action that humans perform effortlessly but remains highly challenging for robotic systems to execute reliably across diverse objects [29, 30].

The conventional approach to grasping involves decomposing the problem into several sequential sub-tasks, each presenting unique challenges [29]. The first critical step is understanding the object to be grasped, which typically requires reconstructing an accurate three-dimensional model of the object from sensor data [32]. Once reconstructed, the object must then be precisely localized in the robot’s operational environment, a process formally known as 6D pose estimation (determining position and orientation in the physical space) [33]. Finally, the robot must accurately predict the optimal grasp pose based on the object’s geometry and physical properties, and execute the grasp accordingly using available end-effectors and control algorithms [29, 30].

Traditionally, object reconstruction has been performed using depth cameras, which directly measure distances between objects and the camera sensor [34]. Although effective in many scenarios with textured and opaque objects, depth cameras face significant difficulties when dealing with objects that have reflective, transparent, or highly specular surfaces—such as eyeglass lenses, glassware, and certain coated materials [1]. These sensing limitations substantially impact the applicability of conventional depth-based approaches in both industrial settings (e.g., assembling eyeglasses with reflective components) and domestic environments (e.g., handling transparent objects, metal pots, or shiny kitchenware) [35].

## 1.2 Aim of study

This dissertation aims to establish a framework for grasping that addresses the limitations of traditional sensors by exploring alternative approaches that utilize 3D models generated through Novel View Synthesis (NVS) techniques [2]. Establishing this framework requires the integration of NVS, robust to real-world conditions (such as few-viewpoints settings) with algorithms that leverage them to predict object poses and predicting feasible grasp configurations.

### 1.2.1 Novel View Synthesis framework

NVS involves generating photorealistic views of an object from new perspectives based on a set of multiple RGB images captured from known viewpoints. In practice, these

camera poses are often estimated using Structure from Motion (SfM) techniques, that combine epipolar geometry for initial pose estimation with bundle adjustment for refinement [36]. SfM recovers camera pose  $\hat{\mathcal{P}}_i \in \text{SE}(3)$  and 3D structure as points  $\hat{\mathbf{X}} \in \mathbb{R}^{3 \times S}$  by solving:

$$\{\hat{\mathbf{P}}_i, \hat{\mathbf{X}}_j, \hat{\mathbf{K}}_i\} = \underset{\{\mathbf{P}_i, \mathbf{X}_j\}}{\operatorname{argmin}} \sum_{i=1}^N \sum_{j \in \mathcal{V}_i} \left\| \mathbf{x}_{ij} - \pi(\hat{\mathbf{P}}_i, \hat{\mathbf{X}}_j, \hat{\mathbf{K}}_i) \right\|^2, \quad (1.1)$$

where  $\mathbf{x}_{ij} \in \mathbb{R}^2$  denotes the observed 2D pixel coordinates of 3D point  $\hat{\mathbf{X}}_j$  in image  $i$ ,  $\mathcal{V}_i$  is the set of 3D points visible in image  $i$ , and  $\pi : \text{SE}(3) \times \mathbb{R}^3 \times \mathbb{R}^{3 \times 3} \rightarrow \mathbb{R}^2$  is the perspective projection function mapping 3D points to 2D image coordinates given camera pose and intrinsics. The camera poses  $\hat{\mathcal{P}} = \{\hat{\mathbf{P}}_1, \hat{\mathbf{P}}_2, \dots, \hat{\mathbf{P}}_N\}$  and intrinsics  $\hat{\mathcal{K}} = \{\hat{\mathbf{K}}_1, \hat{\mathbf{K}}_2, \dots, \hat{\mathbf{K}}_N\}$  estimated by SfM then serve as known input parameters for NVS. Throughout this thesis, the hat symbol ( $\hat{\cdot}$ ) is used to denote predicted or estimated elements, distinguishing them from ground-truth or observed values. Given a set of  $N$  images  $\mathcal{I} = \{\mathbf{I}_1, \mathbf{I}_2, \dots, \mathbf{I}_N\}$ , the corresponding camera poses  $\hat{\mathcal{P}}$  and intrinsics  $\hat{\mathcal{K}}$ , NVS optimizes a scene representation  $f_{\Xi}$  to render photorealistic images from arbitrary novel viewpoints. Each camera pose  $\hat{\mathbf{P}}_i$  represents the rigid transformation from the world coordinate frame to the camera coordinate frame. The NVS optimization objective can be formulated as:

$$f_{\Xi} = \underset{\Xi}{\operatorname{argmin}} \sum_{i=1}^N \mathcal{L}_{\text{photo}} \left( \mathbf{I}_i, \hat{\mathbf{c}}(f_{\Xi}, \hat{\mathbf{P}}_i, \hat{\mathbf{K}}_i) \right), \quad (1.2)$$

where  $\hat{\mathbf{c}}(\cdot)$  denotes the rendering function that generates an image from the scene representation  $f_{\Xi}$  given camera parameters,  $\mathcal{L}_{\text{photo}}$  measures the photometric reconstruction error (typically L2 or perceptual loss). Although NVS was not originally developed specifically for precise 3D reconstruction, recent research advancements have demonstrated that NVS and 3D reconstruction are interdependent processes that can enhance each other [37]. Improved 3D reconstructions can significantly improve the performance of NVS, and conversely, better NVS models can lead to more accurate 3D reconstructions [38, 39]. This synergistic relationship is particularly valuable for challenging objects with reflective, transparent, or highly specular surfaces that confound traditional reconstruction methods [40]. Unfortunately, there is a deep gap between humans and machines. Humans can imagine object’s shape just from a few viewpoints, and then grasp it based on this mental image [41]. Instead, most NVS methods require hundreds of images to generate accurate 3D reconstructions, making them impractical for real-time robotic applications where data collection must be efficient [16]. To overcome this crucial limitation, a focus of this thesis involves developing a novel method for generating accurate 3D reconstructions from a limited number of RGB images, making the approach feasible for robotic deployment.

### 1.2.2 6DoF pose estimation

Once an object has been successfully reconstructed, accurately estimating its pose in a new environment becomes essential for successful manipulation [29, 30]. The task of object 6 Degrees-of-Freedom (6DoF) pose estimation involves precisely matching an object detected within the robot’s visual scene to a known 3D model of that object. In conventional industrial applications, Computer-Aided Design (CAD) models provided by manufacturers are typically used as the reference 3D model for pose estimation [42]. However, in domestic settings and many flexible manufacturing scenarios, precise CAD models are often unavailable for the diverse objects the robot must manipulate, rendering conventional pose estimation techniques impractical. The 6DoF pose estimation problem presents a fundamental challenge: it is an inverse problem where multiple solutions may appear visually similar, yet only a geometrically precise estimate enables successful manipulation. Unlike image classification where approximate predictions can suffice, manipulation requires millimeter-level precision—errors of even a few millimeters or degrees can cause grasp failures. We can define 6DoF pose estimation as follows: given a query image  $\mathbf{I}_{\text{query}} \in \mathbb{R}^{H \times W \times 3}$  and a pre-trained scene representation  $f_{\Xi}$  as input, the objective of 6DoF pose estimation is to recover the camera pose  $\hat{\mathbf{P}}_{\text{pred}} \in \text{SE}(3)$  that minimizes the error with the ground truth pose—which simultaneously determines the object’s pose in the camera frame through the inverse transformation. It is crucial to note that object pose and camera pose represent a dual perspective of the same spatial relationship: estimating where an object is relative to the camera is mathematically equivalent to determining where the camera is positioned relative to the object’s coordinate frame. This camera-object duality can be formally expressed as:

$$\mathbf{P}_{\text{obj}}^{\text{cam}} = (\mathbf{P}_{\text{cam}}^{\text{obj}})^{-1}, \quad (1.3)$$

where  $\mathbf{P}_{\text{obj}}^{\text{cam}} \in \text{SE}(3)$  represents the object’s pose in the camera coordinate frame, and  $\mathbf{P}_{\text{cam}}^{\text{obj}} \in \text{SE}(3)$  denotes the camera’s pose in the object coordinate frame. This duality is particularly relevant when using NVS models as reference representations, since these models are inherently parameterized by camera poses during training. This thesis proposes two novel techniques for estimating the 6DoF pose of objects using NVS-generated models instead of relying on pre-existing CAD models, thereby expanding the applicability of robotic manipulation systems. Both methods leverage this camera-object duality to naturally formulate the pose estimation problem in terms of camera poses, which aligns with the NVS model parameterization. Additionally, most methods for 6D pose estimation heavily rely on depth information, which makes them unsuitable for many everyday objects [40, 42], including objects with glossy surfaces, reflections, and hit by solar glares. In contrast, the methods presented in this thesis rely primarily on standard RGB images for processing and reconstruction, effectively eliminating the fundamental limitations associated with depth sensors when dealing with optically complex objects.



### 1.2.3 Grasp pose prediction

The critical step is the grasp pose prediction, which determines how the robot should position its end-effector to successfully grasp the target object. Depending on the specific requirements of the task and the type of robotic end-effector available, different approaches to grasp planning can be employed [29, 30, 43]. For instance, in precisely defined assembly tasks, grasp poses are typically predetermined based on the detailed 3D model of the object and the specific capabilities of the end-effector [18]. In contrast, more general applications such as warehouse pick&place and diverse domestic tasks, the grasp pose must be dynamically predicted based on both the object’s shape and the intended subsequent use of the object [32, 44]. Dexterous grasping presents a high-dimensional mapping problem: from visual observations (millions of pixels) to motor commands (dozens of joint angles), where the solution space is vast but only a small subset enables stable grasps. This challenge is amplified by the long-tailed distribution of object shapes encountered in unstructured environments, requiring systems to generalize beyond their training data. Mathematically we define it as: given an RGB image  $\mathbf{I} \in \mathbb{R}^{H \times W \times 3}$  depicting a target object, the objective is to estimate an end-effector configuration that enables successful object manipulation. This configuration can be parameterized as  $\hat{\mathbf{P}}_{\text{grasp}} + \hat{\mathbf{h}}$ . Here,  $\hat{\mathbf{P}}_{\text{grasp}} \in \text{SE}(3)$  denotes the global pose of the robotic end-effector, and  $\hat{\mathbf{h}} \in \mathbb{R}^{J \times \text{SO}(3)}$  encodes the relative rotations of  $J$  articulated joints with respect to their respective parent links.

Grasp prediction models are often specifically tailored to particular end-effectors, which vary significantly in their mechanical design and degrees of freedom [45–48]. These end-effectors range from simple suction-based grippers, which can only handle objects with flat surfaces suitable for vacuum attachment, to highly complex human-like robotic hands with multiple articulated fingers capable of grasping a wide variety of objects with different shapes and physical properties [18]. While traditional approaches using hand-crafted geometric features can be employed for grasp prediction in controlled scenarios, data-driven learning approaches have proven to be significantly more effective at generalizing across diverse object categories [30]. Although these deep learning models tend to generalize better to novel objects not seen during training, they often struggle with precision, especially when dealing with high-DoF end-effectors such as anthropomorphic hands where finger positioning must be carefully coordinated [49].

To address this critical limitation in grasp prediction, this thesis introduces an innovative method that leverages NVS to visualize and systematically optimize grasp poses for human-like robotic end-effectors with multiple degrees of freedom. By integrating NVS capabilities into the grasp prediction process, this approach significantly enhances both the accuracy and adaptability of robotic grasping in unstructured environments where objects and scenarios may vary substantially. The developed method enables the robot to better anticipate the outcome of potential grasps by visualizing them before execution, thereby reducing the likelihood of failed grasp attempts.

The remainder of this chapter is organized as follows. After providing essential background on Deep Learning approaches, which serves as prerequisite for understanding most of the methods proposed in this thesis, we present an overview of NVS techniques

and their evolution over time. We then analyze in detail the principal types of robotic platforms relevant to our research context and their manipulation capabilities in various environments. The last section discusses the core technical contributions of this dissertation and summarizes the complete structure of the rest of the thesis.

## 1.3 Deep Learning context and data-driven approaches

Most of the methods proposed in this thesis rely on data-driven techniques [50]. Deep Learning is a subfield of Machine Learning that focuses on automatically detecting unique characteristics or patterns in data to facilitate the separation of new, unseen data. While the distinction between Machine Learning and Deep Learning is not always clear-cut [51, 52], a key differentiating factor is the number of parameters involved and the use of backpropagation to compute updates for these parameters. Deep Learning models typically employ significantly more parameters than traditional Machine Learning approaches, allowing them to learn more complex representations. Originally developed for image processing in the form of Convolutional Neural Networks [53], Deep Learning has since been applied to a wide range of problems, including Natural Language Processing and robotics [54, 55], particularly in the context of Reinforcement Learning [56]. The versatility of Deep Learning stems from its ability to automatically extract hierarchical features from raw input data, eliminating the need for manual feature engineering. One of the most significant and revolutionary advancements in Deep Learning research is the Transformer architecture, which was originally introduced for Natural Language Processing tasks [57].

Transformers rely on a mechanism called self-attention, which dynamically weighs the importance of different parts of the input data embedding. This mechanism enables the model to capture long-range dependencies and relationships within the data more effectively than previous architectures such as Recurrent Neural Networks (RNNs). The Transformer architecture consists of an encoder and a decoder, both composed of multiple layers of self-attention and feed-forward neural networks. The encoder processes the input data and generates a set of representations, called embeddings, which are then used by the decoder to produce the output. In contrast to sequential models like RNNs, Transformers process all input elements in parallel, significantly improving computational efficiency. Furthermore, Transformers are input-size independent, making them particularly useful in contexts where input sizes vary. This includes NLP tasks, where sentence lengths can differ significantly, point cloud data, where the number of points fluctuates, and image processing tasks, where resolution may change. The parallelizable nature of Transformers, combined with their ability to model complex dependencies, has led to their widespread adoption across various domains.

Recently, the Transformer architecture has been adapted for Computer Vision tasks through the development of the Vision Transformer (ViT) [58]. The ViT model treats an image as a sequence of patches, similar to how words are processed in Natural Language Processing tasks. Each image is divided into fixed-size patches, which are then linearly embedded and combined with positional embeddings to retain spatial information. These patch embeddings are fed into a standard Transformer encoder,

which processes the sequence of patches and generates a representation for the entire image. This approach differs fundamentally from Convolutional Neural Networks (CNNs), which build representations through local convolution operations and pooling layers. ViT has demonstrated remarkable performance on various Computer Vision benchmarks, often surpassing traditional CNN-based models. One of the key advantages of ViT is its ability to capture global context and long-range dependencies within an image, which is particularly beneficial for tasks that require understanding complex spatial relationships. Additionally, ViT models can be pre-trained on web-scale image datasets, similar to how large language models are trained in Natural Language Processing. This pre-training process enables the model to learn rich visual representations, which can then be fine-tuned on specific downstream tasks. Between these methods, it is worth mentioning, DINO [59] that employs a teacher-student architecture where both networks process different augmented views of the same image. This self-distillation process enables the model to learn meaningful visual representations without requiring labeled data. DINOv2 [60] extends this concept by leveraging a mix of supervised and unsupervised data, improving performance on various tasks. Several approaches proposed in this thesis leverage DINOv2’s representations, and the underline ViT architecture, extending beyond black-box applications to enhance downstream task performance through novel architectural modifications.

Beyond image-based methods, Deep Learning has also been extended to other types of data, such as Point Clouds [61, 62] by creating specialized network architectures. These architectures address the unique challenges posed by unordered, irregular point cloud data through permutation-invariant operations. Notably, SIREN [63] demonstrated that it is possible to use neural networks with sinusoidal activation functions to represent continuous 3D scenes with high fidelity. These networks can capture fine details and complex geometric structures that are difficult to model with traditional representations. Building upon this concept, neural networks can be utilized to represent both the 3D world and its appearance, as seen in applications like NVS, which will be discussed in the next section.

## 1.4 Novel View Synthesis

Novel View Synthesis (NVS) has significantly evolved over the past few decades, driven by advancements in computer vision, graphics, and machine learning [64, 65]. The primary goal of NVS is to generate novel images of a scene or object from viewpoints not present in the original dataset. This capability enables users to virtually navigate around objects or scenes by generating realistic renderings from arbitrary perspectives. A visual representation of the NVS concept is shown in Fig. 1.1. This technology is crucial for applications such as virtual and augmented reality, where realistic scene rendering from arbitrary viewpoints is essential for creating immersive experiences [66]. Additional important applications include telepresence, autonomous navigation, and content creation for entertainment industries.

Early approaches to NVS, such as View Morphing [6], focused on interpolating between existing views to generate novel perspectives. These methods functioned by



Figure 1.1: Conceptual illustration of Novel View Synthesis (NVS). Given a set of input images captured from different viewpoints (left), NVS algorithms reconstruct a 3D representation of the scene that can be rendered from arbitrary new viewpoints not present in the original capture (right). This process typically involves establishing correspondences between images, inferring scene geometry and appearance, and rendering techniques to generate photorealistic views from previously unseen perspectives.

establishing correspondences between key points in different images and calculating intermediate viewpoints through geometric transformations. While effective for simple transitions between nearby viewpoints, these methods required manual specification of correspondences between input views, significantly limiting their practical usability. Furthermore, they were unsuitable for synthesizing views from arbitrary angles that deviated substantially from the input views. This limitation was partially addressed by techniques for reconstructing explicit 3D models of a scene or object, such as Structure from Motion (SfM) [36, 67] and Multi-View Stereo (MVS) [68].

MVS complements SfM by densifying initial sparse reconstruction, estimating depth by computing robust correspondences between multiple views [68]. Given the camera poses and the depth of a sparse set of pixels, MVS algorithms infer the depth of the remaining pixels, producing a comprehensive 3D representation in the form of a dense point cloud or mesh [69]. Traditional rendering techniques, such as ray tracing or rasterization, can then be used to generate novel views from these 3D reconstructions [70]. However, these geometry-based methods are computationally expensive and highly dependent on the scene’s characteristics [71]. Similar to depth cameras, they struggle with reflective, transparent, or highly specular surfaces, which frequently lead to reconstruction errors or holes in the resulting models [72]. While significant progress has been made in this field over the years, unlike View Morphing, most modern NVS methods still require accurate camera poses as input to function effectively.

With the advent of deep learning, NVS techniques shifted towards more powerful learning-based approaches [73]. Deep neural networks were trained to implicitly capture the 3D structure of a scene from a set of input images, enabling more flexible novel view generation [74]. These approaches typically consist of an encoder-decoder architecture that processes input images and camera parameters to synthesize novel views, like the one presented before in Sec. 1.3. However, these methods often struggled with generalization, particularly when dealing with complex objects and scenes outside their

training distribution. Additionally, these early deep learning methods were prone to artifacts and distortions, especially when generating views significantly different from the training images, due to the lack of explicit 3D representations [2]. The absence of geometric constraints made it difficult for these networks to maintain consistent 3D structure across different viewpoints.

To overcome these limitations, researchers explored methods that explicitly model the 3D structure of a scene while leveraging the representational power of neural networks. One of the first breakthroughs in this direction was the representation of scenes as 3D volumes, which could then be rendered using an upsampling Convolutional Neural Network (CNN) [75]. These volumetric approaches divided the 3D space into discrete voxels, with each voxel storing appearance information that could be accessed during rendering. While these approaches mitigated issues related to geometric distortions, their generalization capability remained limited due to the vast amount of training data required. Furthermore, the memory requirements for high-resolution voxel grids posed significant practical challenges for complex scenes.

In parallel, Sinusoidal Representation Networks (SIREN) [63] demonstrated that instead of using a single neural network for all scenes, a separate network could be trained for each scene to represent its specific properties. This approach inherently avoided the generalization issues faced by prior methods by focusing on accurately modeling a single scene or object. Moreover, SIREN showed that representing a 3D object with a neural network was significantly more memory-efficient than traditional meshes or point clouds for complex geometries. By storing surface details parametrically through the weights of neural networks, these neural representations provided smoother interpolations than triangular meshes, which are inherently piecewise linear. Additionally, they dynamically allocated computational resources, using higher memory for intricate object details and lower memory for flat surfaces, making them adaptable to varying levels of scene complexity and efficiently capturing high-frequency spatial details.

SIREN originally required ground truth (GT) geometry for training, limiting its applicability to scenarios where 3D data was already available. Neural Radiance Fields (NeRF) [2] removed this requirement by modifying the network architecture to predict both geometry and appearance simultaneously from only 2D images. NeRF introduced ray marching, a rendering technique that samples multiple points along a ray passing through the scene from the camera position through each pixel. The network then predicts the color (RGB values) and density (opacity) of each sampled point in 3D space, and the final rendered color is obtained through volume rendering, which integrates these values along the ray. During training, the predicted color is compared against the ground-truth image using photometric loss functions, allowing the network to learn the scene’s structure without requiring explicit geometry supervision. Since this process is fully differentiable, the underlying neural network can be trained end-to-end using gradient-based optimization methods.

Despite its effectiveness in producing high-quality novel views, NeRF is computationally expensive due to the large number of network interrogations required per ray (typically hundreds of samples) [76–78]. This computational burden results in slow training times (often days for complex scenes) and rendering speeds that fall

short of real-time requirements for interactive applications. This issue was largely addressed with the introduction of 3D Gaussian Splatting (3DGS) [78], which represents a paradigm shift in neural rendering. Instead of representing a scene as a neural network, 3DGS models it as a set of 3D Gaussian distributions, where each Gaussian is centered at a point in space and optimized for shape (through its covariance matrix), appearance (color and opacity), and position during training. A dedicated rasterization-based renderer is then used to synthesize images, replacing the costly ray marching process used in NeRF-based methods.

Rasterization, a widely used technique in computer graphics, projects scene elements onto the image plane and determines their visibility based on depth ordering and blending operations. Unlike ray marching, which queries the 3D space along each ray at numerous positions, rasterization directly renders visible elements, making it significantly more efficient. This efficiency stems from hardware acceleration available in modern GPUs, which are optimized for rasterization operations. By leveraging this approach, 3DGS achieves high-quality novel view synthesis with substantially lower computational overhead, enabling real-time rendering rates even for complex scenes. The explicit nature of the Gaussian primitives also allows for easier scene editing compared to implicit neural representations.

The methods presented in this thesis are built upon both NeRF and 3DGS to enhance the key tasks useful for robotics. The following section describes our contributions that extend these foundational techniques for robotics applications.

## 1.5 Robotic platforms

In the industrial world, robotic arms are the most common robotic platforms [79]. These robots are typically used for tasks such as assembly, welding, painting, and material handling [18]. They are highly precise and can be programmed to perform repetitive tasks with great accuracy [80, 81]. Robotic arms are often mounted on fixed bases and have multiple joints, allowing for a wide range of motion [82]. The kinematic structure of these arms typically consists of a series of rigid links connected by rotational joints, enabling six degrees of freedom. These robots are equipped with various end-effectors, such as grippers, suction cups, or welding torches, depending on the specific task they are designed to perform [18]. The selection of an appropriate end-effector is crucial for optimizing task performance and efficiency in industrial settings. Despite their high precision, a major drawback of robotic arms is their rigidity in programming [19]. While they can execute pre-defined movements with extreme accuracy, these movements cannot be easily altered in response to environmental changes or variations in workpieces. This limitation necessitates the use of support structures and fixtures to ensure that the workpieces remain in a fixed position during operations [83]. Such fixtures often require precise engineering and contribute significantly to manufacturing lead times and costs. As a result, adapting to different product sizes or making quick transitions between manufacturing processes becomes challenging and resource-intensive. Consequently, it is essential to develop methods that enhance the flexibility and adaptability of these robots with minimal programming effort and hardware modifications.

In domestic and industrial settings, humanoid robots are expected to become increasingly prevalent in the coming decades [17]. These robots are designed to assist with household chores, provide companionship, and support senior citizens individuals and those with disabilities [84]. The demographic shift toward aging populations in many developed countries has accelerated research into assistive robotics for eldercare applications. In nursing homes, humanoid robots can help with tasks such as medication reminders, mobility assistance, and social interaction, thereby improving the quality of life for senior citizens. Their anthropomorphic design makes interaction more intuitive for humans, facilitating seamless communication and task execution [17]. The human-like appearance and movement patterns of these robots reduce the cognitive load required for users to understand and predict their behaviors.

For humanoid robots to be effective in domestic environments, they must be capable of autonomous navigation in dynamic spaces, object recognition and manipulation in unstructured environments, and natural interaction through speech and gestures [26]. These capabilities require sophisticated perception systems, robust decision-making algorithms, and adaptive control strategies. However, developing systems that allow such levels of intuitive interaction remains an open research challenge. Particularly challenging is the development of reliable object manipulation skills in varied and unpredictable domestic settings. It is important to note that certain tasks, such as object pose recognition and manipulation, are fundamental to both industrial and domestic robotics. The ability to accurately perceive, grasp, and manipulate objects of different shapes, sizes, and material properties is a prerequisite for many applications across robotic domains.

This thesis will explore these shared challenges and propose innovative solutions to bridge the gap between rigid industrial automation and adaptive domestic robotics. The proposed approaches aim to enhance robot flexibility while maintaining the precision required for practical applications.

## 1.6 Contribution

This thesis contributes several novel methods for enhancing robotic perception and manipulation capabilities using NVS techniques. The main contributions are:

1. A novel approach that combines View Morphing with NeRF to overcome the challenge of sparse viewpoints in NVS [85]. This method enables 3D reconstruction and NVS from a limited number of input images, addressing a critical limitation in robotics applications where acquiring numerous images of an object may be impractical. The integration of traditional View Morphing techniques with modern NeRF provides a solution that efficiently leverages the complementary strengths of both approaches. Experiments demonstrate that our method reduces the required number of input views up to 4 compared to standard NeRF implementations while maintaining comparable high-quality reconstruction quality.
2. An initialization-free and fast method for 6DoF pose estimation from a single image using a pre-trained NeRF model [86]. This technique eliminates the need

for explicit depth information or Computer-Aided Design (CAD) models, making it particularly valuable for objects with reflective, transparent, or highly specular surfaces where traditional depth sensors fail. By leveraging the implicit 3D representation encoded within NeRF, our approach can accurately predict object poses even with challenging objects such as glass or translucent surface. Our experiments show an average angular error reduction of 80.1% and translational error reduction of 67.3% compared to existing single-image pose estimation techniques.

3. A method for 6DoF pose estimation using 3D Gaussian Splatting models, which improve accuracy in real-world scenarios, while maintain near real-time speed [87]. By reformulating the pose estimation problem within the Gaussian Splatting framework, we achieve a 38x speedup compared to previous Gaussian-based methods while increasing accuracy. The method demonstrates robust performance across diverse object categories and challenging environmental conditions, reducing rotation error by 12.0% and translation error by 22.0% in real-world scenarios.
4. A novel approach that leverages 3D Gaussian Splatting for grasp planning with anthropomorphic high-Degree of Freedom (DoF) end-effectors [88]. By integrating NVS into grasp prediction during training, this method enhances the accuracy and adaptability of robotic grasping in unstructured environments, particularly for complex objects that are challenging to manipulate. Experimental results show a 36% improvement in grasp success rate for objects with complex geometries and challenging surface properties compared to existing methods. The system demonstrates generalization capabilities to previously unseen objects, making it applicable to a wide range of scenarios.

Together, these contributions form a comprehensive framework that addresses key challenges in autonomous robotics, from object reconstruction to pose estimation and grasp planning. The contributions developed as part of this dissertation have also been published individually and are listed below.

### 1.6.1 List of publications

The following publications, listed in chronological order, form the core contributions of this thesis:

1. **VM-NeRF: Tackling Sparsity in NeRF with View Morphing**  
**Matteo Bortolon**, Alessio Del Bue, and Fabio Poiesi  
International Conference on Image Analysis and Processing (ICIAP), 2023
2. **IFFNeRF: Initialisation Free and Fast 6DoF Pose Estimation from a Single Image and a NeRF Model**  
**Matteo Bortolon**, Theodore Tsesmelis, Stuart James, Fabio Poiesi, and Alessio Del Bue  
IEEE International Conference on Robotics and Automation (ICRA), 2024



3. **6DGS: 6D Pose Estimation from a Single Image and a 3D Gaussian Splatting Model**  
Matteo Bortolon, Theodore Tsesmelis, Stuart James, Fabio Poiesi, and Alessio Del Bue  
European Conference on Computer Vision (ECCV), 2024
4. **GRASPLAT: Enabling Dexterous Grasping Through Novel View Synthesis**  
Matteo Bortolon, Nuno Ferreira Duarte, Plinio Moreno, Fabio Poiesi, José Santos-Victor, Alessio Del Bue  
IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), 2025

For transparency regarding collaborative research efforts, Tab. 1.1 details the specific contributions of each author to the publications included in this thesis.

The following publications, while not forming part of the core thesis contributions, were completed during the doctoral research period:

1. **The MONET Dataset: Multimodal Drone Thermal Dataset Recorded in Rural Scenarios**  
Riz, Luigi, Andrea Caraffa, Matteo Bortolon, Mohamed Lamine Mekhalfi, Davide Boscaïni, André Moura, José Antunes et al.  
IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) Workshop, 2023  
*Contribution:* Data annotation
2. **Free-form Language-based Robotic Reasoning and Grasping**  
Jiao, Runyu, Alice Fasoli, Francesco Giuliani, Matteo Bortolon, Sergio Povoli, Guofeng Mei, Yiming Wang, and Fabio Poiesi  
IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), 2025  
*Contribution:* Robotic setup assistance

## 1.7 Organization of the dissertation

This dissertation is organized into five chapters, including the introduction, each addressing a specific aspect of the research. Chapter 2 reviews related works, focusing on NVS, 6DoF pose estimation using NVS, and robotic grasping prediction. This chapter establishes the foundation for the methods proposed in the subsequent chapters by providing an in-depth analysis of existing techniques and their limitations. Chapter 3 delves into Sparse View NVS, presenting methods and results for view morphing in few-viewpoint Neural Radiance Fields (NeRF). It highlights the challenges associated with generating novel views from a limited number of input images and discusses the proposed approach to address these challenges effectively. Chapter 4 explores 6DoF

### 1.7. Organization of the dissertation

**Table 1.1** Author contributions for the core publications of this thesis

| Publication             | Author Contributions                                                                                                                                                                                                                                                                                                                                                                                                                                                                        |
|-------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| VM-NeRF<br>(ICIAP 2023) | <b>Matteo Bortolon:</b> Implementation, first paper draft, figures<br><b>Alessio Del Bue:</b> Draft review and editing<br><b>Fabio Poiesi:</b> Concept, draft review and editing                                                                                                                                                                                                                                                                                                            |
| IFFNeRF<br>(ICRA 2024)  | <b>Matteo Bortolon:</b> Concept, implementation, first paper draft, figures<br><b>Theodore Tsesmelis:</b> Supervision, draft review and editing, figures<br><b>Stuart James:</b> Supervision, draft review and editing, funding acquisition<br><b>Alessio Del Bue:</b> Research direction, Draft review and editing, funding acquisition<br><b>Fabio Poiesi:</b> Draft review and editing, funding acquisition                                                                              |
| 6DGS<br>(ECCV 2024)     | <b>Matteo Bortolon:</b> Concept, implementation, first paper draft, figures<br><b>Theodore Tsesmelis:</b> Supervision, draft review and editing, figures<br><b>Stuart James:</b> Draft review and editing, funding acquisition<br><b>Fabio Poiesi:</b> Draft review and editing, funding acquisition<br><b>Alessio Del Bue:</b> Research direction, Draft review and editing, funding acquisition                                                                                           |
| GRASPLAT<br>(IROS 2025) | <b>Matteo Bortolon:</b> Concept, implementation, first paper draft, figures<br><b>Nuno Ferreira Duarte:</b> Robotic simulations, draft review and editing<br><b>Plinio Moreno:</b> Draft review and editing, supervision robotic simulations<br><b>Fabio Poiesi:</b> Draft review and editing, funding acquisition<br><b>José Santos-Victor:</b> Draft review and editing, funding acquisition<br><b>Alessio Del Bue:</b> Research direction, draft review and editing, funding acquisition |

pose estimation using NVS. It examines methods and results for both Neural Radiance Fields and 3D Gaussian Splatting, emphasizing their applications in robotic perception and manipulation. The discussion includes a comparative analysis of these techniques and their respective advantages in estimating object poses accurately. Chapter 5 focuses on robotic grasping, specifically how NVS can enhance grasp prediction. It presents methods and results for improving the accuracy and adaptability of robotic grasping in unstructured environments. This chapter also highlights the outcomes of a collaborative project conducted with the Institute for Systems and Robotics (ISR) at Instituto

Superior Técnico (IST) in Lisbon, Portugal, showcasing robotics applications of the proposed methodologies. Finally, chapter 6 concludes the dissertation by summarizing the key findings and contributions of the research. It also outlines potential future directions to further advance the field of autonomous robotics, identifying key challenges and opportunities for continued exploration.



# Chapter 2

## Related works

We start by introducing one of the key components of this thesis: Novel View Synthesis (NVS), which involves generating new perspectives of a scene from a set of input images (see Sec. 2.1). In particular, we will review some of the most important methods that have led to the current development in NVS. Subsequent sections explore related works addressing the core challenges tackled in this thesis. First, we analyze the difficulties associated with sparse view NVS (see Sec. 2.2). We then examine the connection between NVS and 6DoF pose estimation (see Sec. 2.3). Finally, we investigate the relationship between NVS and robotic grasping prediction (see Sec. 2.4).

### 2.1 From Structure-from-Motion to NVS

NVS can be solved either by using traditional 3D reconstruction techniques [36] or by adopting more recent methods based on neural rendering [65]. We first describe the traditional methods and then focus on the neural rendering methods, which are the main focus of this thesis.

As introduced in Sec. 1.2, most of the approaches for NVS and 3D dense reconstruction start from the output of a Structure-from-Motion (SfM) approach. SfM is a photogrammetric technique used to reconstruct sparse 3D structures from a series of 2D images taken from different viewpoints [67]. It works by identifying and matching key points across multiple images, estimating the camera positions and orientations, and then triangulating the 3D coordinates of the matched points. This process involves feature detection, feature matching, triangulation and bundle adjustment to refine the 3D structure and camera parameters. Starting from the 3D structure obtained using SfM, techniques such as Multi-View Stereo (MVS) [68] are used to transform the sparse 3D structure into a dense point cloud or mesh. The reconstructed 3D dense point cloud or mesh can then be used for generating novel views of the scene using rendering algorithms. Importantly, modern NVS algorithms based on neural rendering methods (such as NeRF and 3DGS discussed later) do not require explicit 3D geometric structures beyond the initial SfM point cloud. Instead, these methods rely solely on the estimated camera poses and the images themselves, learning implicit or explicit 3D representations directly from the photometric data. This fundamental distinction

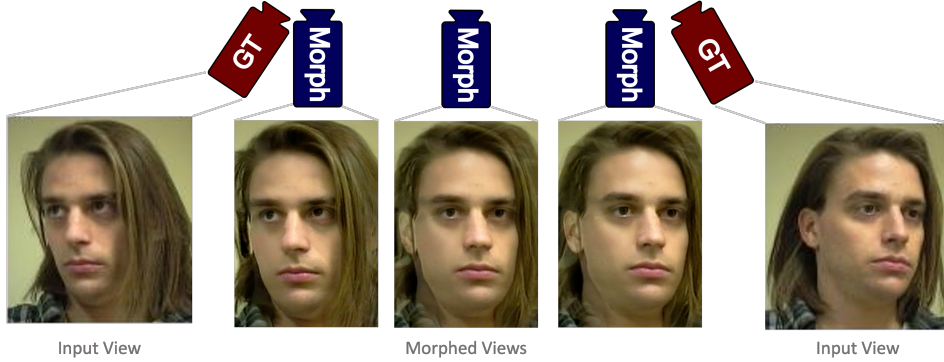


Figure 2.1: Conceptual illustration of View Morphing [6], which enables the synthesis of novel views by interpolating between two reference images. This technique establishes correspondences between images and creates geometrically correct intermediate views.

enables neural rendering approaches to model complex appearance phenomena (such as view-dependent effects and non-Lambertian surfaces) that are difficult to capture with traditional geometric representations.

In the following sections, we first analyze some of the traditional MVS methods that construct explicit geometry, and then the neural rendering methods that learn scene representations from camera poses and images.

### 2.1.1 Multi-View Stereo

Multi-view stereo (MVS) originates from human stereopsis and the stereoscopic matching problem in computer vision. At its core, MVS seeks to establish dense correspondences between multiple images of the same scene, typically starting with the fundamental case of stereo matching along the epipolar line. One of the earlier approaches based on this principle is the View Morphing method [6], which synthesizes a new view of a scene by interpolating between two input images. Due to its relevance to one of our proposed methods, we will analyze View Morphing in detail, then we describe other traditional MVS approaches and finishing with providing an overview of the light fields, that combine 3D geometry densification and image morphing.

#### View morphing

View morphing aims to synthesize natural 2D transitions between an image pair  $\{\mathbf{I}_k, \mathbf{I}_{k'}\}$ , an example of the concept is available in Fig. 2.1. View Morphing can be summarized in three steps: *i)* the two images are *prewarped* through rectification, aligning their image planes without changing their cameras’ optical centers; *ii)* the *morph* is computed between these prewarped images to generate a morphed image whose viewpoint lies on the line connecting the optical centers; *iii)* the image plane of the morphed image is transformed to a desired viewpoint through *postwarping*.

In practice, assuming the two views are prewarped, the morph uses the knowledge of their camera poses  $\mathbf{P}_k, \mathbf{P}_{k'}$ , and the pixel correspondences between the images, rep-

resented as  $q_k : \mathbf{I}_k \Rightarrow \mathbf{I}_{k'}, q_{k'} : \mathbf{I}_{k'} \Rightarrow \mathbf{I}_k$  where  $q_k$  is a function that maps a pixel of  $\mathbf{I}_k$  to the corresponding pixel in  $\mathbf{I}_{k'}$  [6]. Sparse pixel correspondences can be defined by a user or determined by a keypoint detector, then densified via interpolation to create a dense correspondence map. A warp function for each image can be computed from the correspondence map through linear interpolation:

$$\begin{aligned}\hat{\mathbf{I}}_{k,\alpha} &= (1 - \alpha)\hat{\mathbf{I}}_k + \alpha q_k(\hat{\mathbf{I}}_k) \\ \hat{\mathbf{I}}_{k',\alpha} &= (1 - \alpha)q_{k'}(\hat{\mathbf{I}}_{k'}) + \alpha\hat{\mathbf{I}}_{k'} \\ \mathbf{P}_\alpha &= (1 - \alpha)\mathbf{P}_0 + \alpha\mathbf{P}_1,\end{aligned}\tag{2.1}$$

where  $\hat{\mathbf{I}}_k$  are the coordinates of the image from camera  $k$ ,  $\hat{\mathbf{I}}_{k,\alpha}$  are pixel coordinates of the morphed image, and  $\alpha \in [0, 1]$  regulates the position of the morphed view along the line connecting the two views. The morphed image is then computed by averaging the pixel colors of the warped images. For more details on the view morphing process, readers are referred to [6].

### MVS-based methods

Following MVS methods start to determine correspondences using geometric constraints obtained from SfM [89]. These approaches can be categorized into several distinct classes based on their underlying representation and reconstruction strategy. The first class of methods [90, 91] relies on volumetric representations, where voxels are systematically labeled as being inside or outside the surface of the object. This approach provides a straightforward way to represent complex 3D geometries but often suffers from resolution limitations and high memory requirements.

Another approach involves propagating sparse point clouds generated by SfM to form denser reconstructions [34, 92]. These methods typically start with reliable feature correspondences and progressively expand the reconstruction to include more points, balancing efficiency and accuracy. Alternatively, many techniques compute dense point clouds starting from depth maps. These approaches estimate per-view depth maps and fuse them together to recover the complete point cloud [93–95]. Specifically, they reformulate the depth map estimation task as a stereo matching problem between a given view and its neighboring views, leveraging epipolar constraints to reduce the search space for correspondences.

The field of dense point cloud estimation has subsequently experienced significant advancement with the emergence of learning-based MVS methods [96–100]. These methods employ deep neural networks to predict depth maps directly from image inputs, frequently incorporating cost volume representations to capture geometric correspondences across multiple views. Furthermore, they integrate optimization techniques such as regularization and iterative refinement to enhance the accuracy, robustness, and consistency of the estimated geometry.

A recent evolution in MVS techniques is Neural Rendering, which aims to bypass explicit depth estimation by replacing it with differentiable rendering equations and implicit scene representations. This paradigm shift allows for end-to-end optimization of

scene representations, enabling NVS and scene reconstruction without the need for explicit depth computation. These advanced neural rendering techniques will be analyzed in detail in the next section.

### Light fields

Parallel to view morphing and MVS-based methods, other approaches focus on fusing image information from multiple views. These methods, that for brevity we will refer to as light fields [101, 102], directly from a view direction and a specific pixel try to interpolate the color of that pixel [101–103]. This interpolation is made in an intelligent way from multiple images, instead of using only pair of images such as in View Morphing. Furthermore, the approaches try to represent scene appearance as a dense function, avoiding the fragile correspondence matching required by MVS and View Morphing. Levoy and Hanrahan [101] formalized this in the form of a plenoptic function [104]—that allow to encode the flow of light at every position and direction in space. While extremely precise, the plenoptic function is impractical to capture, but Levoy and Hanrahan recognize that radiance remains constant along rays in free space, eliminating a dimension. This, with appropriate limitation regarding the camera position, reconducts the problem to the fusion of two image-like parallel planes.

Concurrently, Gortler et al. [102] proposed the lumigraph with a similar ray parameterization, but demonstrated that incorporating approximate geometric information through depth correction could significantly improve reconstruction quality while requiring fewer input images. This hybrid approach gracefully degrades to standard light field rendering when geometric approximations are coarse, maintaining robustness while leveraging geometry when available. Buehler et al. [103] later generalized this framework to unstructured camera arrangements through Unstructured Lumigraph Rendering (ULR), using camera blending fields to weight contributions from arbitrarily positioned input views. Compared to view morphing’s reliance on establishing accurate correspondences between image pairs—with potential failure in textureless or ambiguous regions—these dense sampling methods trade increased capture requirements for fundamental robustness, representing a paradigm shift from correspondence-based to function-based view synthesis. These techniques properly evolved, for example by increasing the number of planes [105] has shown impressive results in NVS, especially in sparse view settings. In a way, we can say that these methods have inspired the sampling strategy that will later develop in neural rendering methods, such as NeRF [2].

### 2.1.2 Neural Network for NVS

Neural networks have shown great capabilities in solving problems that are difficult to formalize, such as image classification [106]. NVS, with its challenges in formally defining light behavior and the high-dimensionality of the problem, naturally fits the application of neural networks. Flynn et al. [107] introduced DeepStereo, one of the first works to apply deep learning to NVS from real-world imagery. Their approach learns to predict novel views directly from pixels in an end-to-end manner, departing from traditional multi-stage pipelines that require careful tuning of each component. The



network processes pixels from neighboring views and directly produces pixels for unseen viewpoints, demonstrating the potential of learned representations for view synthesis. Concurrently, Zhou et al. [108] proposed View Synthesis by Appearance Flow, which instead of synthesizing pixels from scratch, learns to predict appearance flows—2D coordinate vectors that specify which pixels from input views should be copied to reconstruct target views. This pixel-copying approach achieved higher perceptual quality compared to direct pixel synthesis methods.

Building on these pixel-based methods, Tatarchenko et al. [109] demonstrated that convolutional networks could infer 3D representations of objects from single images, enabling multi-view synthesis by predicting RGB images and depth maps from arbitrary viewpoints. Park et al. [110] extended this by introducing transformation-grounded image generation, which explicitly reasons about geometry visible in both input and novel views, then treats the remaining synthesis problem as image completion. Their method predicts both pixel flow and visibility maps to handle occlusion and disocclusion, combined with adversarial and perceptual losses to reduce common artifacts. The transition toward fully volumetric neural representations began in 2019. Sitzmann et al. [111] proposed DeepVoxels, which encodes view-dependent appearance in a 3D grid of persistent embedded features that learn to exploit underlying scene structure without explicit geometry modeling.

These neural network-based approaches laid the groundwork for the neural rendering revolution, progressively moving from 2D pixel manipulation to continuous 3D representations with differentiable rendering. Their key insight—that scene representations can be learned end-to-end through differentiable rendering rather than constructed through explicit geometric reconstruction—directly influenced the development of NeRF and subsequent methods.

### 2.1.3 Neural Rendering

Neural rendering emerged as a powerful approach to bridge traditional graphics-based rendering and learning-based methods. Early works in this domain focused on representing 3D vision using discrete parameterizations such as voxels or point clouds. A breakthrough came with the idea of encoding variations of physical measurements (such as density) across space as continuous functions [112, 113]. DeepSDF [112] and Occupancy Networks [113] pioneered this approach by parameterizing 3D geometry as continuous fields encoded in neural network weights. Concurrently, Neural Volumes [75] introduced end-to-end rendering equations to synthesize novel views directly from multi-view 2D images. Despite its innovative approach, Neural Volumes relied on discrete voxel grids as its underlying 3D volume representation, making it susceptible to resolution limitations imposed by the Nyquist sampling theorem. Nevertheless, Neural Volumes demonstrated the significant capabilities enabled by end-to-end training of the entire pipeline, bridging the gap between 2D image inputs and 3D geometry reconstruction. Still, Neural Volumes rely on voxel grids as 3D volume representation, making them susceptible to the Nyquist theorem issues [114], but demonstrating the capability derived from end-to-end training. Neural Radiance Fields (NeRF) [2] fused the ideas of

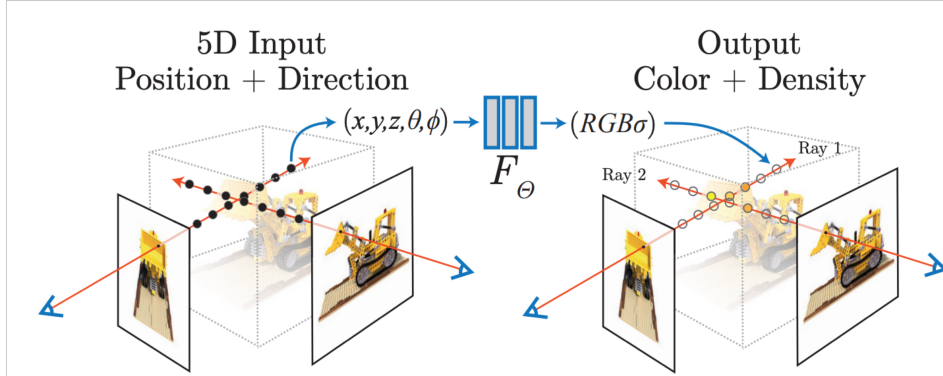


Figure 2.2: Conceptual illustration of ray marching in Neural Radiance Fields (NeRF). Points are sampled along each ray, and the network predicts color and density at each point, which are then composed to render the final pixel color. Image source: [2]

continuous fields and end-to-end training that combine the benefits of both approaches and sparked a paradigm change in graphics and 3D vision. We will analyze the NeRF method in more detail in Sec. 2.1.3. Unfortunately, NeRF presented several drawbacks. In particular, NeRF showed long training time and the inference was not real-time. To address these issues, several methods have been proposed, such as InstantNGP [76] or TensorRF [77] that replace the MLP with grid-like custom structures to improve training and inference performance. While these methods have broken the psychological boundary of 1 frame per second, they are still far from the real-time performance required for many applications, especially on limited hardware such as mobile phones. The ability to perform inference in real-time even on mobile devices came with the introduction of 3D Gaussian Splatting (3DGS) [78]. 3DGS challenges some of the ideas behind NeRF, such as the need to represent the scene as a continuous field, replacing it with a discrete field, and the formulation of the rendering process used by NeRF. Furthermore, the passage to a discrete field allows the use of instruments developed for point clouds even on this new representation, with limited effort to adapt them. Sec. 2.1.3 analyzes the 3DGS method in more detail.

## Neural Radiance Fields

NeRF represents a scene as a continuous function parameterized by an MLP, trained on a set of known views. Mathematically, NeRF models the scene as a function  $f_{\Xi}$  that maps a 5D input, consisting of a 3D spatial location  $(x, y, z)$  and a 2D viewing direction  $(\theta, \phi)$ , to a corresponding color  $\mathbf{c}$  and volume density  $\sigma$ :

$$(\mathbf{c}, \sigma) = f_{\Xi}(x, y, z, \theta, \phi). \quad (2.2)$$

To capture high-frequency details, NeRF employs positional encoding to map the input coordinates  $(x, y, z)$  into a higher-dimensional space. This is achieved by applying a set of sinusoidal functions to each element of the input coordinates separately, defined

as:

$$\gamma(p) = (\sin(2^0\pi p), \cos(2^0\pi p), \dots, \sin(2^{L-1}\pi p), \cos(2^{L-1}\pi p)), \quad (2.3)$$

where  $p$  represents the input coordinates and  $L$  is the number of frequency bands. The encoded coordinates are then concatenated and fed into the MLP, enabling the model to learn fine details and complex variations in the scene. The model is optimized using stochastic gradient descent, minimizing the discrepancy between rendered and input images. Novel views are synthesized by querying the function along camera rays. A visual representation of this process is depicted in Fig. 2.2.

Rendering a novel view consists of estimating the integral of radiance along each camera ray  $\mathbf{r}$  starting from the optical center. Due to the computational challenges of evaluating a continuous function along the ray, a discrete sampling approach is adopted. A set of points denoted as  $\mathbf{r}_i$  are sampled along each ray  $\mathbf{r}$ , and the radiance is approximated as a weighted sum:

$$\hat{\mathbf{c}}(\mathbf{r}) = \sum_{i=1}^{\Gamma} T(i) (1 - e^{-\hat{\sigma}_f(\mathbf{r}_i)\delta_i}) \hat{\mathbf{c}}_f(\mathbf{r}_i), \quad (2.4)$$

where  $\hat{\sigma}_f(\mathbf{r})_i$  is the predicted density of the  $i$ -th point along the ray  $\mathbf{r}$ ,  $\delta_i = \|\mathbf{r}_i - \mathbf{r}_{i-1}\|_2$  represents the Euclidean distance between the current point and the previous one,  $T(i)$  denotes the accumulated transmittance along the ray up to point  $i$ , which controls the contribution of each sampled point to the final rendering and can be computed recursively as:

$$T(i) = e^{-\sum_{j=1}^{i-1} \hat{\sigma}_f(\mathbf{r}_j)\delta_j}, \quad (2.5)$$

Similarly, we can also compute the depth of a pixel as:

$$\hat{\mathbf{d}}(\mathbf{r}) = \sum_{i=1}^{\Gamma} T(i) (1 - e^{-\hat{\sigma}_f(\mathbf{r}_i)\delta_i}) \mathbf{z}_i. \quad (2.6)$$

The NeRF optimization process requires a dataset  $\mathcal{I} = \{\mathbf{I}_k\}_{k=1}^N$  of images with corresponding camera poses  $\mathcal{P} = \{\mathbf{P}_k\}_{k=1}^N$  and intrinsic parameters  $\mathcal{K} = \{\mathbf{K}_k\}_{k=1}^N$ . The camera poses  $\mathcal{P}$  and the intrinsics  $\mathcal{K}$  are used to create a set of rays  $\mathcal{R} = \{\mathbf{r}\}^{N \times H \times W}$ . The training procedure aims to minimize the L2 loss between predicted and input pixel colors:

$$\mathcal{L} = \frac{1}{NHW} \sum_{\mathbf{r} \in \mathcal{R}} (\|\mathbf{c}(\mathbf{r}) - \hat{\mathbf{c}}_c(\mathbf{r})\|_2^2 + \|\mathbf{c}(\mathbf{r}) - \hat{\mathbf{c}}_f(\mathbf{r})\|_2^2). \quad (2.7)$$

Despite its impressive results, NeRF suffers from several limitations. NeRF requires extensive training time and demands a large number of input views for high-quality synthesis. Additionally, its implicit representation poses challenges for integration with traditional 3D graphics pipelines, which predominantly rely on explicit geometric representations such as meshes or point clouds.

To mitigate these limitations, numerous improvements have been proposed. Mip-NeRF [4] introduces Integrated Positional Encoding enabling multi-scale rendering,

effectively addressing aliasing artifacts and improving rendering quality at different levels of detail. Similarly, Ref-NeRF [115] enhances the handling of specular reflections by introducing a modified density MLP that outputs additional surface properties such as roughness and normal vectors.

While methods such as Mip-NeRF and Ref-NeRF have significantly improved rendering quality, they have not addressed NeRF’s long training and inference times. To tackle this challenge, one category of approaches incorporates additional supervision signals, such as depth information or SfM point clouds, which are frequently used to estimate camera poses for NVS. For instance, DS-NeRF [116] employs depth supervision from sparse point clouds extracted via COLMAP and regularizes the predicted depth by constraining it within a normal distribution. NerfingMVS [117] further improves depth reconstruction by leveraging monocular depth priors obtained from a pretrained depth estimation network, which helps guiding volume sampling and improving geometric consistency. PointNeRF [118] introduces an intermediate feature point cloud representation, significantly accelerating rendering by focusing computations only on occupied regions, thus avoiding unnecessary sampling in empty space. This concept was later revisited and expanded upon in the development of 3DGS.

While these methods have reduced training time, inference speed remains a bottleneck, particularly for real-time applications, such as rendering on mobile devices or robotic systems. The most effective solution to this issue has been the development of baking techniques, which distill the information from a trained NeRF into an optimized representation that enables faster querying and rendering. For example, PlenOctree [119] introduces an octree-based structure where each node stores spherical harmonic coefficients, allowing for efficient scene representation and rapid inference. On the other hand, MobileNeRF [120] converts NeRF representations into polygonal meshes, leveraging small MLPs for shading, thereby enabling real-time rendering on mobile devices without significant loss of visual fidelity.

In this section, we have reviewed the advancements in NeRF and their applications in NVS. While the improvements discussed have partially addressed many of NeRF’s original limitations, some challenges remain unresolved, particularly concerning real-time performance, scalability, and seamless integration into industrial workflows. Consequently, in the following section, we introduce 3DGS, a novel approach that draws inspiration from prior research aimed at overcoming NeRF’s constraints. In particular, 3DGS incorporates ideas from methods such as PointNeRF [118] while significantly improving efficiency and rendering speed.

### 3D Gaussian Splatting

3DGS [78] introduces 3D Gaussians as an efficient scene representation, typically initialized from a sparse point cloud obtained via Structure-from-Motion (SfM). This fundamental idea is similar to DS-NeRF [116], which leverages the SfM point cloud to improve convergence, although 3DGS demonstrates the ability to start from a random point distribution and still converge.

Each 3D Gaussian has a set of properties, including its 3D center, opacity, anisotropic

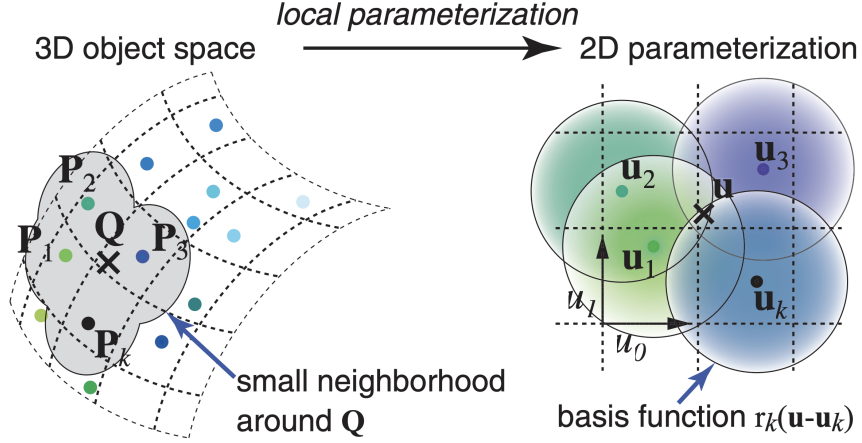


Figure 2.3: Visual representation of the Gaussian splatting process, showing how 3D Gaussians are projected onto the image plane. This approach, initially proposed in Elliptical Weighted Average (EWA) splatting and later adopted by 3D Gaussian Splatting, efficiently renders complex scenes by projecting 3D Gaussian primitives to 2D. Image source: [7]

covariance, and spherical harmonic coefficients to encode view-dependent effects. 3D Gaussians represent a good compromise: they retain the differentiable volumetric representation of NeRF, while using a rasterization similar to point-based methods for speed. 3D Gaussians rasterization idea is shown in Fig. 2.3. Furthermore, 3DGS implements the entire splatting and rendering pipeline in CUDA, fully leveraging GPU parallelism and optimizing computations by constraining operations to specific GPU architectures.

We now delve into the mathematical principles underlying 3DGS. We consider the 3D Gaussians as ellipsoids, denoted as  $\mathcal{Q} = \{\mathbf{Q}\}_{i=1}^K$ . Ellipsoids parameters are optimized from a set of input images  $\mathcal{I} = \{\mathbf{I}\}_{i=1}^J$  and their corresponding camera projection matrices  $\mathcal{P} = \{\mathbf{P}\}_{i=1}^J \in \mathbb{R}^{3 \times 4}$ . A point  $\mathbf{d}$  lies on the surface of an ellipsoid if it satisfies the equation:

$$(\mathbf{d} - \mathbf{x})^T \Sigma (\mathbf{d} - \mathbf{x}) = 1, \quad (2.8)$$

where  $\mathbf{x} \in \mathbb{R}^3$  is the ellipsoid center, and  $\Sigma \in \mathbb{R}^{3 \times 3}$  is its covariance matrix. The covariance matrix  $\Sigma$  can be decomposed as:

$$\Sigma = \mathbf{R} \mathbf{U} \mathbf{U}^T \mathbf{R}^T, \quad (2.9)$$

where  $\mathbf{R} \in \mathbb{R}^{3 \times 3}$  is the ellipsoid rotation matrix and  $\mathbf{U} \in \mathbb{R}^{3 \times 3}$  is the scaling matrix. The projection matrix  $\mathbf{P} \in \mathbb{R}^{3 \times 4}$  maps the ellipsoid  $\mathbf{Q}$  onto the image plane, generating the corresponding ellipse representation:

$$\check{\mathbf{y}} = \mathbf{P} \check{\mathbf{x}}, \quad \check{\mathbf{E}} = \mathbf{P} \Sigma \mathbf{P}^T, \quad (2.10)$$

where  $\mathbf{y} \in \mathbb{R}^2$  is the ellipse center in Euclidean coordinates, and  $\check{\mathbf{y}} \in \mathbb{R}^3$  is its homogeneous counterpart. The matrix  $\check{\mathbf{E}} \in \mathbb{R}^{3 \times 3}$  represents the ellipse covariance in

homogeneous space. The 2D covariance matrix  $\mathbf{E} \in \mathbb{R}^{2 \times 2}$  is derived by selecting the first two rows and columns of  $\tilde{\mathbf{E}}$  and normalizing by the last diagonal element of  $\tilde{\mathbf{E}}$ .

The projected ellipses, denoted as  $\mathcal{B} = \{\langle \mathbf{y}, \mathbf{E} \rangle\}_{i=1}^K$ , contribute to pixel color computation using the rendering function  $\phi$  via rasterization techniques [78, 121]. The function  $\phi$  operates independently on each pixel  $\mathbf{p}$  and considers neighboring projected ellipses, their center distances, orientations, and scales. Assuming depth ordering of ellipses,  $\phi$  is defined as:

$$\phi(\mathcal{B}, \mathbf{p}) = \sum_{i=1}^K \rho_i \alpha_i e^{-\tau(\mathcal{B}_i, \mathbf{p})} \gamma(i, \mathbf{p}), \quad (2.11)$$

where  $\rho$  and  $\alpha$  represent the color and opacity attributes of each ellipse. The function  $\gamma$  accounts for accumulated volume density up to the  $i^{th}$  ellipse and is defined as:

$$\gamma(i, \mathbf{p}) = \prod_{j=1}^{i-1} (1 - \alpha_j e^{-\tau(\mathcal{B}_j, \mathbf{p})}). \quad (2.12)$$

The function  $\tau$  models light absorption by a 2D Gaussian ellipse and is given by:

$$\tau(\mathcal{B}, \mathbf{p}) = \frac{1}{2} (\mathbf{1}_2 \mathbf{d}^T \mathbf{E} \mathbf{d} \mathbf{1}_2^T), \quad (2.13)$$

where  $\mathbf{1}_2 \in \mathbb{R}^2$  is a vector of ones. A visual representation of the splatting process is depicted in Fig. 2.3.

Although 3DGS demonstrates superior computational efficiency during inference, it nevertheless exhibits certain inherent constraints. High-quality rendering requires a large number of Gaussians, leading to increased memory usage and computational demands. Unlike NeRF, whose MLP weights are around  $\approx 5\text{MB}$ , a comparable 3DGS model can exceed  $\approx 600\text{MB}$ . To mitigate this, pruning and compression techniques have been explored [122–125]. Additionally, researchers have investigated improved Gaussian duplication and pruning strategies to enhance training efficiency [126, 127]. Moreover, the original 3DGS formulation is less flexible than NeRF regarding supported camera models, prompting efforts to merge NeRF’s ray tracing advantages with 3DGS’s rasterization speed [128–131].

Having introduced the two primary NVS methods, their limitations, and potential solutions, the subsequent sections will analyze key techniques addressing the challenges that this thesis aims to resolve.

## 2.2 Sparse View Novel View Synthesis

One of the main drawbacks of NeRF is its reliance on a large number of input views to generate high-quality novel views. This limitation poses significant challenges in robotics applications, where acquiring numerous views is often impractical. Consequently, several methods have been developed to address the challenge of sparse view NVS. This problem is relevant not only for NeRF but also for 3DGS. Some approaches

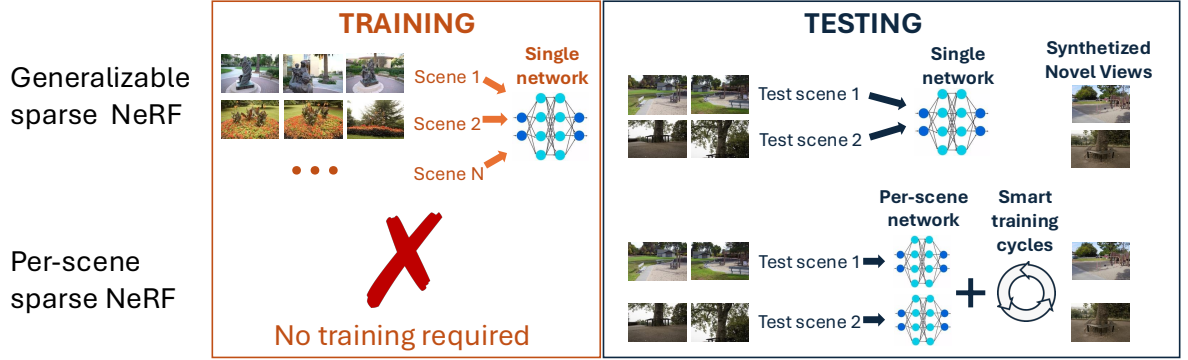


Figure 2.4: Comparison between generalizable sparse NeRF and per-scene sparse NeRF pipelines. A single generalizable network enables novel view synthesis across multiple test scenes without requiring per-scene fine-tuning, whereas per-scene models demand separate training for each individual scene. While per-scene models may lack generalization capabilities, their performance limitations are compensated by sophisticated training procedures that effectively constrain the optimization process.

are generic and can work independently of the underlying representation, while others are specifically designed for either NeRF or 3DGS. Notably, most methods have been evaluated primarily on one of these representations. Therefore, we first describe methods developed with NeRF in mind, followed by those designed for 3DGS.

### 2.2.1 NeRF-based Sparse View Novel View Synthesis

NeRF-based approaches for sparse-view NVS can be broadly categorized into two groups. The first group aggregates generalizable methods that train a single network to generate novel views for multiple scenes. The second group optimizes per-scene following the original NeRF paradigm, training a separate network for each scene. A visualization of this concept is presented in Fig. 2.4.

#### Generalizable NeRF

The first category of methods trains on datasets characterized by similar scenes, such as DTU [132]. While multi-scene training allows for generalization, it can introduce dataset biases, leading to poor performance on scenes outside the training domain [133, 134]. Typically, these methods leverage large pretrained feature extractors, such as CNNs, to condition the NeRF model. The key idea is that the NeRF MLP does not encode a single continuous field but rather multiple fields, where scene-specific information can be activated through conditioning features. Furthermore, interpolating between these features enables the synthesis of novel views of unseen scenes.

PixelNeRF [135] was among the first methods to adopt this approach. It utilizes pretrained CNN layers combined with bilinear interpolation to extract image features. Each point sampled along camera rays is projected onto known images, and the corresponding feature map is used for interpolation. After aggregating features from multiple

views, the final representation is fed into the NeRF MLP, which predicts density and color. General Radiance Fields [136] follow a similar approach but operate in a canonical space rather than the view-space representation used by PixelNeRF.

MVSNeRF [137] takes a different approach by constructing a voxelized cost volume using plane sweeping stereo, where CNN features are mapped to depth-dependent voxel layers. A 3D CNN further processes this cost volume to generate per-point latent codes through trilinear interpolation. Similar to PixelNeRF, the NeRF MLP then uses these latent features, along with point coordinates and viewing direction, to predict density and color values. The training procedure involves joint optimization of both the 3D feature volume and the NeRF MLP parameters.

The Neural Rays approach [138] also incorporates a cost volume, leveraging multi-view stereo depth estimation [98, 139]. A CNN converts depth maps into feature maps, which serve as visibility indicators. Instead of averaging features across all views, NeuRay uses these visibility-aware weights to selectively aggregate features before passing them to the NeRF MLP for further processing. This approach helps to avoid rendering artifacts in empty regions of the scene.

Although NeuRay successfully addresses visibility challenges, it potentially sacrifices critical view-dependent chromatic information in the process. GeoNeRF [140] addresses this by replacing simple averaging with a learned transformer [57] that intelligently fuses features. Specifically, view-independent and view-dependent feature tokens are extracted and refined via transformer layers. The view-independent tokens are then processed using an autoencoder to predict densities, while view-dependent tokens are used for color prediction. NeRFusion [141] follows a similar idea but employs a Gated Recurrent Unit instead of a transformer for feature fusion.

Research in NVS demonstrates the intrinsic interconnection between geometric reconstruction and view synthesis capabilities—enhancements to either component yield corresponding improvements in the complementary function. Consequently, methods such as NeuS [142] and Neuralangelo [38] encode surfaces as signed distance functions (SDFs) rather than densities, which are difficult for MLPs to represent. SparseNeuS [143] extends this concept to sparse NVS by using an SDF-based NeRF MLP, reducing ambiguity in surface reconstruction.

While generalizable NeRF methods offer fast inference on new scenes, they often exhibit a bias toward the training dataset, leading to suboptimal results on unseen scenes with novel geometry or appearance. One way to mitigate this limitation is fine-tuning the NeRF MLP per scene, as demonstrated in ShaRF [144].

### **Per-scene sparse NeRF**

This category of methods adheres to the original per-scene optimization paradigm of NeRF, where a single network is trained and tested on only one specific scene or environment. This eliminates dataset bias but often leads to overfitting on the given views. To mitigate overfitting, various approaches introduce additional semantic or geometric constraints during the optimization training process.

One of the pioneering works in this category is DietNeRF [11]. The core idea behind



DietNeRF is that an object remains semantically identical regardless of the viewpoint from which it is observed. To incorporate this intuition, DietNeRF utilizes features extracted from a pretrained and frozen neural network, such as CLIP [145]. During training, in addition to using ground-truth camera poses, DietNeRF randomly samples novel camera viewpoints and renders corresponding images. These renderings are processed by the CLIP encoder, and their feature embeddings are compared against those from known images. The optimization employs a cosine similarity loss between the corresponding feature embeddings extracted from rendered views. However, NeRF’s high computational cost makes rendering high-resolution images impractical for training. To address this limitation, DietNeRF computes the loss using low-resolution images, which, while computationally efficient, results in a significant loss of fine geometric details.

RegNeRF [12] extends this approach by shifting the focus from whole-image constraints to patch-based regularization. Instead of rendering entire images, RegNeRF samples small patches from randomly generated camera poses and compares them to corresponding patches from the nearest known views. This comparison is performed based on both depth and color information.

To further address the challenges of sparse input views, RegNeRF introduces a patch-based regularization technique that enforces smooth geometry and realistic color consistency. The depth smoothness regularization is defined as:

$$\mathcal{L}_{DS}(\mathcal{S}) = \sum_{i,j=\mathbf{p}}^S \left( \hat{\mathbf{d}}(r_{\mathbf{p}}) - \hat{\mathbf{d}}(\mathbf{r}_{i+1,j}) \right)^2 + \left( \hat{\mathbf{d}}(\mathbf{r}_{ij}) - \hat{\mathbf{d}}(\mathbf{r}_{ij+1}) \right)^2, \quad (2.14)$$

where  $\mathcal{S} = \{\mathbf{p}\}$  is the set of pixels inside the patch, and  $\mathbf{r}_{\mathbf{p}}$  denotes the ray corresponding to pixel  $\mathbf{p}$ . To prevent degenerate color predictions and ensure stable optimization, RegNeRF also incorporates a color regularization term. The key idea is that the rendered patches should exhibit natural appearance characteristics. To achieve this, a RealNVP normalizing flow model [146] is trained on image patches from the JFT-300M dataset [147], learning to map patches to a standard normal distribution. The color regularization loss is then formulated as:

$$\mathcal{L}_{NLL}(\mathcal{S}) = -\log p_Z(N(\hat{\mathbf{c}}(\mathcal{S}))), \quad (2.15)$$

where  $N$  is the learned bijective mapping from the RGB patch to a latent space following a normal distribution,  $\hat{\mathbf{c}}(\mathbf{p})$  represents the predicted RGB color patch, and  $-\log p_Z$  is the negative log-likelihood under a Gaussian prior  $p_Z$ .

FreeNeRF [148] proposes a training methodology that mostly consist on the progressive activation of the frequency bands in the positional encoding. This allows to train the network on low frequencies first, and then progressively increase the frequency bands, allowing the network to learn the high frequency details. More recent methods, such as ZeroRF [149], begun incorporating generative models like Stable Diffusion [150] to synthesize new views. ZeroRF, in particular, integrates a tailored Deep Image Prior, based on Stable Diffusion, into the TensorRF [77] representation.

Single-scene NeRF-based methods trained on sparse views are prone to overfitting,

leading to artifacts when rendering novel viewpoints. The general approach to mitigating overfitting is data augmentation [151]. This approach is followed by AugNeRF [10] and GeoAug [152]. AugNeRF improves NeRF generalization by leveraging adversarial data augmentation, ensuring that a given ray and its augmented counterpart yield consistent predictions. GeoAug, on the other hand, perturbs the translation and rotation of known viewpoints during training. Specifically, GeoAug generates additional training images by rendering new views with noisy camera poses via warping.

Our proposed approach, VM-NeRF [85], developed as part of this thesis, falls within this category. Unlike GeoAug, VM-NeRF does not perturb existing input views and rays. Instead, it generates novel viewpoints using 3D projective transformations derived from pairs of known views. This enables the enforcement of coherence between novel viewpoints and distant known-view pairs, improving generalization.

Nevertheless, the introduction of 3DGS, which shall be examined in the subsequent section addressing sparse NVS, has necessitated a fundamental reconsideration of most methodologies predicated on NeRF.

### 2.2.2 3DGS-based Sparse View Novel View Synthesis

Among 3DGS-based methods for sparse view NVS, some directly extend their NeRF counterparts. Notable examples include PixelSplat [153], derived from PixelNeRF [135], and MVSplat [154], which builds upon MVSNeRF [137]. Alternatively, other methods integrate concepts from traditional sparse-view NVS techniques. One such approach is SparseGS [155], which combines the diffusion-based supervision of ZeroRF [149] with depth supervision techniques inspired by DS-NeRF [116] and Reg-NeRF [12]. GaussianObject [156] leverages the capabilities of ControlNet [157] to guide the image generation process of diffusion models [150] using input images. Similar to prior works, GaussianObject randomly samples new camera poses during training, renders them using 3DGS, and employs ControlNet to refine the generated images, which are subsequently used to update the 3DGS model.

Depth supervision also plays a crucial role in CoherentGS [158]. This method introduces a two-stage depth refinement process: first, a monocular depth estimation network [159] initializes the 3DGS model, and then a residual depth estimation network refines the optimization process, enforcing smoothness constraints. Similarly, DNGaussian [160] proposes a depth normalization approach that compares rendered depth with estimates from a monocular depth network. DNGaussian introduces a multi-scale depth consistency loss, comparing estimated and rendered depth both locally (at the image patch level) and globally (across the entire image).

Unlike these methods, FSGS [161] is the first approach specifically designed for 3DGS rather than an adaptation of existing NeRF-based techniques. It initializes the 3DGS model with a sparse point cloud obtained from a SfM method and introduces a novel pooling mechanism to generate new Gaussians in previously empty regions. The core idea is akin to vertex subdivision in mesh processing, where new vertices are inserted at the midpoints of triangle edges, with the caveat of adding Gaussians instead of individual vertices. A notable limitation of existing methods is their reliance

on known camera poses. This dependency is not problematic in robotic applications, where camera poses can be readily obtained using joint encoders in an eye-in-hand configuration or through calibration matrices in an eye-to-hand setup. The reliance on known camera poses presents challenges in fields such as 3D modeling. In many cases, the bottleneck lies in the failure of the SfM methods used to estimate camera poses. To address this issue, recent works like InstantSplat [162] attempt to tackle the entire pipeline, from images without pose information to NVS. In particular, InstantSplat employs a pretrained model to estimate both camera poses and initialization points before leveraging this information to optimize the 3DGS model. Once an object model is reconstructed, the next step involves its relocalization within the scene, that will be analyzed in the next section.

## 2.3 6DoF Pose Estimation using NVS

Estimating the 6 Degrees of Freedom (6DoF) pose of an object is a fundamental task for robotics, as it usually the starting point to decide how grasp the object. Similar to previous sections, we will first discuss methods developed for NeRF, followed by those designed for 3DGS.

### 2.3.1 Early works on NeRF and Pose Refinement

In the context of NVS, the first techniques that appear that try to merge the estimation or refinement of a pose emerged from the necessity to generate novel views without precise camera pose information. While not specifically tackling the 6DoF pose estimation issue as objective, these methods laid the groundwork for subsequent research in this area.

NeRF-based methods for 6DoF pose estimation have evolved significantly, with early approaches focusing on jointly optimizing camera poses and NeRF parameters simultaneously. The primary objective was to enable the generation of novel views without requiring explicit camera pose information, similar to what InstantSplat [162] would later achieve in the context of 3DGS. Unfortunately, most of these methods are constrained by the fundamental limitation of needing an initial pose estimate for convergence. While the end goal of these approaches is not primarily pose estimation, the underlying principle is that at testing time, a test image can be added to the training, the optimization restarted, and the new pose is jointly optimized alongside the scene representation—thereby estimating the camera’s position and orientation in the scene.

One of the pioneering works in this field is NeRF-- [163], which introduces the concept of making camera pose parameters optimizable alongside the NeRF model parameters. This innovation enables NVS directly from images with potentially noisy or inaccurate pose information. A significant limitation of NeRF--, however, is its effectiveness only on front-facing scenes, as it struggles with rotational motion and object tracking in more complex scenarios. To address these limitations, Bundle-Adjusted NeRF [164] proposes a coarse-to-fine registration strategy that adaptively masks the NeRF positional encoding—a technique similar to the approach later used in FreeNeRF [148].

This methodological refinement significantly improves pose estimation robustness during training, particularly in challenging scenarios with limited visual features, complex geometries, or difficult lighting conditions.

GARF [165] explores the use of Gaussian activations as an alternative to traditional positional encoding in NeRF architectures. Specifically, these Gaussian activations can be mathematically defined as:

$$e^{-\frac{\mathbf{l}^2}{2\sigma^2}}, \quad (2.16)$$

where  $\mathbf{l}$  represents the input feature and  $\sigma$  is a hyperparameter that controls the bandwidth of the network: a larger  $\sigma$  corresponds to a lower bandwidth, and vice versa. The enhanced network differentiability provided by GARF, when combined with BARF’s coarse-to-fine approach, allows the system to function effectively even in challenging scenarios such as scenes dominated by low-textured regions.

The methods proposed thus far, however, do not optimize camera intrinsic parameters, which can be particularly difficult to estimate accurately, especially for complex optical systems like fisheye lenses. SCNeRF [166] addresses this limitation by introducing a self-calibrating joint optimization model capable of refining both camera poses and intrinsic parameters, including those for non-linear camera models. SCNeRF’s approach involves gradually incorporating non-linear camera and noise parameters into the optimization process, making the framework modular and compatible with various NeRF variants. Unfortunately, the necessity of an initialization pose and the requirement to restart training with all images, including those with the inaccurate pose, makes this categories of approaches unfeasible for the 6DoF pose estimation task.

#### 2.3.2 NeRF-based 6DoF Pose Estimation

To address these operational limitations, researchers have developed methods specifically designed for pose estimation using NVS models. These methods can be broadly categorized into two complementary paradigms: camera pose refinement methods, which optimize camera parameters with respect to a pre-trained neural scene representation, and object pose estimation methods, which leverage neural representations as implicit 3D models for correspondence-based pose prediction. Understanding this distinction is crucial, as both approaches exploit the camera-object duality introduced in Sec. 1.2.2 of Chapter 1, where the relationship  $\mathbf{P}_{\text{obj}}^{\text{cam}} = (\mathbf{P}_{\text{cam}}^{\text{obj}})^{-1}$  establishes that estimating object pose relative to the camera is mathematically equivalent to determining camera pose relative to the object’s coordinate frame.

Among the methods that leverage neural implicit representations for object pose estimation, NeRF-Pose [167] represents one of the first approaches to use the neural representation not for direct pose optimization, but rather as a source of supervision for learning-based correspondence prediction. This paradigm enables more efficient inference by decoupling the reconstruction and pose estimation stages. NeRF-Pose follows a first-reconstruct-then-regress strategy, operating in two distinct stages. First, an object-centric NeRF model (termed OBJ-NeRF) is trained from multiple views using only weak supervision in the form of 2D segmentation masks and relative camera poses—notably, neither CAD models nor accurate 6D pose annotations are required. During this recon-

struction phase, OBJ-NeRF simultaneously optimizes the neural scene representation  $f_{\Xi}$  and a canonical object pose  $\mathbf{P}_0$  by transforming sampled 3D points from camera coordinates to object-centric coordinates before querying the NeRF network. The key innovation is that OBJ-NeRF can then render ground-truth Normalized Object Coordinate Space (NOCS) maps [168], which encode dense 2D-3D correspondences between image pixels and the object surface in canonical space. In the second stage, a CNN-based regression network is trained to predict these dense correspondences directly from RGB images, supervised by the NOCS maps rendered from OBJ-NeRF. At inference time, given a query image, the regression network predicts pixel-wise 2D-3D correspondences, which are then used in a novel NeRF-enabled PnP+RANSAC algorithm to estimate the object pose. This algorithm enhances traditional PnP+RANSAC by incorporating additional scoring criteria based on mask overlap between predicted segmentation and NeRF-rendered masks, leading to more robust pose hypotheses selection. By using the NeRF model to generate supervision signals for training a feed-forward pose regression network, NeRF-Pose achieves efficient inference (approximately 0.25 seconds per image) while leveraging the rich geometric information encoded in neural implicit representations without requiring explicit CAD models.

In contrast, iNeRF inverts the process: it treats the camera pose as an unknown variable and directly optimizes it at test time by minimizing photometric error between the query image and NeRF-rendered views. This inversion exploits the mathematical equivalence between estimating object pose in the camera frame and camera pose in the object frame, allowing pose estimation without explicit correspondences or offline training. The camera-object duality thus underpins both approaches, but iNeRF demonstrates that neural rendering models can be used not only for generating supervision but also for direct, test-time pose recovery through differentiable rendering. Given a pre-trained NeRF model and a query image, iNeRF iteratively refines the camera pose by minimizing the photometric discrepancy between the query image and images rendered from the NeRF at candidate poses. This optimization process is defined as:

$$\hat{\mathbf{P}}_{\text{cam}} = \underset{\mathbf{P}}{\operatorname{argmin}} \|\mathbf{I}_{\text{query}} - \operatorname{Render}(f_{\Xi}, \mathbf{P}, \mathbf{K})\|^2, \quad (2.17)$$

where  $f_{\Xi}$  represents the pre-trained NeRF model,  $\operatorname{Render}(\cdot)$  denotes the differentiable volume rendering function, and the optimization is performed via gradient descent. Through the camera-object duality, once the camera pose  $\hat{\mathbf{P}}_{\text{cam}}^{\text{obj}}$  is estimated with respect to the object-centric NeRF model, the object pose in the camera frame is simply obtained as  $\hat{\mathbf{P}}_{\text{obj}}^{\text{cam}} = (\hat{\mathbf{P}}_{\text{cam}}^{\text{obj}})^{-1}$ . While demonstrating impressive accuracy in controlled laboratory environments, this direct optimization approach suffers from several critical shortcomings, including high inference time (approximately 30 seconds per image compared to NeRF-Pose’s 0.25 seconds), inefficient pose parameter updates, and strong dependence on accurate initial pose estimates for successful convergence. These fundamental limitations significantly hamper iNeRF’s effectiveness in real-world robotic scenarios where conditions are less controlled and computational resources constrained.

Recent advancements attempt to address some of these issues. Parallel iNeRF [169] introduces parallel optimization using Monte-Carlo sampling techniques, generating

and simultaneously optimizing multiple candidate poses that are pruned and regenerated based on optimization trends. While this strategy improves convergence rates and robustness, it does not eliminate the fundamental need for an initial camera pose estimate and still exhibits high computational demands during inference. Similarly, Loc-NeRF [170] employs Monte-Carlo and particle filtering techniques to localize a robot within an environment using a 3D NeRF map as a reference. Although this approach does not require prior pose information in the traditional sense, it relies on multiple sequential images and prior knowledge of their inter-image movement patterns rather than operating on a single image, which limits its applicability in certain scenarios.

Alternative methodologies, such as CROSSFIRE [171], incorporate learned descriptors directly into the NeRF model. CROSSFIRE employs a joint training process of a visual backbone and the NeRF model to establish alignments between 3D scene representations and 2D image features. During testing, CROSSFIRE adopts an analysis-by-synthesis approach, minimizing feature discrepancies rather than direct color differences. However, both CROSSFIRE and Parallel iNeRF require reasonably accurate initial camera positions to effectively optimize for the actual camera pose, limiting their practical utility in scenarios where such initial estimates are unavailable.

The approach presented in this dissertation, IFFNeRF [86], proposes a method that effectively inverts the NeRF model to re-render an image that matches a target input. Our inversion technique does not require an initial camera pose estimate, making it substantially more suitable for real world applications where such information are usually unavailable. The methods developed specifically for 3DGS-based pose estimation will be discussed in the following section.

#### 2.3.3 3DGS-based 6DoF Pose Estimation

While NeRF-based methods have been extensively explored, 3DGS-based approaches for 6DoF pose estimation are still emerging. As established in the previous section, the camera-object duality  $\mathbf{P}_{\text{obj}}^{\text{cam}} = (\mathbf{P}_{\text{cam}}^{\text{obj}})^{-1}$  applies equally to 3DGS-based methods, where estimating an object’s pose relative to the camera is mathematically equivalent to determining the camera’s pose relative to the object’s coordinate frame. Consequently, 3DGS-based approaches can be similarly categorized into camera pose refinement methods (often termed relocalization in the 3DGS literature) and object pose estimation methods that leverage 3DGS as an implicit 3D model. We review both categories below, noting that the literature has predominantly focused on the relocalization problem.

Among the existing methods, NeMo+VoGe [172, 173] introduced object models based on volumetric Gaussian reconstruction kernels as geometric primitives. Their rendering strategy, VoGE, differs from 3DGS as it relies on ray marching rather than point-based splatting. NeMo+VoGe iteratively aligns learned features from target and rendered images to estimate the pose. However, its training process necessitates multiple objects, contrasting with methods that leverage a single-object 3DGS model.

Pose estimation from ellipsoidal representations has been explored in SfM [174–179] and SLAM [180–185] scenarios. These approaches typically recover object ellipsoid

representations alongside camera 6DoF poses by solving linear systems that minimize projection errors. However, such methods suit handling a few large ellipsoids modeling individual objects rather than the numerous smaller ellipsoids used in 3DGS.

In traditional 6DoF image matching, feature-based approaches such as SIFT [186], SuperGlue [187], and TransforMatcher [188] are commonly used for establishing correspondences between images. SuperGlue employs a Graph Neural Network for feature attention and utilizes the Sinkhorn algorithm [189] for optimal matching, while TransforMatcher performs global match-to-match attention to enhance localization accuracy. However, these methods assume homogeneous feature spaces and consistent geometric representations, which is not the case for 3DGS, where explicit pixel-to-ray correspondences must be established.

Hybrid approaches such as OnePose++ [190] attempt to bridge this gap by matching point clouds with images, while CamNet [191] directly regresses pose parameters from images. However, these methods require extensive datasets and are not optimized for the sparse data typically encountered in 3DGS.

A related method, 3DGS-ReLoc adopts a staged pipeline similar to traditional 6DoF methods based on image matching but leverages 3DGS as its underlying representation. The relocalization map is stored in a 2D voxel grid with a KD-tree for efficient spatial queries. During relocalization, a submap near the initial pose estimate is selected and used to render images and depth maps. The query image is then compared against the rendered sequence using a brute-force search to identify the closest match. Feature-based matching and the Perspective-n-Point method are subsequently applied to iteratively refine the query image’s pose, ensuring accurate localization within the global map. A similar approach is followed in GS-Pose [192], which first constructs a reference database to aid in pose estimation. Specifically, it generates a set of rotation-aware embedding vectors from reference images, which are used to estimate an initial pose. This estimate is then refined using a pose refinement module based on 3DGS, enhancing the final localization accuracy.

In contrast, our method, 6DGS [87], directly leverages the 3DGS representation to estimate the 6DoF pose without requiring any reference images. Furthermore, it is designed to handle the large number of Gaussians typically found in 3DGS models, making it more scalable and efficient in complex scenes.

In a robotic pipeline the natural next step after estimating the 6DoF pose of an object is to predict the best grasp pose to manipulate it. The following section will discuss the state-of-the-art methods for robotic grasping prediction based on NVS and some other related methods.

## 2.4 Robotic Grasping Prediction

Robotic grasping prediction is a fundamental task in robotics, enabling robots to interact with objects in their environment effectively and safely. The ability to predict appropriate grasps is crucial for successful manipulation tasks in both industrial and domestic settings. A first classification of robotic grasping prediction techniques can be based on the type of end-effector considered. Most techniques are designed with

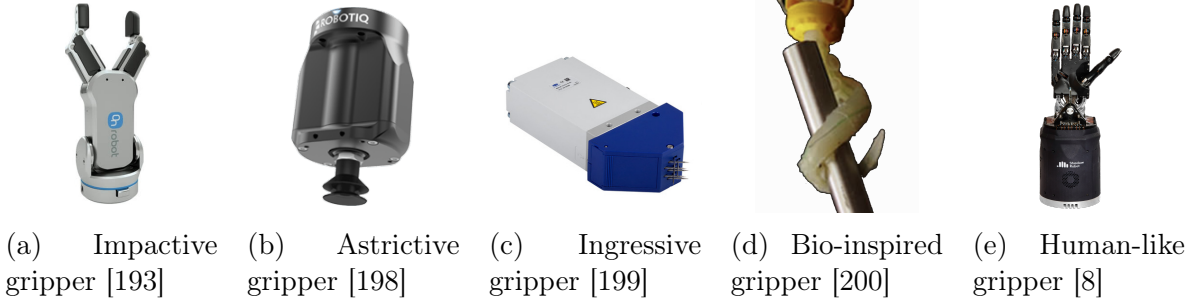


Figure 2.5: Examples of different robotic end-effectors: (a) Robotiq 2F-85 gripper, (b) Robotiq EPick vacuum gripper, (c) Schmalz needle gripper SNGi-AE, (d) Octopus-inspired soft gripper, and (e) Shadow Dexterous Hand.

a specific end-effector in mind, as the mechanical constraints and capabilities significantly influence the grasping strategy. Consequently, Sec. 2.4.1 will present the most common robotic end-effectors and their characteristics. Then, we will discuss existing approaches for robotic grasping prediction based on NVS in Sec. 2.4.2. Considering that the approach proposed as part of this dissertation involves the application of NVS to human-like end effectors, which to the best of our knowledge has not been explored by NVS methods, we will also review existing methods not based on NVS (Sec. 2.4).

### 2.4.1 Robotic end-effectors

Robotic end-effectors, or grippers, are essential components in robotics, designed to interact with and manipulate objects in various environments. They come in diverse types, each tailored to specific applications and challenges, with their design significantly influencing the approach to grasp prediction. The most common type of gripper is the impactive gripper [18]. Impactive grippers rely on physical force applied from opposite sides to maintain the object in position, usually in a claw-like shape. They are widely used for tasks requiring strong and precise grips, such as industrial assembly lines. However, they may struggle with delicate or irregularly shaped objects. Commercial impactive grippers include the Robotiq 2F-85 [193] (Fig. 2.5a), the Schunk grippers series [194], and the Franka Hand [195]. Another common type of gripper is the astrictive gripper. These grippers employ suction or magnetic forces and are ideal for handling fragile items or soft materials but can face limitations with non-flat surfaces. Examples of astrictive grippers include the OnRobot Gecko [196] and MG10 [197], and the Robotiq EPick [198] (Fig. 2.5b). Ingressive grippers, which penetrate or pierce through materials, are particularly effective in handling fabrics, thin sheets, or flexible materials but can potentially damage the objects they manipulate (Fig. 2.5c). They are often used in applications such as textile handling and industrial fabric processing.

Finally, one of the most recent developments in the area of end effectors is the contiguous gripper. This type of gripper is designed to conform to the object’s shape, offering adaptability and flexibility through continuous surface contact. Contigutive grippers can be further distinguished into two major categories: bio-inspired mecha-



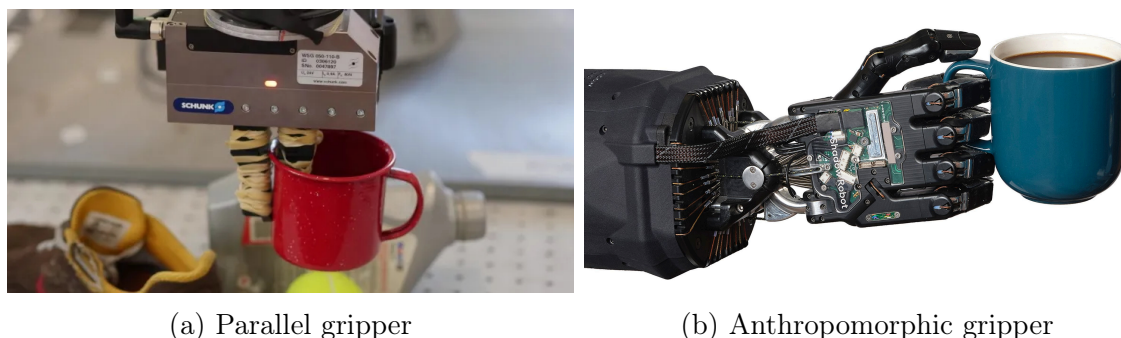


Figure 2.6: Examples of the same object (a mug) grasped with two different gripper types: (a) parallel and (b) anthropomorphic (or dexterous). While both successfully grasp the mug, the parallel gripper’s approach appears unnatural and limits functionality—common actions like pouring could potentially damage its circuits. In contrast, the anthropomorphic gripper’s grasp enables nearly all typical mug interactions, including pouring. Additionally, attempting to grasp the handle with a parallel gripper would likely result in an unstable grip, as it cannot apply sufficient force to prevent the mug from slipping. Image sources: Tom Buehler/MIT CSAIL (parallel gripper) and [8] (anthropomorphic gripper).

nisms [201] and human hand-like end-effectors [27].

Bio-inspired end-effectors draw inspiration from natural mechanisms found in the animal and plant kingdoms [202]. These designs seek to replicate the remarkable adaptability and efficiency observed in biological systems. For example, Mazzolai et al. [200] propose a gripper inspired by octopus tentacles that can conform to objects of various shapes and sizes without requiring complex control systems [203] (Fig. 2.5d). The inherent compliance of these bio-inspired grippers enables excellent performance in delicate object handling and adaptation to irregular shapes, primarily due to the implementation of smart materials with variable stiffness properties [202, 203]. These characteristics allow such grippers to excel in environments where traditional rigid grippers would struggle, such as agricultural settings with delicate produce or biomedical applications requiring gentle tissue manipulation [201, 202].

In contrast, human-like end-effectors (also known as anthropomorphic hands) mimic the structure and functionality of the human hand. Their primary advantage lies in their ability to act upon and use tools originally designed for human manipulation, making them ideal for deployment in robotic platforms that must interact with human environments such as homes, offices, or healthcare facilities. Figure 2.6 illustrates this distinction, demonstrating how the same object can be manipulated by both a parallel gripper and an anthropomorphic end-effector. However, the parallel gripper usage often results in unnatural grasping pose, whereas the anthropomorphic design enables more intuitive object manipulation. These human hand-like manipulators, commonly referred to as dexterous grippers, are available in various commercial models that offer different degrees of anthropomorphism and mechanical complexity. The Shadow Dexterous Hand Series [8] (Fig. 2.5e), with 24 Degrees of Freedom, closely mimics hu-

man hand kinematics and actuation patterns, providing exceptional dexterity and fine manipulation capabilities. Each finger joint can be independently controlled, allowing for complex in-hand manipulation tasks that would be impossible with simpler gripper designs. The Allegro Hand Series [204], a more cost-effective alternative with 16 DoF, features three fingers and a thumb. While less directly comparable to a human hand in structure, it remains capable of executing a wide range of grasping configurations suitable for most manipulation tasks. Another commercially available solution is the QB SoftHand [205], which represents an innovative approach to dexterous manipulation through an underactuated design. In this system, a single motor drives multiple joints through a synergistic control scheme inspired by human hand neuroscience, enabling adaptive and compliant grasping while significantly reducing control complexity and implementation costs. The development of these advanced dexterous grippers has made significant contributions to robotic manipulation capabilities, though challenges remain in control strategies, sensory integration, and grasp planning algorithms that can fully exploit their mechanical potential.

### 2.4.2 Novel View Synthesis for Robotic Grasping Prediction

NVS has recently emerged as a powerful approach for robotic grasping prediction, offering several advantages over traditional methods. By providing rich 3D structural information and the ability to synthesize views from arbitrary perspectives, NVS enables more robust grasp planning, particularly for objects with complex geometry or challenging surface properties. NVS-based grasping methods can be categorized into NeRF and 3DGS based approaches, each with distinct characteristics and capabilities.

#### NeRF-based Grasping Methods

NeRF-based methods leverage the rich 3D structural information encoded in NeRF to plan and execute grasping operations. These approaches typically follow two main paradigms: using NeRF as a scene prior generator or directly incorporating NeRF into the grasping pipeline for enhanced performance and robustness. It is important to note that most of these algorithms work exclusively with parallel grippers, primarily due to the simplicity of the grasping model and the reduced dimensionality of the search space compared to dexterous hands.

One group of approaches involves using NeRF to provide strong 3D scene priors for robust grasp planning. Hu et al. [206] train a NeRF model of target objects to generate template images, which are then used as input for a neural network. Given a human demonstration of a grasp with a different object, this network translates the demonstration to the template object. However, it is worth noting that the output of this methodology is not the complete grasp pose, but rather the object’s bounding box, requiring additional processing to determine the final grasp configuration.

Chen et al. [207] propose a comprehensive two-stage method where the first stage predicts the object pose using a neural network trained on a synthetic dataset created from a NeRF model. The second stage employs a robust poking strategy that iteratively

tries alternative approaches until the object is successfully grasped, demonstrating the system’s adaptability to different scenarios.

NeRF demonstrates exceptional performance with transparent or reflective objects, which are traditionally challenging for vision-based grasping systems. Dex-NeRF [40] exploits NeRF’s volume density field to capture globally consistent scene geometry, facilitating grasp planning for transparent objects that are difficult to perceive with conventional depth sensors. After scene reconstruction, Dex-NeRF employs a traditional grasp planning method [208] to predict optimal grasp poses.

Similarly, Tang et al. [209] utilize a NeRF-like model to reconstruct the scene while simultaneously outputting an Euclidean Signed Distance Function (ESDF). This ESDF representation is then used to predict grasp poses using a mesh-based method [210], enabling effective grasp planning even in complex environments.

While these approaches show promising results, they often require significant computational resources. To address this limitation, Kerr et al. introduced Evo-NeRF [35], a method capable of obtaining results comparable to Dex-NeRF but with a significantly lower computational cost. This efficiency is achieved through an evolutionary algorithm that optimizes the number and positioning of input views before executing the grasp, making it more suitable for real-time applications.

GraspNeRF [211] represents one of the first approaches to directly modify the NeRF architecture for grasp prediction. It implements a generalizable NeRF architecture that predicts Truncated Signed Distance Function (TSDF) values, which are then processed by a dedicated grasp detection module to determine optimal grasping poses. This end-to-end approach effectively eliminates the need for separate scene reconstruction and grasp planning steps.

Blukis et al. [212] propose a modified generic NeRF architecture that directly predicts grasp pose effectiveness alongside color and density. This approach effectively encodes the grasping pose as a continuous field, an idea shared with the concurrent work by Weng et al. [213]. Additionally, inspired by iNeRF [214], the authors propose using backpropagation to create motion paths for various joints leading to the grasp, enabling smooth and natural grasping trajectories.

NeRF-Dy [215] demonstrates that integrating a NeRF model into a low-level visual motion control pipeline improves performance and robustness. By providing accurate scene geometry and appearance information, NeRF-Dy enables more precise visual servoing and trajectory planning. Mira [216] showcases how NeRF’s capability to convert from perspective to orthographic projection can lead to improved policy prediction. This transformation simplifies the representation of spatial relationships between objects, making it easier for the grasping policy to understand the scene geometry.

Rather than training end-to-end like NeRF-Dy or Mira, Neural descriptor fields [217, 218] demonstrate that the process can be effectively split into two distinct stages: first training the NeRF as a 3D representation, and then using this representation to train the grasping policy. This decoupled approach leads to more stable training dynamics and often better performance.

Additionally, NeRF-Supervision [219] has shown that NeRF can be extremely useful for learning robust visual encoders for various robotic tasks. By providing accurate 3D

supervision signals, NeRF helps train encoders that are more resistant to viewpoint changes and lighting variations.

SPARTN [220] proposes using NeRF models as a data augmentation method to train policies that are more robust to noise. This is achieved by providing slightly perturbed viewpoints from those used in the original demonstrations, effectively expanding the training dataset with realistic variations.

Some approaches have extended beyond visual perception to incorporate tactile information. For example, Zhong et al. [221] train a GAN to generate tactile images representing contacts based on NeRF-rendered images. The resulting dataset can then be used to detect grasping success using tactile sensors attached to the gripper tips, providing a multi-modal approach to grasp evaluation.

Furthermore, NeRF has been effectively applied to enhance reinforcement learning (RL) based grasping approaches. NeRF-RL [222] introduces an innovative framework where novel view rendering serves as a proxy task, leveraging the latent space generated by a generalizable NeRF as a rich state representation for action learning. This approach follows a two-stage methodology: first, a generalizable NeRF model is trained on an offline dataset to capture scene geometry and appearance, then this model is frozen and utilized as a feature extractor during the subsequent RL optimization phase. By incorporating NeRF’s comprehensive scene understanding capabilities into the reinforcement learning pipeline, NeRF-RL enables more sample-efficient policy learning and improved generalization to novel objects and environments. SNeRL [223] enhances this approach by supervising the encoder with both RGB information and semantic labels, jointly trained with a self-supervised teacher network. This semantic understanding allows the system to better generalize to novel objects and scenes.

While NeRF-based methods have shown impressive results in robotic grasping, they often suffer from slow inference times and significant computational overhead, limiting their applicability in real-time robotic systems and practical deployment scenarios. This substantial limitation has motivated the exploration of alternative NVS approaches, particularly 3DGS, which offers more efficient rendering capabilities.

### 3DGS-based Grasping Methods

The emergence of 3DGS has led to new approaches for robotic grasping that leverage its computational efficiency and representational capabilities. Among these methods, GaussianGrasper [224] reconstructs a 3D Gaussian semantic feature field by incorporating an embedding with each Gaussian that maps it to a language space. This enables language-guided manipulation tasks, where natural language commands can specify which objects to grasp and how to manipulate them. After constructing this semantic field, a traditional grasp planning method based on point clouds [225] is used to predict optimal grasp poses.

Similarly, GraspSplats [226] also encodes language features into the 3DGS model for robotic manipulation. However, it adopts a more advanced training methodology designed to guarantee sharper object boundaries, which is crucial for accurate grasp planning in cluttered environments.

The advent of 3DGS has also enabled methods that tackle multi-stage manipulation tasks, where object movement results in dynamic environmental changes. ManiGaussian [227] introduces a dynamic 3DGS framework to model scene-level spatiotemporal dynamics. This approach represents the environment as a combination of a static scene world model and a dynamic 3DGS model representing the robot, which is updated at each step based on the action to be executed. During training, the goal is to fit the model to observations in the training environment, enabling accurate prediction of how actions will affect the scene.

Object-Aware GS [228] takes a different approach by first segmenting the scene into distinct objects. The photometric loss is then used to understand object displacement, after which the 3DGS model is updated by applying the same transformation to all Gaussians belonging to the same object. For non-rigid transformations, such as a gripper closing around an object, small per-Gaussian adjustments are applied, allowing the model to accurately represent deformations. Splat-MOVER [229] proposes a multi-stage pipeline for connecting language, vision, and grasping. The first stage involves distilling language and vision features into the 3DGS model, creating a rich representation that understands both visual and semantic properties of the scene. A scene-editing module then visualizes potential object motions resulting from the robot’s interactions with the environment. Finally, a fine-tuned traditional grasp prediction module [208] determines optimal grasp poses based on this enhanced representation.

SparseGrasp [230] introduces an open-vocabulary robotic grasping system based on 3DGS, designed specifically for sparse-view scenarios where only limited observations of the scene are available. The approach follows the InstantSplat [162] procedure to generate the initial 3DGS model, after which vision foundation models are used to incorporate language features. A render-and-compare procedure allows for quick and effective Gaussian updates in multi-stage manipulation scenarios, enabling the system to adapt to changing environments. As with several other methods, the grasping prediction is performed using a traditional point cloud-based method [225], fine-tuned specifically on 3DGS data to account for the unique characteristics of Gaussian representations.

A limitation of most approaches discussed so far is that they do not consider the physics of 3D Gaussians. To address this, Physically-Embodied GS [231] combines particle physics with 3DGS to create a representation that can both visualize the scene and simulate physics. This integration simplifies the creation of simulation environments and enables policy training for complex tasks such as pushing, rope manipulation, and object falling, where accurate physical modeling is crucial for success.

It is worth noting that none of the NVS-based approaches discussed thus far consider human-like end effectors, focusing instead on simpler parallel grippers. Given the potential advantages of dexterous manipulation with anthropomorphic hands, this represents a significant gap in the current research landscape. In the following sections, we will explore geometric and RGB-based methods specifically designed for dexterous grasping, providing context for our contribution in this area.

### 2.4.3 Dexterous Grasping

The complexity of human-like end effectors presents unique challenges for grasp prediction, leading to the development of specialized geometric approaches. These methods typically leverage 3D representations such as point clouds or meshes to predict appropriate hand configurations. UniDexGrasp [232] implements a two-stage pipeline for dexterous grasping. In the first stage, the system samples potential grasp rotations, after which a normalizing flow neural network predicts the corresponding translation and joint angles. These initial predictions are subsequently refined using an ideal contact map, which encodes optimal contact points between the hand and object surface. In the second stage, a reinforcement learning-based grasping policy further refines the grasp, adjusting finger configurations to maximize stability and task performance. This hybrid approach combining analytical and learning-based components demonstrates robust performance across diverse object geometries.

Building upon transformer architectures, the Dexterous Grasp Transformer [233] offers a different approach to the same problem. This method begins by encoding the object’s point cloud using a PointNet++ network, creating a rich feature representation that captures local and global geometric properties. A transformer architecture then processes these features to generate a set of candidate grasp poses, leveraging self-attention mechanisms to understand relationships between different parts of the object. The learning process follows a coarse-to-fine strategy: the model initially learns rough grasp poses to understand general hand-object relationships, and subsequently refines these poses to ensure precision. At test time, an additional optimization-based refinement step further improves the results by fine-tuning hand configurations based on collision detection and contact quality metrics.

While these geometric methods demonstrate impressive capabilities, they typically require complete 3D object models or dense point clouds as input. In contrast, our proposed approach, GRASPLAT, part of this dissertation, leverages transformers similarly to the Dexterous Grasp Transformer but differs fundamentally in its input requirements. Instead of relying on comprehensive 3D data, GRASPLAT operates on a single RGB image of the object, making it more practical for real-world robotic applications where complete geometric information may not be readily available.

Methods that predict dexterous grasps directly from RGB images are comparatively rare but offer significant practical advantages. Two notable approaches in this domain are GanHand [49] and Affordance Diffusion [234].

GanHand [49] employs a three-stage pipeline to address the problem of predicting anthropomorphic hand grasps from RGB images. The process begins with an object shape or pose estimation step, where a neural network infers the 3D geometry of the target object from the input image. Using this detailed geometric information, a CNN then predicts an initial grasp configuration, encoding hand position, orientation, and joint angles. Finally, GanHand carefully refines these hand parameters through an extensive optimization process that considers the reconstructed object geometry in detail, ensuring stable contact points and minimizing penetration between the hand and object surface. While our GRASPLAT method shares GanHand’s approach of directly predicting hand parameters, it completely eliminates the intermediate object reconstruction

step, resulting in a more streamlined and computationally efficient pipeline.

Affordance Diffusion [234] takes a generative approach to the problem, extending diffusion models to human grasp prediction. This method first predicts a hand-object interaction layout, encoding spatial relationships and contact points. It then synthesizes realistic hand grasps through a controlled diffusion process, generating diverse yet plausible hand configurations. To convert these synthesized images into actual hand parameters, Affordance Diffusion employs a pre-trained hand pose estimation network. This approach excels in producing natural-looking grasps but may struggle with precise control due to the probabilistic nature of hand pose estimation network. In contrast to both these methods, our approach, GRASPLAT offers several distinct advantages. Unlike GanHand, it eliminates the need for explicit object reconstruction, reducing computational overhead and improving efficiency. And unlike Affordance Diffusion, GRASPLAT directly visualizes the predicted pose during training by leveraging the deterministic properties of 3DGS. This allows for immediate visual feedback on grasp quality and enables more efficient optimization. Additionally, GRASPLAT’s integration with 3DGS facilitates NVS of the predicted grasp, providing multi-view consistency that is crucial for robust robotic manipulation in real-world scenarios.

#### 2.4.4 Chapter conclusion

This chapter has presented a comprehensive review of NVS approaches and their applications to robotic manipulation, tracing the evolution from traditional structure-from-motion and multi-view stereo techniques to advanced neural rendering methods. Our analysis reveals that while NVS technologies show tremendous promise for addressing the limitations of traditional perception systems—particularly for optically challenging objects—several critical obstacles remain before these techniques can be effectively integrated into practical robotic systems.

As highlighted in the abstract, the perception bottleneck in robotic systems involves three interdependent subtasks: geometry reconstruction, pose estimation, and grasp prediction. Our literature review confirms that current NVS approaches face significant challenges in each of these areas when applied to robotic manipulation:

First, geometry reconstruction using neural representations requires an impractically dense set of input views for high-quality results. Despite advances in sparse-view NVS from methods like DietNeRF [11], RegNeRF [12], and FSGS [161], the quality degradation with highly limited viewpoints remains a significant barrier to practical deployment in robotic systems across various environments. This limitation directly motivates our work on VM-NeRF in Chapter 3, which combines classical geometric constraints with neural rendering to achieve high-quality NVS from as few as four input images—a crucial advancement for real-world robotic applications where collecting numerous views is often impractical or impossible.

Second, pose estimation methods based on neural representations still struggle with robustness and computational efficiency in complex scenes. Current approaches like iNeRF [214] require accurate initialization and involve time-intensive optimization, making them unsuitable for real-time robotic applications in dynamic environments. Chap-

ter 4 addresses these limitations through two complementary approaches: IFFNeRF, which eliminates the need for initialization while enabling real-time performance, and 6DGS, which reformulates the problem within the 3DGS framework to achieve both high accuracy and computational efficiency. These methods enable direct and reliable pose validation by comparing query images with synthesized viewpoints—a capability highlighted in the abstract as essential for robust robotic manipulation.

Third, existing NVS-based grasping approaches have predominantly focused on simple parallel-jaw grippers, neglecting the potential of dexterous multi-fingered hands that could significantly expand the range and versatility of manipulable objects. While methods like GanHand [49] and Affordance Diffusion [234] have made progress in RGB-based dexterous grasping, they either require explicit 3D reconstruction or rely on probabilistic image generation without immediate visual feedback. Chapter 5 introduces GRASPLAT, which enhances grasping precision for anthropomorphic robotic hands by incorporating NVS during grasp planning—enabling the system to visually assess potential grasps before physical execution.

The comprehensive framework developed in this dissertation systematically addresses these challenges by integrating geometric principles with neural rendering techniques across the entire perception-to-manipulation pipeline. By reducing the number of required views for neural object representation, enabling initialization-free pose estimation, and incorporating visual feedback for dexterous grasp prediction, our work advances the practical application of NVS in robotic manipulation. The following chapters detail our methodology and experimental validation for each contribution, demonstrating how these advances bring us closer to robust, adaptive robotic systems capable of manipulating optically challenging objects in complex real-world environments.



# Chapter 3

## Sparse View for Novel View Synthesis

This chapter addresses one of the fundamental challenges in enabling Novel View Synthesis (NVS) for robotic applications: using fewer images to reconstruct a scene [64,235]. Here, we introduce a novel approach that combines the geometric principles of View Morphing with the representational power of Neural Radiance Fields (NeRF) to achieve high-quality novel view synthesis from a sparse set of input images. Our proposed method significantly reduces the number of required input views for high-quality scene reconstruction, from the original 100 down to as few as 4 images, while maintaining the reconstruction quality. This significantly simplifies the integration of NVS techniques in robotic perception pipelines. This represents a critical step toward making neural rendering techniques viable for robotic systems operating in dynamic environments.

### 3.1 View Morphing for Few-Viewpoint Neural Radiance Fields

NVS has been largely driven by NeRF [2, 4], which provide an implicit neural representation of 3D scenes capable of rendering photorealistic novel views. Despite their impressive rendering quality, as mentioned in Sec. 1.4, NeRF models typically require a dense collection of training images—often hundreds of viewpoints uniformly distributed around the subject of interest [2]. When trained with sparse or non-uniformly distributed viewpoints, NeRF models frequently suffer from overfitting to the available views, resulting in poor generalization to novel viewpoints that are distant from the training data [11, 105]. This limitation severely restricts the applicability of NeRF in robotic scenarios where collecting numerous views is impractical, such as in time-sensitive operations or resource-constrained environments.

To address this challenge, we propose VM-NeRF, a novel approach to the few-shot view synthesis problem that leverages View Morphing [6] within the NeRF framework (Fig. 3.1). View Morphing is a purely geometric technique that can synthesize intermediate views between two reference viewpoints while preserving realistic perspective transitions. The original View Morphing, described in Sec. 2.1.1, is not viable as is in a learning-based pipeline, hence our approach define a novel view morphing strategy

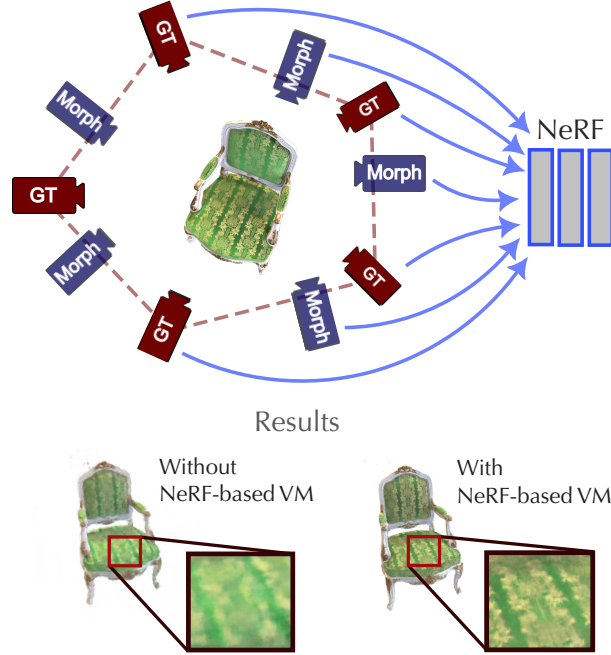


Figure 3.1: Given a set of known views (ground truth), View Morphing-NeRF (VM-NeRF) generates image transitions between views (morph) that can be effectively used to train a NeRF model in the case of few-shot view synthesis. Results are of a higher quality when VM-NeRF is used.

for a NeRF-based network architecture. Our key insight is to utilize the implicit depth information estimated by NeRF to establish dense correspondences between views after an image rectification process (Fig. 3.2), eliminating the need for manual feature matching that traditional View Morphing requires. This crucial step allows us to seamlessly integrate geometric constraints into the neural rendering pipeline.

VM-NeRF does not require any prior knowledge about the scene geometry and can effectively synthesize 3D projective transformations (including rotations, translations, and shears) by operating entirely on the input images. The approach tightly integrates the geometric view generation process during the standard NeRF training procedure, leveraging the fundamental principles of projective geometry to create geometrically consistent image transitions between viewpoints. This effectively addresses the overfitting problem that typically occurs when NeRF models are trained with few viewpoints, significantly improving novel view synthesis quality in sparse-view scenarios. Geometrically accurate intermediate views generation expands the training dataset with physically plausible synthetic data, making it more suitable for robotic applications.

Evaluations on the benchmark dataset from the original NeRF paper [2] demonstrate that our method improves PSNR by up to 1.8dB and 1.0dB when trained with eight and four views, respectively. Comparative analysis with state-of-the-art few-shot methods including DietNeRF [11], AugNeRF [10], and RegNeRF [12] shows that VM-NeRF produces higher-quality renderings across multiple test scenarios. These results validate the effectiveness of integrating classical geometric constraints with modern neural

rendering approaches, bringing NVS closer to robotic applications.

The primary contributions of this chapter are:

- A novel methodology that effectively addresses the few-shot view synthesis problem in NeRF by incorporating geometric constraints;
- A depth-based View Morphing technique that seamlessly integrates with neural rendering pipelines without requiring manual feature matching;
- Empirical evidence demonstrating that VM-NeRF achieves superior rendering quality compared to existing approaches in the literature across various scenes and view configurations, facilitating the integration of NVS in robotic systems.

### 3.1.1 Approach

#### NeRF-based View Morphing

The goal of NeRF-based View Morphing (VM-NeRF) is to use the geometrical constraints of the morphing technique to synthesize a set of additional training input views  $\mathcal{M} = \{\mathbf{M}_{(k,k'),\alpha}\}$ , where  $\mathbf{M}_{(k,k'),\alpha}$  is a morphed view generated from the view pair  $k$  and  $k'$  with a given value of  $\alpha$ . Adapting view morphing in a learning-based pipeline is challenging as we need reliable pixel correspondences ( $q_k$  and  $q_{k'}$ ) to synthesize morphed views. Our intuition is that it is possible to compute one-to-one correspondences from the disparity information, a function of the depth as in Eq. 2.6, which we can render with the very same NeRF model. We can then linearly interpolate the photometric content of the view pair to produce the morphed view.

Based on the description in Sec. 2.1.1, we integrate in NeRF only the steps of prewarping and morphing. We experimentally found that postwarping does not lead to better results. We perform the initial rectification of the two cameras (Sec. 3.1.1), compute the image morphing (Sec. 3.1.1), and provide detailed information on our practical approach to training NeRF with View Morphing (Sec. 3.1.1). Fig. 3.2 shows the block diagram of our approach.

#### Rectification

Our first step is rectification, which leads to rotating the known camera poses  $\mathbf{P}_k$  and  $\mathbf{P}_{k'}$  around their optic centers until their image planes become coplanar. We can then compute the common image plane by using a selection of algorithms such as [9, 236]. We represent this plane as the rotation matrix

$$\tilde{\mathbf{R}} = [\mathbf{a}_x, \mathbf{a}_y, \mathbf{a}_z], \quad (3.1)$$

where  $\mathbf{a}_x, \mathbf{a}_y, \mathbf{a}_z$  are the axis components of the coplanar plane resulting from the rectification. Stereo rectification is applied to the original images  $\{\mathbf{I}_k, \mathbf{I}_{k'}\}$  and depth maps  $\{\mathbf{D}_k, \mathbf{D}_{k'}\}$  predicted in Eq. 2.6. The new camera pose of view  $k$  is equal to  $\tilde{\mathbf{P}}_k = [\tilde{\mathbf{R}}, \mathbf{t}_k]$ , where  $\mathbf{t}_k$  is the translation of the original camera pose  $\mathbf{P}_k$  (same applies to view  $k'$ ).

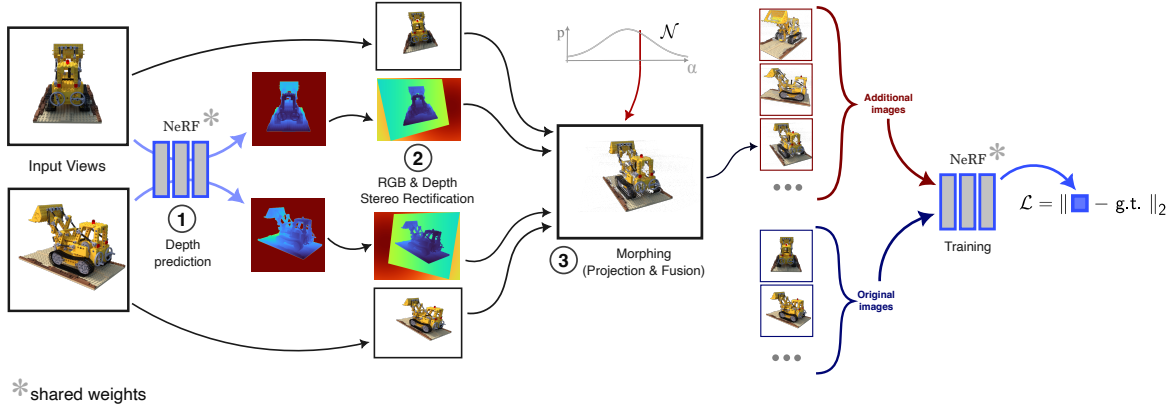


Figure 3.2: Block diagram of NeRF-based View Morphing (VM-NeRF). From the left, we (1) predict the depth with NeRF, (2) rectify the input images and predicted depths, and (3) compute the image morphing of a view randomly positioned between the view pair.  $\alpha$  determines the new view position and it is sampled from a Gaussian distribution.

Rectification algorithms are typically based on the assumptions that viewpoints are aligned horizontally and that the reference viewpoint is the left-hand side camera (from an observer positioned behind the cameras) [9, 236]. However, this assumption is problematic in NeRF scenarios, where viewpoints may have arbitrary camera configurations. To address this challenge, we build upon the Fusiello et al. [9] algorithm, which aims to balance distortion between cameras during rectification. However, the original Fusiello algorithm fails when camera poses are vertically aligned or inverted relative to the world coordinate system. We extend this algorithm to handle general stereo configurations by analyzing the orientation of the rectified coordinate system. Specifically, we compare  $\mathbf{a}_z$  with the  $z$  component of the original view poses. When this angle exceeds  $45^\circ$  with respect to both  $\mathbf{P}_k$  and  $\mathbf{P}_{k'}$ , we apply corrective rotations of  $90^\circ$  or  $180^\circ$  to the warping matrices and poses. This modification ensures the correct generation of rectified images  $\{\tilde{\mathbf{I}}_k, \tilde{\mathbf{I}}_{k'}\}$  and rectified depth maps  $\{\tilde{\mathbf{D}}_k, \tilde{\mathbf{D}}_{k'}\}$ .

Fig. 3.3 shows the application of [9] to two stereo configurations where the reference viewpoint is (i) the right-hand side camera (horizontal configuration) and (ii) the top camera (vertical configuration). Here, we can observe that when the algorithm in [9] is applied as is the 2D transitions are incorrectly flipped. Instead, if we use our algorithm, the 2D transitions are consistent with starting camera poses.

### Image morphing

The second step is image morphing, i.e. fusing the rectified images to obtain the new morphed image. This procedure is divided in three steps: *i*) finding the pixel correspondences; *ii*) computing the position of each pixel on the morphed camera; *iii*) fusing pixels that fall in the same position. To determine the image correspondences,

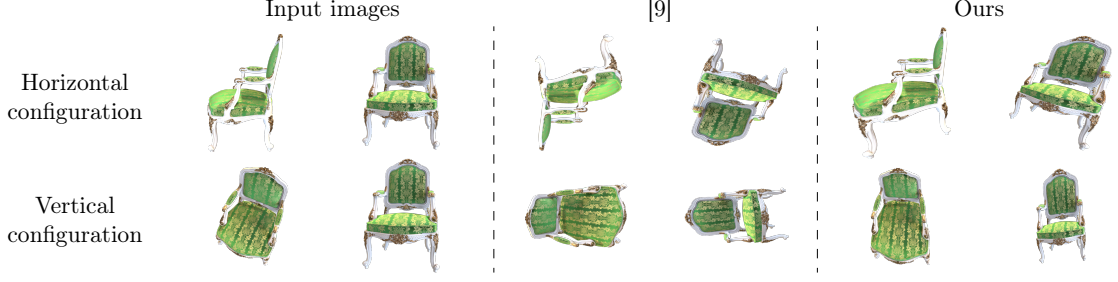


Figure 3.3: Cases where the algorithm in [9] fails to produce rectification results that are consistent with the original camera configuration.

we initially compute the disparity maps as functions of the rectified estimated depths

$$\mathbf{E}_k = \frac{f_k}{\tilde{D}_k} \|\mathbf{o}_k - \mathbf{o}_{k'}\|_2, \quad \mathbf{E}_{k'} = \frac{f_{k'}}{\tilde{D}_{k'}} \|\mathbf{o}_k - \mathbf{o}_{k'}\|_2, \quad (3.2)$$

where  $\{\mathbf{o}_k, \mathbf{o}_{k'}\}$  are the principal points and  $\{f_k, f_{k'}\}$  are the focal lengths of cameras  $k$  and  $k'$ . Then, we determine the correspondences of the pixel positions between images defined in Eq. 2.1 as

$$q_k(\hat{\mathbf{I}}_k) = \hat{\mathbf{I}}_k + \frac{\tilde{\mathbf{b}}_k}{\|\tilde{\mathbf{b}}_k\|_2} \mathbf{1}^\top \odot \mathbf{E}_k, \quad (3.3)$$

where  $\mathbf{1}$  is a vector of ones,  $\odot$  indicates the Hadamard product and  $\hat{\mathbf{I}}_k$  is the baseline direction with respect to the common plane defined in Eq. 3.1 that is computed as

$$\tilde{\mathbf{b}}_k = \mathbf{a}_z \times ((\mathbf{o}_k - \mathbf{o}_{k'}) \times \mathbf{a}_z). \quad (3.4)$$

The same operation is computed for the corresponding view  $k'$ . Then, we apply the warp functions of Eq. 2.1 to compute the position of each pixel on the morphed view, thus obtaining  $\hat{\mathbf{I}}_{k,\alpha}$  and  $\hat{\mathbf{I}}_{k',\alpha}$ .

Lastly, a coalescence operation [237] fuses the pixels of the two views  $k$  and  $k'$ . The coalescence operation concatenates two sets of coordinates and fuses pixels with the same position, preserving only the pixel values of the points that are nearest to the camera. We use  $\{\tilde{D}_k, \tilde{D}_{k'}\}$  to determine the distance of the points.

### Coalescence

The objective of the coalescence is to remove duplicate pixels. We implemented it to be executed on GPU. Algorithm 1 details our implementation. Input  $\hat{\mathbf{I}}$  to coalescence is the concatenation of the two morphed images as coordinates  $\hat{\mathbf{I}} = [\hat{\mathbf{I}}_{k,\alpha}, \hat{\mathbf{I}}_{k',\alpha}]$ . The associated depths  $\mathbf{v} = [\tilde{D}_k, \tilde{D}_{k'}]$  are instead the values.

To simplify formulation, we reconvert the minimum case to the case of selecting the coordinates with the maximum value. Coalescence's first step is to project into a 1D space  $\hat{\mathbf{I}}$  and the coordinates and the associated value  $\mathbf{v}$ . We then sort the two

projections  $\mathbf{H}$  and  $\mathbf{T}$ . This allows us to filter only the more significant point of a pixel.

---

**Algorithm 1** Coalescence algorithm

---

**Input:** Concatenation of the two morphed images as pixel coordinates  $\hat{\mathbf{I}} = [\hat{\mathbf{I}}_{k,\alpha}, \hat{\mathbf{I}}_{k',\alpha}]$ , such as  $((1, 0), (0, 2), \dots, (n, m))$ . The depth associated with each pixel coordinate  $\mathbf{v} = [\tilde{\mathbf{D}}_k, \tilde{\mathbf{D}}_{k'}]$ .

**Variables:**  $\mathbf{v}_{max}$  indicates the max value of  $\mathbf{v}$ . While  $\hat{\mathbf{I}}_{max}$  and  $\hat{\mathbf{I}}_{min}$  indicate respectively the minimum and maximum along dimension 0.  $\mathbf{H}$  and  $\mathbf{T}$  indicate the two created 1D projections.  $\mathbf{M}$  is the resulting coordinates mask.

**Output:** Mask  $\mathbf{M}[\mathbf{B}]$  to filter the coordinates, leaving for each pixel only the one with the lowest depth.

**procedure**  $\psi(\hat{\mathbf{I}}, \mathbf{v})$

$\mathbf{v}_{max} \leftarrow \max(\mathbf{v})$   $\triangleright$  Reconduct the min number reduction to a max number reduction

$\mathbf{v} \leftarrow \mathbf{v}_{max} - \mathbf{v}$

$\hat{\mathbf{I}}_{min} \leftarrow \min(\hat{\mathbf{I}}, \dim = 0)$   $\triangleright$  Move each axis independently to start at zero

$\hat{\mathbf{I}} \leftarrow \hat{\mathbf{I}} - \hat{\mathbf{I}}_{min}$

$\hat{\mathbf{I}}_{max} \leftarrow \max(\hat{\mathbf{I}}, \dim = 0) + 1$   $\triangleright$  Compute the amount of space in the projection to avoid overlap

$\mathbf{F} \leftarrow \text{cumprod}(\text{flip}(\mathbf{I}_{max}, \dim = 1), \dim = 1)$

$\mathbf{H} \leftarrow \text{sum}(\mathbf{I} \odot \mathbf{M}, \dim = 1)$   $\triangleright$  Projection of coordinates in a 1D space

$\mathbf{v}_{max} \leftarrow \max(\mathbf{v})$   $\triangleright$  Maximum value to build the combined projection

$\mathbf{T} \leftarrow \mathbf{H}\mathbf{v}_{max} + \mathbf{v}$   $\triangleright$  Projection of coordinates and value in a 1D space

$\mathbf{B} \leftarrow \text{argsort}(\mathbf{T})$

$\mathbf{H} \leftarrow \mathbf{H}[\mathbf{B}]$

$\mathbf{M} \leftarrow \mathbf{H}[1:] > \mathbf{H}[: -1]$

$\mathbf{B}^{-1}[\mathbf{B}] \leftarrow (0 \dots n)$   $\triangleright$  Return the mask but it is necessary to reverse argsort operation

**return**  $\mathbf{M}[\mathbf{B}^{-1}]$

---

### Training with VM-NeRF

VM-NeRF is subject to the same geometric constraints as the original view morphing technique [6]. These constraints impose that singular camera configurations should not exist. These configurations happen whenever the optic center of a camera is within the field of view of another one [6]. We also discard cameras that are distant from each other more than a threshold  $\gamma$ , as the morphed cameras may be on a transition path that crosses regions where the object of interest is not actually visible (so being rather useless for training a NeRF based model).

Because view morphing allows the synthesis of a new view at any point on the line that connects the known camera pair, we randomly sample new views using a Gaussian distribution centered halfway through the camera pair. Specifically, let us consider a normalized distance between the two cameras. The Gaussian distribution is centered

at 0.5 and the standard deviation  $\sigma$  is chosen such that  $3\sigma \rightarrow \epsilon$  at the optic center positions. Therefore, we sample  $\alpha \sim \mathcal{N}(0.5, \sigma)$  with  $0 \leq \alpha \leq 1$ . The depth information that NeRF can render at the first few iterations is inherently noisy, therefore, we let NeRF warm up on the known views for  $\lambda$  iterations before synthesizing and injecting VM-NeRF views in the next training iterations. After the warm-up phase, for each valid camera pair, we regenerate  $M$  new views every  $\eta$  training iterations as the predicted depth quality improves over time during training.

### 3.1.2 Results

#### Experimental setup

We evaluate our method on three training setups using the NeRF realistic synthetic 360° dataset [2], which is composed of eight scenes, i.e. Chair, Drums, Ficus, Lego, Materials, Ship, Mic, Hot Dog. **First setup:** We select  $N = 8$  views out of 100 available for each scene using the Farthest Point Sampling (FPS) [238] (the first view is used for FPS initialization in each scene). **Second setup:** we use the same  $N = 8$  views used in DietNeRF [11]. **Third setup:** we select  $N = 4$  views using the previous FPS approach. We test each trained model on all the test views of NeRF realistic synthetic 360°. We quantify the rendering results using the peak signal-to-noise ratio (PSNR) score, the structured similarity index measure (SSIM) [239] and the learned perceptual image patch similarity (LPIPS) [240]. We quantitatively compare our approach against DietNeRF [11] and RegNeRF [12] as the most recent methods for few-shot view synthesis. We also compare against AugNeRF [10] because it is the only data augmentation for NeRF, and data augmentation can be a useful strategy to promote generalization. We choose to use the Chair scene for our ablation study, which consists of testing VM-NeRF on four different, randomly-chosen, configurations of eight views and on the DietNeRF configuration.

We implement NeRF and our approach in PyTorch Lightning, and run experiments on a single Nvidia A40 with a batch size of 1024 rays. A single scene can be trained in about two days. We use the original implementations of DietNeRF, AugNeRF and RegNeRF to evaluate the different setups. We set the same training parameters as in [2], and set  $\gamma = 6$ ,  $\sigma = 0.2$ ,  $M = 1$ ,  $\eta = 5$ ,  $\lambda = 500$ .

#### Analysis of the results

##### Quantitative.

Tab. 3.1 shows the results averaged over the eight scenes. Our NeRF implementation can achieve nearly the same results reported in [2] on the 100-view setup, i.e. PSNR equal to 31.21 (ours) compared to 31.01 [2].

VM-NeRF outperforms all the other methods in the eight-view setting. Interestingly, the original version of NeRF is the one that performs as second best, followed by DietNeRF and RegNeRF. AugNeRF fails to produce satisfactory results. We also observe that VM-NeRF achieves slightly better quality than its version with oracle depth

### 3.1. View Morphing for Few-Viewpoint Neural Radiance Fields

**Table 3.1** Results on the NeRF realistic synthetic 360° dataset.

| # views | Method        | PSNR $\uparrow$ | SSIM $\uparrow$ | LPIPS $\downarrow$ |
|---------|---------------|-----------------|-----------------|--------------------|
| 100 [2] | NeRF [2]      | 31.21           | 0.9513          | 0.0465             |
| 8       | NeRF [2]      | <u>23.45</u>    | 0.8673          | 0.1303             |
|         | DietNeRF [11] | 22.98           | 0.8545          | <u>0.1258</u>      |
|         | AugNeRF [10]  | 10.04           | 0.5415          | 0.3866             |
|         | RegNeRF [12]  | 22.91           | <u>0.8756</u>   | 0.1138             |
|         | VM-NeRF       | <b>24.39</b>    | <b>0.8768</b>   | <b>0.1146</b>      |
| 8 [11]  | NeRF [11]     | 20.09           | 0.8220          | 0.1790             |
|         | DietNeRF [11] | <u>23.59</u>    | <u>0.8740</u>   | <b>0.0970</b>      |
|         | VM-NeRF       | <b>24.14</b>    | <b>0.8729</b>   | <u>0.1180</u>      |
| 4       | NeRF [2]      | 10.98           | 0.6550          | 0.3620             |
|         | DietNeRF [11] | 12.61           | 0.6591          | 0.3302             |
|         | AugNeRF [10]  | 8.14            | 0.3924          | 0.4802             |
|         | RegNeRF [12]  | <u>15.88</u>    | <b>0.7932</b>   | <b>0.1994</b>      |
|         | VM-NeRF       | <b>16.90</b>    | <u>0.7563</u>   | <u>0.2461</u>      |

**Table 3.2** Ablation study results. Keys: # views: represent the number of views and the relative subset. avg. dist.: average distance between view pairs.

| # views | avg. dist. | Method        | PSNR $\uparrow$ | SSIM $\uparrow$ | LPIPS $\downarrow$ |
|---------|------------|---------------|-----------------|-----------------|--------------------|
| 8 [11]  | 5.20       | DietNeRF [11] | 25.59           | 0.9120          | 0.0770             |
| 8 [11]  | 5.20       | VM-NeRF       | 26.90           | 0.9180          | 0.0797             |
| 8 (s1)  | 5.18       | VM-NeRF       | 27.87           | 0.9294          | 0.0693             |
| 8 (s2)  | 4.38       | VM-NeRF       | 27.13           | 0.9178          | 0.0855             |
| 8 (s3)  | 4.73       | VM-NeRF       | 27.48           | 0.9317          | 0.0676             |
| 8 (s4)  | 4.69       | VM-NeRF       | 28.39           | 0.9365          | 0.0598             |



**Table 3.3** SSIM ( $\uparrow$ ) metric per-scene results on the NeRF realistic synthetic 360° dataset. Best-performing results are highlighted in **bold** and second-best results are underlined.

| # views | Method                      | Chair         | Drums         | Ficus         | Hot Dog       | Lego          | Materials     | Mic           | Ship          | Avg           |
|---------|-----------------------------|---------------|---------------|---------------|---------------|---------------|---------------|---------------|---------------|---------------|
| 100     | NeRF [2]                    | 0.9790        | 0.9288        | 0.9627        | 0.9785        | 0.9610        | 0.9557        | 0.9753        | 0.8694        | 0.9513        |
| 8       | NeRF [2] w/<br>oracle image | 0.9505        | 0.8748        | 0.8789        | 0.9606        | 0.9129        | 0.8870        | 0.9625        | 0.8031        | 0.9038        |
|         | VM-NeRF w/<br>oracle depth  | 0.9418        | 0.8631        | 0.8753        | 0.9249        | 0.8633        | 0.8420        | 0.9498        | 0.7796        | 0.8800        |
| 8       | NeRF [2]                    | <u>0.8829</u> | <b>0.8320</b> | <b>0.8720</b> | <u>0.8894</u> | 0.8820        | 0.8561        | 0.9540        | 0.7696        | 0.8673        |
|         | DietNeRF [11]               | 0.9079        | 0.8020        | 0.8664        | 0.9101        | 0.8033        | 0.8453        | 0.9496        | 0.7515        | 0.8545        |
|         | AugNeRF [10]                | 0.7576        | 0.0004        | 0.7760        | 0.7480        | 0.0012        | 0.6653        | 0.7497        | 0.6340        | 0.5415        |
|         | RegNeRF [12]                | 0.9074        | <u>0.8050</u> | <u>0.8664</u> | <b>0.9268</b> | <b>0.8984</b> | <b>0.8667</b> | <b>0.9601</b> | <u>0.7742</u> | <u>0.8756</u> |
|         | VM-NeRF (ours)              | <b>0.9294</b> | 0.8302        | 0.8650        | 0.9266        | <u>0.8790</u> | <u>0.8576</u> | <u>0.9508</u> | <b>0.7758</b> | <b>0.8768</b> |
| 4       | NeRF [2]                    | 0.8156        | 0.5657        | 0.8207        | 0.6507        | 0.5765        | 0.4765        | 0.8427        | 0.4912        | 0.6550        |
|         | DietNeRF [11]               | 0.8130        | 0.5231        | 0.8195        | 0.6416        | 0.5249        | 0.7632        | 0.7187        | <u>0.7264</u> | 0.6591        |
|         | AugNeRF [10]                | 0.6669        | 0.5330        | 0.7340        | 0.0009        | 0.5459        | 0.0008        | 0.6563        | 0.0011        | 0.3924        |
|         | RegNeRF [12]                | <u>0.8237</u> | <b>0.7450</b> | <u>0.8239</u> | <b>0.8341</b> | <u>0.7414</u> | <u>0.7792</u> | <b>0.9391</b> | 0.6593        | <b>0.7932</b> |
|         | VM-NeRF (ours)              | <b>0.8439</b> | <u>0.7327</u> | <b>0.8247</b> | <u>0.7237</u> | <b>0.7737</b> | <b>0.7953</b> | <u>0.8819</u> | 0.4744        | <u>0.7563</u> |

maps, i.e. 24.39 vs. 24.22 PSNR. In fact, we observe that VM-NeRF effectively leverages the depth information that is estimated during training, although it is noisy. We also evaluate VM-NeRF on the same eight views originally tested by DietNeRF [11]. Also here we achieve higher quality results on average, i.e. 24.14 vs. 23.59. We also improve in the four-view setup where we obtain an improvement of +1.02 PSNR on average. The results also show that the perturbation of the known input views, done by AugNeRF, has adverse effects in all the tested setups.

The numerical results for each scene are reported in Tab. 3.3 for SSIM, and Tab. 3.4 for LPIPS. In terms of SSIM and LPIPS, we can see that our approach is typically on par with NeRF and DietNeRF in the case of 8-views setting. This is mainly due to how these two measures are formulated that are less sensitive to high-frequency variations, but they rather aim at measuring the perceptual similarity between two images. In the case of 4-views setting our approach often outperforms the other approaches as it allows NeRF to correctly learn the geometric structure of the scene.

**Qualitative.** Fig. 3.4 presents qualitative results on several scenes (Chair, Hot Dog, and Lego), demonstrating that VM-NeRF produces renderings with notably finer details compared to DietNeRF. This visual improvement can be attributed to fundamental differences in the approaches. While DietNeRF employs a CLIP-based semantic consistency loss for regularization [145], this method relies on a low-dimensional (global) representation vector of the image, which may inadvertently sacrifice fine-grained details during the encoding process. In contrast, our VM-NeRF approach directly interpolates the original photometric information from view pairs to synthesize new training examples, thereby preserving high-frequency details without the information bottleneck created by low-dimensional representation vectors.

Both Figs. 3.4 and 3.5 further illustrate that VM-NeRF produces fewer artifacts compared to RegNeRF by effectively leveraging geometric correlations between nearby

### 3.1. View Morphing for Few-Viewpoint Neural Radiance Fields

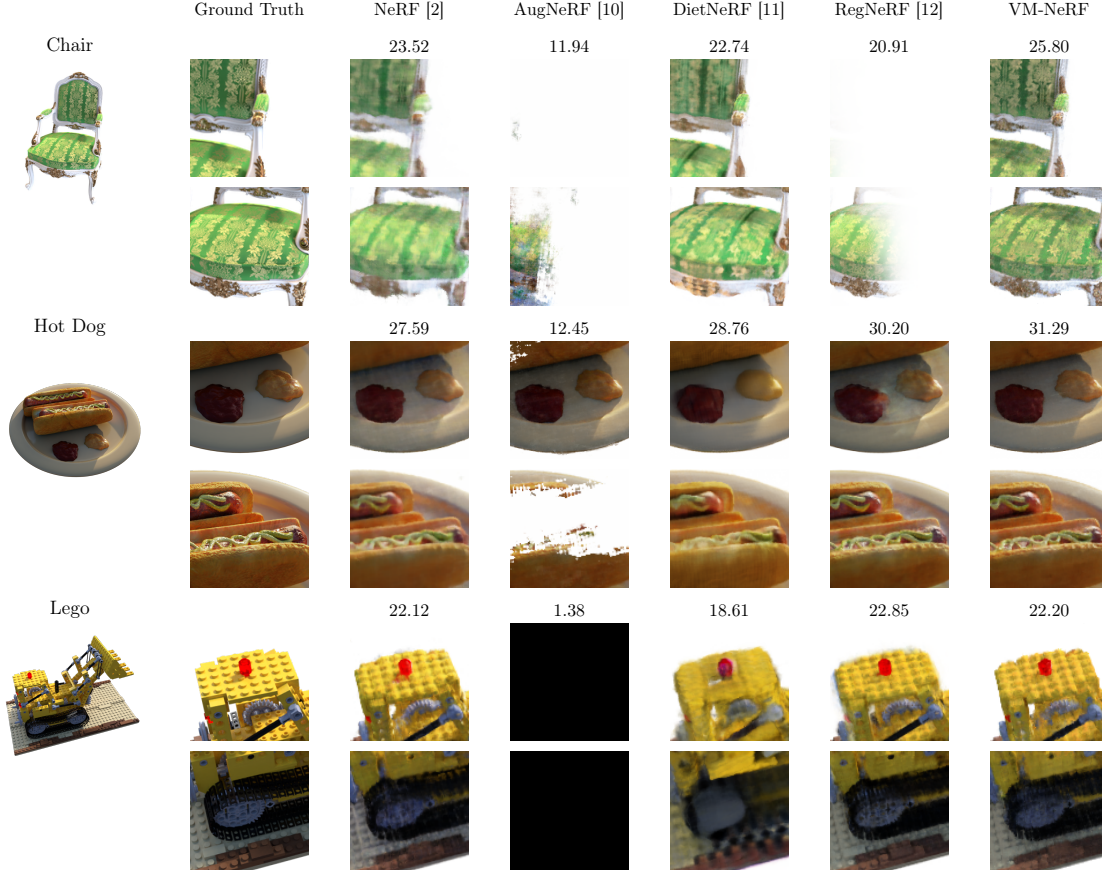


Figure 3.4: Comparisons on test-set views of scenes of NeRF realistic synthetic 360°. Unlike AugNeRF [10], VM-NeRF is an effective method that can be used for few-shot view synthesis problems. Unlike DietNeRF [11], VM-NeRF enables NeRF to learn scenes with a higher definition. VM-NeRF helps to produce fewer artifacts than RegNeRF during rendering [12]. We report the PSNR that we measured for each method and for each rendered image. AugNeRF unsuccessfully learns Chair and Lego (white and black outputs).

views. These visualizations clearly demonstrate how our proposed method enhances NeRF’s ability to learn comprehensive scene representations from sparse inputs. The advantage of VM-NeRF becomes particularly evident in extremely challenging scenarios with very wide baselines between input views, as shown in Fig. 3.7. In the four-view setup, VM-NeRF successfully reconstructs the correct geometry of complex scenes such as Mic, Lego, and Drums, where DietNeRF notably fails. This demonstrates VM-NeRF’s superior capability to generalize from extremely limited viewpoints by exploiting the geometric constraints provided by our view morphing approach. We should note, however, that certain scenes like Ship and Hot Dog remain challenging for all methods in this extremely sparse setting due to their complex geometry and appearance, highlighting the intrinsic difficulty of the few-shot view synthesis problem.

**Ablation study** We assess the stability of VM-NeRF by evaluating the rendering

quality when different combinations of views are used to train NeRF. Tab. 3.2 shows that the performance is fairly stable throughout different view configurations. We also observe that the algorithm is robust to variations in the distance between view pairs. As long as a view pair is not singular and the distance between cameras is adequate to create acceptable 3D projective transformations of the object, we can successfully synthesize new views with VM-NeRF.

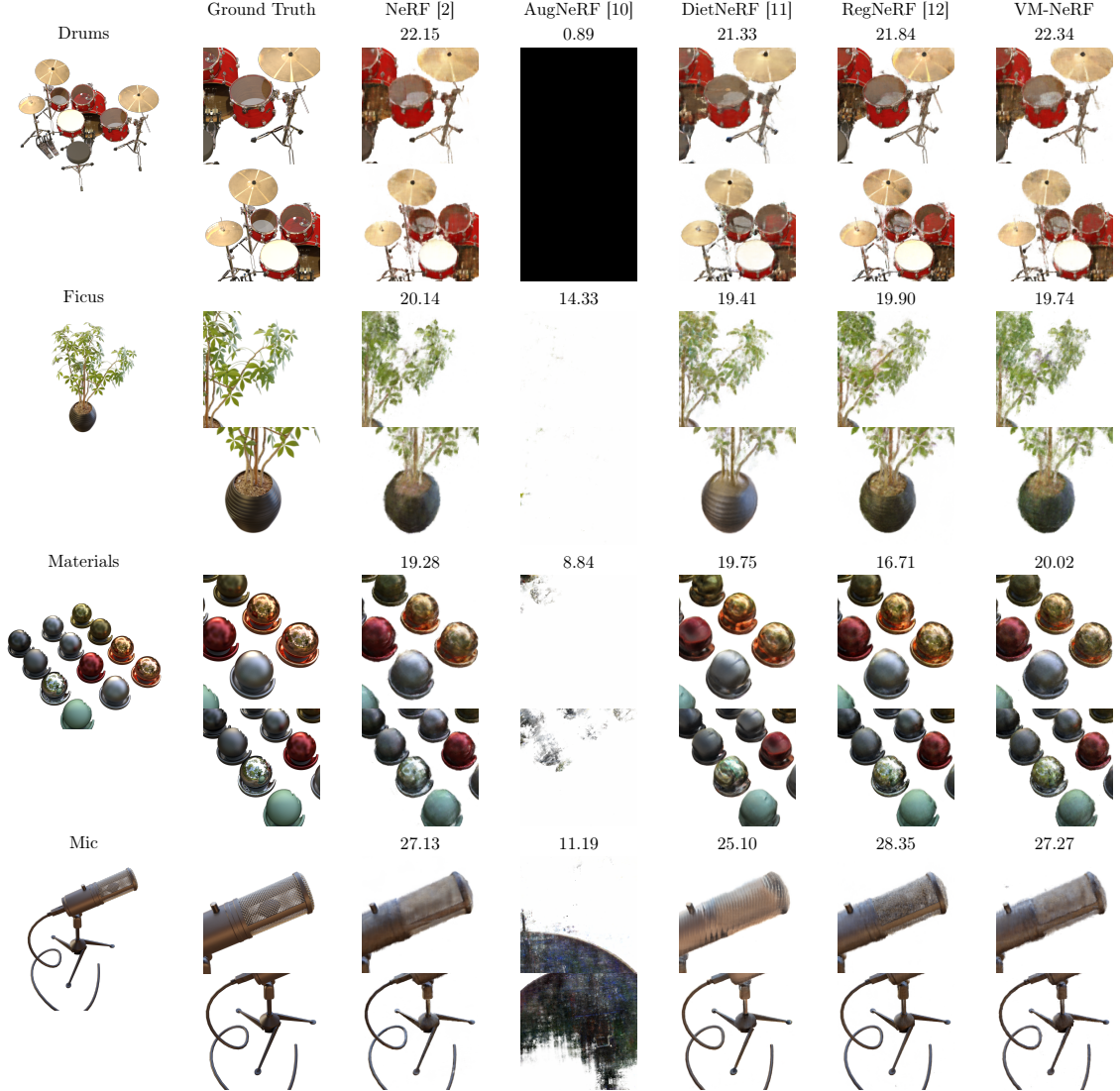


Figure 3.5: Comparisons on test-set views of scenes of NeRF realistic synthetic 360°, trained on 8-views. Unlike AugNeRF [10], VM-NeRF is an effective method that can be used for few-shot view synthesis problems. Unlike DietNeRF [11], VM-NeRF enables NeRF to learn scenes with a higher definition. VM-NeRF helps to preserve high-frequency details and geometric consistency. We report the PSNR that we measured for each method and for each rendered image.

## 3.2 Chapter conclusions

This chapter has presented VM-NeRF, a novel method for few-shot view synthesis that integrates Neural Radiance Fields with the classical View Morphing technique. By leveraging the geometric principles of projective transformations, we have addressed one of the fundamental challenges in making Neural Radiance Fields viable for robotic perception systems—the ability to generate high-quality novel views from a sparse set of input images. The approach we developed eliminates the need for manual feature matching typically required by traditional View Morphing by utilizing the depth information estimated by NeRF. This innovation allows us to establish dense correspondences between views automatically, creating a seamless integration of geometric constraints into the neural rendering pipeline. We have demonstrated that our method can effectively train NeRF models with as few as 4-8 input views, compared to the hundreds typically required. Through extensive experiments on standard benchmark datasets, we have shown that VM-NeRF consistently outperforms state-of-the-art few-shot view synthesis methods including DietNeRF, RegNeRF, and AugNeRF. The quantitative results demonstrate improvements of up to 1.8dB PSNR in the eight-view setting and 1.0dB PSNR in the more challenging four-view setting. Qualitatively, our approach produces renderings with finer details and fewer artifacts by preserving high-frequency information through direct photometric interpolation rather than relying on low-dimensional representation vectors.

The contributions presented in this chapter represent an important step toward making neural rendering techniques practical for robotic applications where collecting numerous views is often infeasible. By reducing the required number of input images while maintaining rendering quality, we have shown that geometrically-informed neural rendering can overcome key limitations of purely learning-based approaches. While VM-NeRF shows promising results, it still has limitations that could be addressed in future work. Current approach only interpolate along the line connecting the optical centers of each camera pair, meaning it cannot fully reconstruct objects that are only partially visible in the training views. Finally, exploring how this approach could be extended to dynamic scenes would be particularly relevant for real-world robotic applications that operate in changing environments.

**Table 3.4** LPIPS ( $\downarrow$ ) metric per-scene results on the NeRF realistic synthetic 360° dataset. Best-performing results are highlighted in **bold** and second-best results are underlined.

| # views | Method                      | Chair         | Drums         | Ficus         | Hot Dog       | Lego          | Materials     | Mic           | Ship          | Avg           |
|---------|-----------------------------|---------------|---------------|---------------|---------------|---------------|---------------|---------------|---------------|---------------|
| 100     | NeRF [2]                    | 0.0225        | 0.0621        | 0.0339        | 0.0286        | 0.0347        | 0.0356        | 0.0254        | 0.1295        | <u>0.0465</u> |
| 8       | NeRF [2] w/<br>oracle image | 0.0512        | 0.1067        | 0.1128        | 0.0498        | 0.0690        | 0.0830        | 0.0362        | 0.2060        | <u>0.0893</u> |
|         | VM-NeRF w/<br>oracle depth  | 0.0569        | 0.1139        | 0.1242        | 0.0862        | 0.1131        | 0.1151        | 0.0520        | 0.1994        | <u>0.1076</u> |
| 8       | NeRF [2]                    | 0.1286        | <b>0.1623</b> | <u>0.1225</u> | 0.1380        | <u>0.0962</u> | 0.1197        | 0.0505        | 0.2247        | <u>0.1303</u> |
|         | DietNeRF [11]               | <u>0.0857</u> | 0.1696        | <b>0.1029</b> | 0.1039        | 0.1638        | <b>0.1006</b> | <u>0.0473</u> | 0.2323        | <u>0.1258</u> |
|         | AugNeRF [10]                | 0.2871        | 0.6423        | 0.2416        | 0.3042        | 0.6556        | 0.3360        | 0.3064        | 0.3195        | <u>0.3866</u> |
|         | RegNeRF [12]                | 0.1049        | 0.1836        | 0.1241        | <b>0.0738</b> | <b>0.0883</b> | <u>0.1030</u> | <b>0.0399</b> | <u>0.1924</u> | <b>0.1138</b> |
|         | VM-NeRF (ours)              | <b>0.0693</b> | <u>0.1672</u> | 0.1243        | <u>0.0795</u> | 0.0985        | 0.1144        | 0.0657        | <b>0.1980</b> | <u>0.1146</u> |
| 4       | NeRF [2]                    | 0.2267        | 0.4545        | 0.1822        | 0.4125        | 0.4285        | 0.4977        | 0.1967        | 0.4971        | <u>0.3620</u> |
|         | DietNeRF [11]               | <u>0.1676</u> | 0.4652        | <b>0.1477</b> | 0.4094        | 0.4481        | 0.1879        | 0.3150        | 0.5009        | <u>0.3302</u> |
|         | AugNeRF [10]                | 0.3820        | 0.4684        | 0.3016        | 0.6024        | 0.4561        | 0.6102        | 0.3875        | 0.6336        | <u>0.4802</u> |
|         | RegNeRF [12]                | 0.2075        | <u>0.2750</u> | 0.1760        | <b>0.1688</b> | <u>0.2429</u> | <u>0.1801</u> | <b>0.0653</b> | <b>0.2798</b> | <b>0.1994</b> |
|         | VM-NeRF (ours)              | <b>0.1628</b> | <b>0.2657</b> | <u>0.1575</u> | <u>0.3527</u> | <b>0.1936</b> | <b>0.1771</b> | <u>0.1696</u> | <u>0.4899</u> | <u>0.2461</u> |

### 3.2. Chapter conclusions



Figure 3.6: Comparisons on test-set views of scenes of NeRF realistic synthetic 360°, trained on 4-views. Unlike other techniques VM-NeRF help to increase the probability of building a consistent implicit representation with very sparse view. We report the PSNR that we measured for each method and for each rendered image.



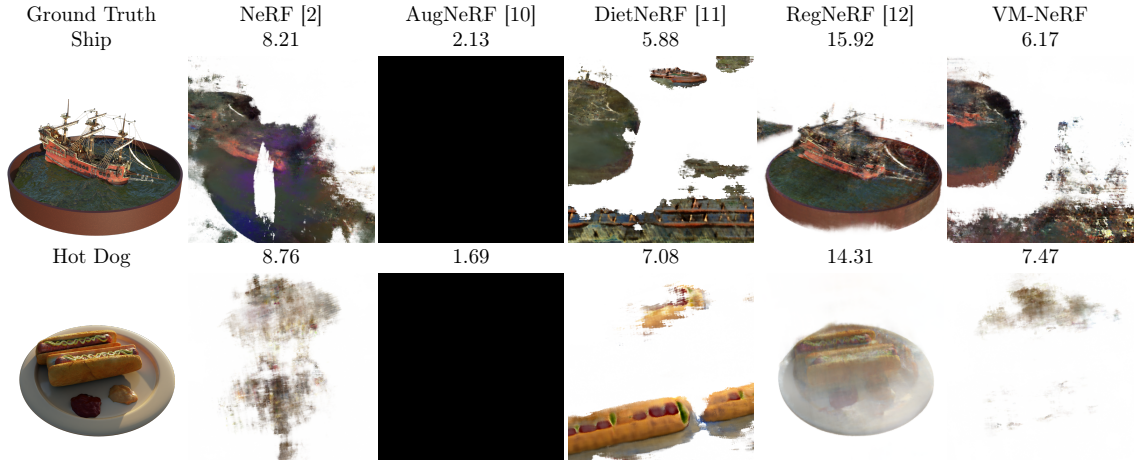


Figure 3.7: Comparisons on test-set views of scenes of NeRF realistic synthetic 360°, trained on 4-views. Unlike other techniques VM-NeRF help to increase the probability of building a consistent implicit representation with very sparse view. We report the PSNR that we measured for each method and for each rendered image.





## Chapter 4

# 6DoF Pose Estimation for Novel View Synthesis

Accurate 6DoF pose estimation is a critical capability for autonomous robots across both industrial and domestic settings [232, 241–243]. As discussed in Chapter 1, traditional pose estimation approaches face significant limitations when dealing with objects featuring reflective, transparent, or highly specular surfaces—precisely the types of objects commonly encountered in everyday environments. This chapter presents methods that leverage Novel View Synthesis (NVS) representations to overcome these limitations through camera relocalization: estimating the camera pose of a query image given a pre-trained NVS model. Through duality we can then also apply this techniques to object localization.

## Camera Relocalization and Object Localization

As formally defined in Sec. 1.2.2, camera relocalization addresses the following problem: given a trained NVS model  $f_{\Xi}$  (NeRF or 3DGS) and a query image  $\mathbf{I}_{\text{query}}$ , estimate the camera pose  $\hat{\mathbf{P}}_{\text{query}}$  that best explains the observed image. This inverse problem—recovering the viewpoint from which an image was captured—is fundamentally different from the forward rendering problem (generating an image from a known pose) that NVS methods are trained to solve.

While camera relocalization has traditionally been motivated by applications in augmented reality and robot navigation, it has profound implications for robotic manipulation through the camera-object duality (Eq. 1.3). When the NVS model represents a single object of interest (common in robotic grasping scenarios), estimating the camera pose relative to that object-centric scene representation is mathematically equivalent to estimating the object’s pose in the camera frame:

$${}^c\mathbf{T}_{\mathcal{O}} = ({}^{\mathcal{O}}\mathbf{T}_{\mathcal{C}})^{-1},$$

where  ${}^c\mathbf{T}_{\mathcal{O}}$  is the object pose in the camera frame and  ${}^{\mathcal{O}}\mathbf{T}_{\mathcal{C}}$  is the camera pose in the object-centric reference frame.

This duality enables robust object localization for manipulation without requiring explicit depth sensors or CAD models—directly addressing the limitations of traditional pose estimation methods when dealing with optically challenging objects. NVS-based representations (NeRF and 3DGS) can model complex appearance phenomena including view-dependent effects, reflections, and transparency that confound depth cameras, making camera relocalization particularly valuable for robotic applications.

## Chapter Overview and Contributions

We present two complementary approaches that address the camera relocalization problem using different NVS frameworks, each eliminating the need for initial pose estimates that plague traditional analysis-by-synthesis methods [214].

First, in Section 4.1, we introduce IFFNeRF (Initialization-Free Formulation for NeRF), a method for estimating camera pose from a single image using a pre-trained NeRF model. IFFNeRF employs Metropolis-Hastings sampling to generate candidate rays from the NeRF volume and uses attention mechanisms to correlate these rays with image features, enabling direct pose estimation without iterative optimization. The method achieves near real-time performance (34fps) while improving accuracy by 80.1% (angular error) and 67.3% (translation error) compared to existing NeRF-based approaches.

Then, in Section 4.2, we present 6DGS, which reformulates the camera relocalization problem within the 3D Gaussian Splatting (3DGS) framework. 6DGS introduces a novel Ellicell ray-casting procedure that uniformly radiates rays from ellipsoid surfaces and employs attention-based ray selection to estimate pose in closed form. This approach achieves near real-time performance (15fps) with improved accuracy compared to both 3DGS-based and NeRF-based methods, demonstrating  $38\times$  speedup over previous Gaussian-based approaches.

Both methods demonstrate that NVS-based camera relocalization provides a viable path to robust object localization in robotic manipulation, particularly for objects where traditional depth-based methods fail. The estimated poses can be directly integrated into downstream manipulation pipelines, as demonstrated in Chapter 5, where object localization enables grasp prediction for dexterous manipulation.

Additionally, we demonstrate that the same 6DGS framework can be applied to visual relocalization in larger-scale environments, showing the generality of the approach beyond single-object scenarios.

## 4.1 6DoF using Neural Radiance Fields

NeRF-based approaches can encode a broader range of appearance variations and eliminate the need for any mesh reconstruction [85]. However, utilizing these models for subsequent pose estimation in 6DoF presents significant challenges. They typically demand an initial pose guess and rely on time-intensive optimization processes [169, 214]. iNeRF [214] pioneered 6DoF pose estimation using NeRF through an analysis-by-synthesis

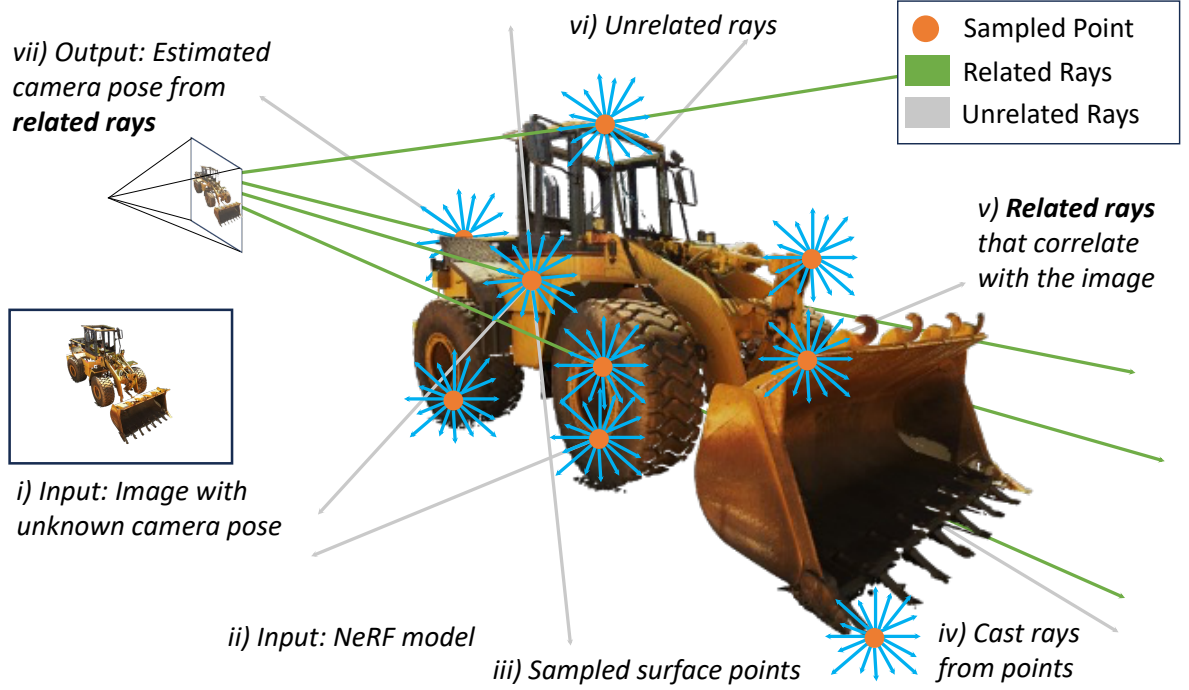


Figure 4.1: From (i) a given image with an unknown pose and (ii) a NeRF model, we recover the pose by first (iii) sampling surface points using Metropolis-Hasting algorithm and (iv) casting rays from them in isocell distribution. We then (iv/v) correlate rays with the image to identify relevant rays using attention and (vii) recover the unknown 6DoF camera pose.

method. Because NeRF can render an image of a novel view based on a specified pose, the problem can be inverted to deduce the pose from an uncalibrated image. This optimization process entails minimizing the photometric error between the supplied image and the ones rendered from potential poses, iterating until error is minimized.

While effectively addressing the pose estimation problem, iNeRF and similar methods are computationally expensive, requiring multiple iterations to converge to a solution. Furthermore, they are sensitive to the initial pose guess, which must be close to the true solution for the optimization process to succeed. In this section, we introduce IFFNeRF (Initialization-Free Formulation for Neural Radiance Fields), a novel approach to estimate the 6DoF camera pose of a given image, building on the NeRF formulation. Unlike existing methods, IFFNeRF is designed to operate in real-time and eliminates the need for an initial pose guess that is proximate to the sought solution.

Our approach, IFFNeRF utilizes the Metropolis-Hasting algorithm to sample surface points from within the NeRF model. Then IFFNeRF generates rays from surface points sampled within the NeRF volume, with each ray characterized by its origin, direction, and an RGB value obtained through pixel-level view synthesis. To efficiently identify rays that are most relevant to the query image, we employ a transformer-based attention mechanism [57] that computes a similarity score between the embeddings of the query image and each cast ray. This allows us to select a subset of rays with high relevance

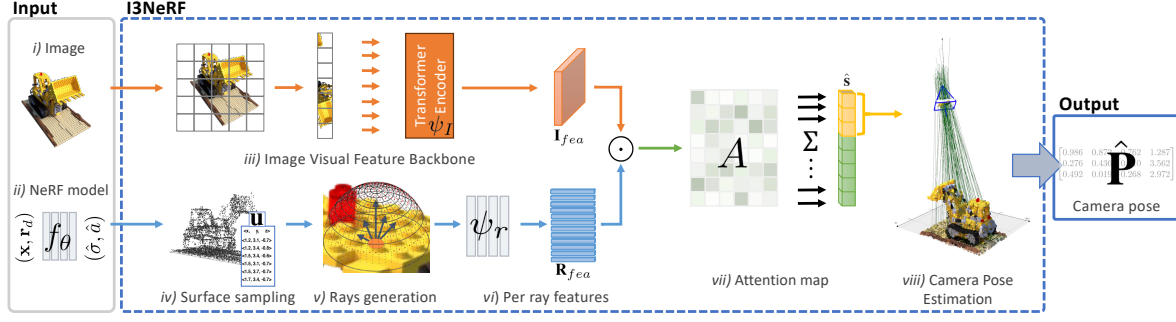


Figure 4.2: Our IFFNeRF takes as input an (i) image, and (ii) a NeRF model. The model  $\psi_I$  encodes the image (iii) using a visual backbone. As for the NeRF model, we sample surface points (iv) using Metropolis-Hastings to identify candidates on the surface  $\mathbf{u}$ , which can act as locations to project rays. We uniformly project rays (v) from the center isocells from the points, and estimate the ray corresponding parameters, color, and normal, using the NeRF model. We then embed the ray representation  $\psi_r$  (vi) and (vii) learn attention  $A$  between the ray embedding  $I_{fea}$  and  $R_{fea}$  to rank rays in relation to the image. We select the top-N rays and estimate the camera location (viii) using least squares, resulting in a 6DoF pose  $\hat{\mathbf{P}}$  for the image.

scores, which are then used to determine the camera pose by solving a weighted Least Squares problem in closed form. A visualization of this key idea is shown in Fig. 4.1.

Evaluations on both synthetic and real-world datasets demonstrate that IFFNeRF consistently outperforms previous methods, particularly in challenging scenarios where no initial pose estimate is available or when initialization is random. Through comprehensive evaluation on both synthetic and real-world datasets, we demonstrate that our method improves the angular and translation error accuracy by **80.1%** and **67.3%**, respectively, compared to iNeRF while performing at 34fps on consumer hardware without requiring an initial pose guess. Through ablation studies, we thoroughly analyze the impact of various components including the attention mechanism, feature extraction backbone, and initialization strategies on the overall performance, providing insights into the method’s behavior across different configurations.

The key contributions of our work can be summarized as:

- A novel real-time method for estimating 6DoF camera pose from a single image using a pre-trained NeRF model;
- An initialization-free approach that eliminates the need for approximate pose estimates, overcoming a significant limitation of existing methods;
- A transformer-based attention mechanism that effectively correlates image and ray embeddings to identify the most relevant rays for pose estimation.

### 4.1.1 Approach

IFFNeRF aims to predict the camera pose  $\hat{\mathbf{P}} \in \text{SE}(3)$  given an observed image  $\mathbf{I}$  and a pre-computed NeRF model  $f_{\Xi}$  (as define in Sec. 2.1.3).

We firstly apply a Metropolis-Hastings algorithm to sample  $G$  surface points within the scene volume (Sec. 4.1.1), then we cast  $V$  rays  $\tilde{\mathbf{r}}$  from an isocell at each surface point (Sec. 4.1.1), thus obtaining  $\mathbf{R}$  containing  $G \times V$  rays. We then learn an attention map  $A$  between embeddings of the image  $\psi_I(\mathbf{I})$  and generated rays  $\psi_r(\mathbf{R})$  (Sec. 4.1.1). Based on the information contained in the attention map, we select a subset of candidate rays that are likely to fall within the image. Finally, to recover  $\hat{\mathbf{P}}$  at test time, we optimize using Least Squares over the selected rays (Sec. 4.1.1). A visual overview of the IFFNeRF approach is shown in Fig. 4.2.

#### Surface point sampling

We begin by extracting from the 3D points  $\mathbf{x}$  a number  $G$  of points  $\mathbf{u}$  on the surface of the object. To achieve this, we employ the Metropolis-Hastings (M-H) algorithm. M-H works by sampling from a distribution  $\mathcal{Q}$ , in our case this distribution is the NeRF density output  $\mathcal{Q} = \hat{\sigma}$  [244]. We bootstrap the M-H algorithm by uniformly sampling points inside the bounding box that encloses the object’s surface:

$$\mathbf{u}_0 = \boldsymbol{\beta} \otimes (\mathbf{B}_{max} - \mathbf{B}_{min}) + \mathbb{1}_G \otimes \mathbf{B}_{min}, \quad (4.1)$$

where  $\mathbf{u}_0 \in \mathbb{R}^{G \times 3}$  are the sampled points at iteration 0,  $\boldsymbol{\beta} = [\beta_0, \beta_1, \dots, \beta_{G-1}]^T$  with each  $\beta \sim \mathcal{U}[0, 1]$ ,  $\mathbb{1}_G$  is a column vector of ones of length  $G$ ,  $\otimes$  is the Kronecker product, and  $\mathbf{B}_{max}, \mathbf{B}_{min} \in \mathbb{R}^{1 \times 3}$  are the maximum and minimum 3D coordinates of the scene bounding box, respectively.

The M-H algorithm moves the point  $g$ , with  $g = 1 \dots G$ , only if  $\mathcal{Q}(\mathbf{u}_{i+1,g}) \geq q$ , where  $q$  is the acceptance ratio. If the condition is satisfied,  $\mathbf{u}_{i+1,g}$  is a newly sampled random position. We define  $q$  as dependent from the Cumulative Distribution Function (CDF)  $Pr$  of the previous iteration:

$$q = Pr[Q_{i-1} < Q_{i-1,g}] \geq 0.6, \quad (4.2)$$

that indicates  $q$  as the 60<sup>th</sup> percentile, of the density of the selected points at the previous iteration. We define the points that are sampled at the last iteration of M-H as  $\mathbf{u}$ . These points are the ray origin,  $\tilde{\mathbf{r}}_o$ , for the following ray generation step, where we indicate with  $\tilde{\mathbf{r}}$  the generated ray.

#### Rays generation from sampled surface points

From our sampled points  $\mathbf{u}$ , we cast a fixed number of rays in the direction oriented by the normal direction  $\hat{\mathbf{n}}(\mathbf{u}_g)$ , thus avoiding to cast rays toward the unobserved internal volume of the object. The normals  $\hat{\mathbf{n}}(\mathbf{u})$  are obtained by querying the NeRF model. We adopt a deterministic method termed as Isocell [245, 246], which, compared to other commonly used rays sampling approaches, *e.g.* Monte-Carlo [247], achieves higher

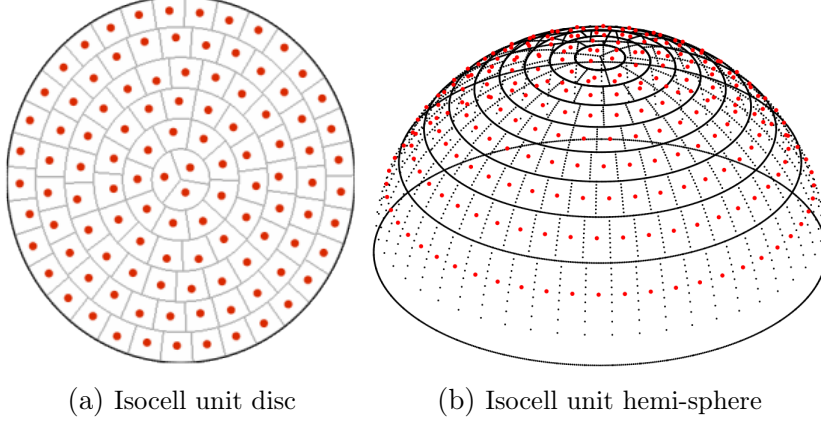


Figure 4.3: Illustration of the Isocell ray generation method over a circular domain of a unit disk and unit sphere. The generated points indicate the ray positions within the equally spaced circle cells (we will denote them as “cell centres” for simplicity).

precision with fewer rays [248, 249]. The uniform ray distribution of an isocell partitions the surface of a unit sphere into equal area cells, where the cells center forms a uniform distribution on the unit sphere (Fig. 4.3). The distribution of an isocell provides the direction of the generated rays, identified as  $\tilde{\mathbf{r}}_d$ . Therefore, we cast  $V$  rays for each point  $\mathbf{u}_g$ , as shown in Fig. 4.4. We set  $V = 27$  rays per isocell as we empirically found that this configuration works well in practice.

The generated rays represent a collection of potential hypotheses, meaning that a subset of them will intersect the target image  $\mathbf{I}$ . In total, we have  $N = V \times G$  generated rays  $\tilde{\mathbf{r}}$  across the sampled points. Each generated ray has an associated color  $\tilde{\mathbf{r}}_c$ , which is computed through the same pixel-level volumetric rendering approach of NeRF (Eq. 2.4). It is important to note that the application of the volumetric rendering function of Eq. 2.4 produces a single pixel color value for each individual ray. Moreover, if we assume an object is made of opaque surfaces, we can make our algorithm computationally more efficient by sampling only a few points located near the ray’s origin (points that are located further away from the origin does not contribute or influence the rendered color of the final pixel).

In summary, the generated rays are defined by their origin  $\tilde{\mathbf{r}}_o$ , direction  $\tilde{\mathbf{r}}_d$ , and rendered color  $\tilde{\mathbf{r}}_c$ . To simplify notation, we group and term these three components as  $\tilde{\mathbf{r}}$ . We can now leverage all the ray features to associate the rays with the image pixels.

### Matching by attenuating cast rays to image

From all the cast rays  $\tilde{\mathbf{r}}$ , we identify a subset of them that maximally correlates with the given image  $\mathbf{I}$  in order to recover the image pose ( $\hat{\mathbf{P}}$ ). To avoid a time-consuming brute force search, we learn to score rays that are related to the image pixels. We achieve this by embedding ray parameters and the image into a higher-dimensional feature space.

Specifically, to enable our approach to discriminate rays that share similar appearance and position, we utilize a multi-layer perceptron (MLP) denoted as  $\mathbf{R}_{fea} = \psi(\tilde{\mathbf{r}})$ ,

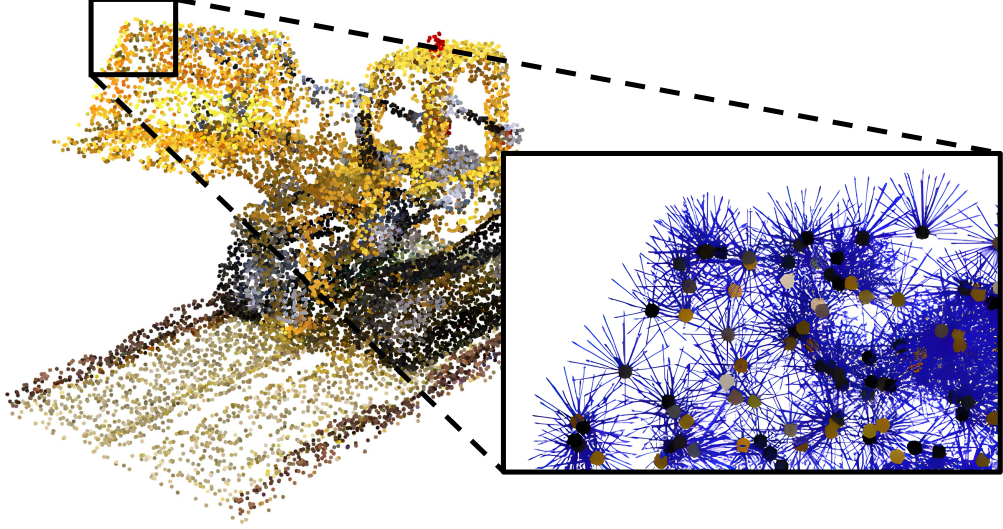


Figure 4.4: Example of generated rays using our approach. The zoomed region highlights the ray cast operation (in this case 27 rays per isocell).

where  $\mathbf{R}_{fea} \in \mathbb{R}^{N \times C}$ , and  $C$  denotes the number of channels. This MLP processes each ray independently, making the procedure insensitive to changes regardless how the rays are ordered. We project the MLP input into a higher-dimensional space using NeRF positional encoding [2]. This positional encoding expands subtle differences in the input, enabling the network to better distinguish similar rays [250].

To represent  $\mathbf{I}$ , we use DINOv2 [60] feature extractor, resulting in a set of features  $\mathbf{I}_{fea} \in \mathbb{R}^{W \times H \times C}$ , where  $W$  and  $H$  are the width and height of the image backbone’s output, respectively. Then, we define an attention module,  $\mathbf{M} \in \mathbb{R}^{W \times H \times C}$  to allow our model to correlate ray features,  $\mathbf{R}_{fea}$ , and image features,  $\mathbf{I}_{fea}$ , by using the image feature as query and the ray features as key:  $\mathbf{M} = A(\mathbf{R}_{fea}, \mathbf{I}_{fea})$ .

We optimize the attention map by summing along the rows and converting it into a correlation score,  $\hat{\mathbf{s}}$  per ray as:

$$\hat{\mathbf{s}} = \sum_{k=1}^{WH} \mathbf{M}_k. \quad (4.3)$$

The ground-truth score  $\mathbf{s}$  is defined based on the distance between the camera and its projection onto the considered ray. We can compute the projection of the point on the line with:  $t = \max((\mathbf{p} - \tilde{\mathbf{r}}_o) \cdot \tilde{\mathbf{r}}_d, 0)$ , where  $\mathbf{p}$  is the camera position,  $\tilde{\mathbf{r}}_o$  the generated ray and  $\tilde{\mathbf{r}}_d$  the corresponding direction. Rays are infinite only in one direction, so we restrict  $t \in \mathbb{R}^+$  using the max operator.

Then we can compute the distance between the camera origin and its projection on the ray as follows  $\mathbf{d} = \|(\tilde{\mathbf{r}}_o + t\tilde{\mathbf{r}}_d) - \mathbf{p}\|_2$ .  $\mathbf{d}$  is normalize in  $[0, 1]$  with:

$$\mathbf{d}_{\tilde{\mathbf{r}}} = 1 - \tanh\left(\frac{\mathbf{d}}{\lambda}\right), \quad (4.4)$$



where  $\lambda$  is a regularization parameter that adjusts the score values range. This affects how the rays are correlated and ranked based on the image features.

Lastly, we normalize all the scores  $\mathbf{s}$  as follows:

$$\mathbf{s} = \mathbf{d}_{\mathbf{r}} \frac{HW}{\sum_{j=0}^{HW} \mathbf{d}_{\mathbf{r},j}}. \quad (4.5)$$

During training our model optimizes  $\hat{\mathbf{s}}$  based on  $\mathbf{s}$  by using the  $L2$  norm loss function.

### Test-time pose estimation

At test-time the predicted scores  $\hat{\mathbf{s}}$ , are used as a correlation metric to select the best  $N_{top}$  rays, which should be the most relevant rays to the image, thus facilitating pose estimation. Fig. 4.5 shows an example of the selected rays and the score distribution. Note that only a small set of rays is sufficient to estimate the camera pose. However, we empirically set  $N_{top} = 100$  for robustness (Fig. 4.5a).

Because the camera pose estimation can be seen as the intersection point between the selected rays, we compute the ray intersection by searching for the solution of a weighted linear system of equations. Naturally, a set of 3D lines will not intersect at a single point, especially considering the noise introduced by discretizing the rays directions in the isocell. To overcome this, we compute the Least Squares solution that minimizes the sum of the squared perpendicular distances.

Formally, for the generated ray,  $\tilde{\mathbf{r}}_j$  with  $j = 1 \dots N_{top}$ , the error is given by the square of the distance from point  $\mathbf{p}$  to its projection on  $\tilde{\mathbf{r}}_j$ :

$$\sum_{j=0}^{N_{top}-1} ((\mathbf{p} - \tilde{\mathbf{r}}_{o,j})^T (\mathbf{p} - \tilde{\mathbf{r}}_{o,j}) - ((\mathbf{p} - \tilde{\mathbf{r}}_{o,j})^T \tilde{\mathbf{r}}_{d,j})^2). \quad (4.6)$$

An efficient way to compute the Least Squares solution to Eq. 4.6 is to compute its derivative respect to  $\mathbf{p}$ , resulting in

$$\mathbf{p} = \sum_{j=0}^{N_{top}-1} \hat{\mathbf{s}}_j (\mathbb{I} - \tilde{\mathbf{r}}_{d,j} \tilde{\mathbf{r}}_{d,j}^T) \tilde{\mathbf{r}}_{o,j}, \quad (4.7)$$

where  $\mathbb{I}$  is the identity matrix and  $\hat{\mathbf{s}}_i$  the predicted ray scores.

### 4.1.2 Results

We compare IFFNeRF to the baseline method iNeRF, with same backbone as ours. Using iNeRF’s protocol, we compare results on Synthetic NeRF [2] and, the real-world dataset, Tanks & Temples [3]. For each dataset, we use the predefined training-test splits and evaluate iNeRF with two initializations: *i)* the initialization proposed by iNeRF, where a given starting pose is between  $[-40^\circ, +40^\circ]$  degrees and of  $[-0.2, +0.2]$  units of rotational and translation error from the ground-truth target pose, respectively;



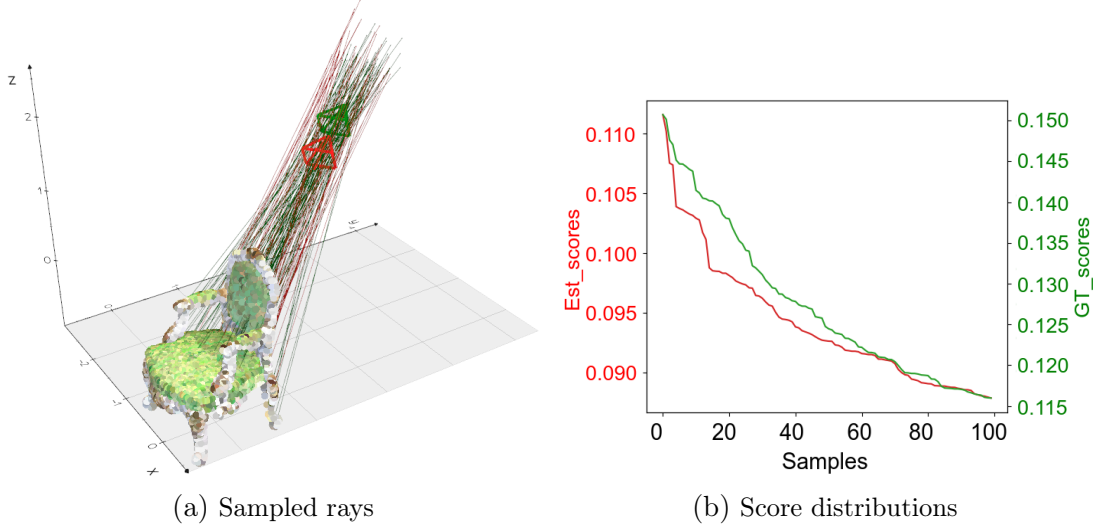


Figure 4.5: Example of the test-time pose estimation on the Chair object of Synthetic NeRF. (a) Top  $N$  (red) *vs.* top  $N$  ground-truth (green) rays. (b) Corresponding score distribution of top  $N$  and ground-truth rays.

*ii)* the initialization obtained by randomly selecting a pose from the ones available in the training set. Additionally, we conduct an ablation study on IFFNeRF’s backbone configuration (pretrained *vs.* fine-tuned) and the use of IFFNeRF as an initialization for iNeRF. Evaluation metrics include mean angular (MAE) and translation (MTE) errors, and we measure the inference time in Tab. 4.1.

*Implementation Details:* We implement IFFNeRF with PyTorch and train for 1.5K iterations ( $\sim 45$ mins) on the training split on a standard workstation with a NVIDIA GeForce RTX 3090. We use the Adam optimizer with learning rate  $10^{-3}$ . Hyper-parameters are set as  $\lambda = 1$ ,  $G = 5000$  and 800 M-H iterations.

## Datasets

We use two datasets: *i)* *Synthetic NeRF* [2]: This dataset was released together with NeRF. Synthetic NeRF comprises eight scenes (Chair, Drums, Ficus, Lego, Materials, Ship, Mic, Hot Dog), includes both camera intrinsic and extrinsic parameters, provides train and test splits of 100 images (33%) and 200 images (66%), respectively. *ii)* *Tanks & Temples* [3]: This dataset was originally created to evaluate 3D reconstruction methods and embeds challenging scenarios as real-world objects with variations in size (small and large), acquired from a human-like viewpoints, at various distances from the object, and with different illumination conditions, such as shadows. As in [77], we evaluate on five scenes: Barn, Caterpillar, Family, Ignatius and Truck. We use the train and test splits as specified in [251], with the split depending on the object, having on average  $\approx 247$  training images (87%) and  $\approx 35$  testing images (12%). Like [169], we resize all the objects for both datasets to fit inside a unit box. Therefore, the translation error is relative to the object size and we define it in units.

#### 4.1. 6DoF using Neural Radiance Fields

**Table 4.1** Evaluation of 6DoF pose estimation on two datasets: (a) Synthetic NeRF [2] and (b) Tanks & Temples [3]. We report the results in terms of Mean Angular Error (MAE) and Mean Translation Error (MTE). The lower MAE and MTE, the better. Best-performing results are highlighted in **bold** and second-best results are underlined.

| (a) Synthetic NeRF                  |             |              |             |              |             |              |             |              |             |              |             |              |             |              |             |              |             |              |              |
|-------------------------------------|-------------|--------------|-------------|--------------|-------------|--------------|-------------|--------------|-------------|--------------|-------------|--------------|-------------|--------------|-------------|--------------|-------------|--------------|--------------|
| Method                              | Drums       |              | Chair       |              | Lego        |              | Ship        |              | Ficus       |              | Mic         |              | Hot Dog     |              | Materials   |              | Mean        |              |              |
|                                     | MAE         | MTE          | MAE         | MTE          | MAE         | MTE          | MAE         | MTE          | MAE         | MTE          | MAE         | MTE          | MAE         | MTE          | MAE         | MTE          | MAE         | MTE          | Time (s)     |
| iNeRF (initialization y [214])      | 17.4        | 0.430        | 15.4        | 0.441        | 12.7        | 0.293        | 9.8         | 0.344        | 18.1        | 0.485        | 17.6        | 0.501        | 3.8         | 0.061        | 15.1        | 0.375        | 13.7        | 0.366        | 6.1          |
| iNeRF (Random initialization)       | 74.2        | 1.493        | 100.5       | 2.571        | 93.0        | 1.868        | 89.1        | 1.535        | 93.9        | 1.985        | 99.3        | 2.520        | 86.5        | 1.707        | 84.3        | 1.734        | 90.1        | 1.927        | 6.1          |
| Ours (w. DINOv2 pretrained)         | <b>22.5</b> | 1.210        | <u>13.2</u> | 1.106        | <b>11.8</b> | 1.009        | 50.5        | 1.318        | <u>19.4</u> | 1.417        | <b>10.7</b> | 1.046        | <b>10.5</b> | 1.164        | 33.1        | 1.252        | <u>21.5</u> | 1.190        | <b>0.029</b> |
| Ours (w. DINOv2 fine-tuned)         | 28.3        | <b>0.916</b> | <b>7.1</b>  | <u>0.429</u> | <u>17.6</u> | <u>0.547</u> | <u>25.7</u> | <u>0.600</u> | <b>18.8</b> | <u>0.818</u> | 14.8        | <u>0.576</u> | 15.6        | <u>0.650</u> | <b>15.4</b> | <u>0.493</u> | <b>17.9</b> | <u>0.629</u> | <b>0.029</b> |
| Ours (w. DINOv2 fine-tuned) w/iNeRF | <u>26.2</u> | <u>0.921</u> | 25.7        | <b>0.351</b> | 18.4        | <b>0.462</b> | <b>23.1</b> | <b>0.524</b> | 27.8        | <b>0.639</b> | <u>13.5</u> | <b>0.521</b> | <u>15.4</u> | <b>0.537</b> | <u>23.4</u> | <b>0.455</b> | 21.7        | <b>0.551</b> | 6.2          |

| (b) Tanks And Temples (real)     |             |              |             |              |             |              |             |              |             |              |             |              |              |          |
|----------------------------------|-------------|--------------|-------------|--------------|-------------|--------------|-------------|--------------|-------------|--------------|-------------|--------------|--------------|----------|
| Method                           | Barn        |              | Caterpillar |              | Family      |              | Ignatius    |              | Truck       |              | Mean        |              |              |          |
|                                  | MAE         | MTE          | MAE         | MTE          | MAE         | MTE          | MAE         | MTE          | MAE         | MTE          | MAE         | MTE          | MAE          | Time (s) |
| iNeRF (initialization by [214])  | 26.5        | 0.208        | 42.9        | 0.166        | 42.8        | 0.794        | 31.4        | 0.723        | 31.6        | 0.370        | 35.0        | 0.452        | 6.1          |          |
| iNeRF (Random initialization)    | 89.2        | 0.682        | 89.3        | 2.559        | 93.9        | 1.505        | 84.1        | 1.489        | 94.4        | 1.042        | 90.2        | 1.455        | 6.1          |          |
| Ours (w. DINOv2 pretrained)      | <b>22.3</b> | 0.121        | 33.2        | 0.472        | 24.3        | 0.638        | <b>15.6</b> | 0.495        | 22.3        | 0.140        | <u>23.5</u> | 0.373        | <b>0.029</b> |          |
| Ours (w. DINOv2 fine-tuned)      | <u>26.9</u> | <u>0.060</u> | <u>30.4</u> | <u>0.427</u> | <u>22.1</u> | <u>0.585</u> | <u>18.7</u> | <u>0.455</u> | <b>20.7</b> | <u>0.104</u> | 23.7        | <u>0.326</u> | <b>0.029</b> |          |
| Ours (DINOv2 fine-tuned) w/iNeRF | <u>26.9</u> | <b>0.037</b> | <b>28.3</b> | <b>0.395</b> | <b>18.1</b> | <b>0.561</b> | 19.5        | <b>0.297</b> | <u>21.9</u> | <b>0.096</b> | <b>22.9</b> | <b>0.277</b> | 6.2          |          |

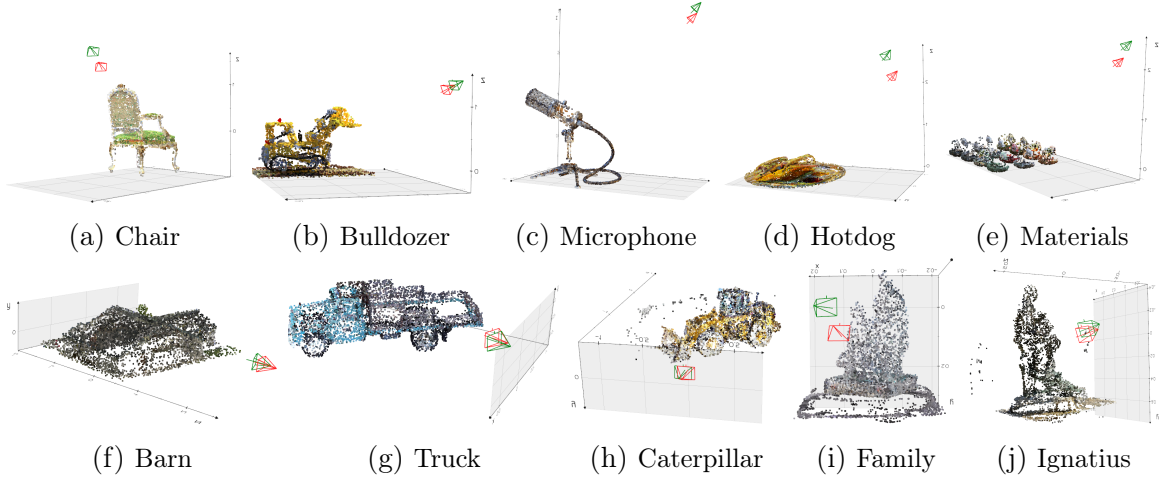


Figure 4.6: Examples of camera pose estimation results of our approach on 10 objects from both the synthetic NeRF (1st row) and the Tanks&Temples (2nd row) datasets. Green indicates the ground-truth camera and red indicates the estimated camera.

#### Discussion on results

We report the quantitative results in Tab. 4.1 and the corresponding qualitative results in Fig. 4.6 for both evaluated datasets.

**Effect of initialization on iNeRF:** In both Synthetic NeRF (Tab. 4.1a) and Tanks and Temples (Tab. 4.1b), iNeRF performs best when initialized from poses near the known camera (following [214] evaluation). However, in more realistic settings with random initialization, the relative error is  $21.2^\circ$  degrees and 0.218 units worse in terms of angular and translation errors, respectively.

**Synthetic & real datasets:** Tab. 4.1a shows that our IFFNeRF consistently outperforms iNeRF with random initialization reducing the average angular error by  $72.2^\circ$  degrees and the translation by 1.29 units, respectively, in the case of the fine-tuned backbone. The same significant performance improvement can be observed in Tanks & Temples (real). Tab. 4.1b shows a  $66.5^\circ$  and 1.129 improvement for rotation and translation errors, respectively. These results suggest that IFFNeRF can be resilient to images taken from a wide range of camera poses. Conversely, iNeRF with the initialization bounds specified in [214], on synthetic data, consistently achieves better results. However, this behavior of iNeRF is not confirmed with real data, whereas IFFNeRF clearly stands out with better performance while not requiring constrained initialization. Furthermore, applying iNeRF as a refinement step for IFFNeRF’s solution leads to improved results, especially on real data.

**Comparison of backbones and noise robustness:** The effect of pretrained *vs.* fine-tuned backbones depends on the dataset. In Synthetic NeRF (Tab. 4.1a), we can observe a significant improvement with the fine-tuned backbone for both rotation and translation errors. However, in Tanks & Temples (Tab. 4.1b), we can observe that only the translation error slightly decreases by 0.047, while the rotation error increases by 0.2. Such small variation may be caused by the pretrained DINOv2 as it can already extract highly informative features from real-world data. Our architecture seems also to be robust to noise, without re-training in the NeRF-synthetic dataset, and despite introducing a 50% Gaussian noise to the image, the errors increased only by  $14.01^\circ$  and  $0.201u$ . Synthetic data might represent a domain shift because it is unlikely that they were used during training.

**IFFNeRF as initialization to iNeRF:** As the output pose of IFFNeRF provides a near estimate of the solution, we can also use this as an initialization for iNeRF optimization process. It can be seen that in Synthetic NeRF, in contrast to our best-performing result, only the mean translation error decreases by 0.078, whereas the angular error increases by 3.8. In Tanks and Temples, both translation and angular errors decrease by 0.049 and 0.6 respectively. However, it is interesting to observe that, in contrast to iNeRF with its original initialization [214], the combination of IFFNeRF and iNeRF performs relatively poorly on synthetic (by 15.2 and 0.399 higher angular and translation error, respectively), but significantly better on real-world data. This could be due to the fact that real data present shadows in the scenes. The camera and its handling system is not transparent, as in the synthetic data. These generated shadows seem to affect iNeRF, especially in the case of lateral translation.

**Computational performance:** In terms of inference time, IFFNeRF is significantly more efficient than iNeRF, achieving real-time performance at 34fps. In the case of IFFNeRF with iNeRF optimization, we revert to the running time of iNeRF; however, as noted in the case of real-world datasets (*i.e.* Tanks & Temples) this step further

improves accuracy. In addition, it is worth mentioning that our method is also faster than the CROSSFIRE [171] and Parallel-iNeRF [169], where on the same hardware they report 200ms and 20s, respectively, compared to 29ms of our solution. A Limitation of IFFNeRF is to require training for each scene. Finally, in terms of memory, at test time IFFNeRF only uses 3294 MB of memory as opposed to the 4870 MB of iNeRF.

## 4.2 6DoF using 3D Gaussian Splatting

During the development of this dissertation, the field of 3D representation has witnessed a remarkable transformation, with the emergence of geometrical representations, such as 3DGS [78, 252, 253]. These novel representations have experienced rapid adoption beyond academic research, with integration into everyday applications such as mapping services [254]. 3DGS offers an alternative to neural NVS models, providing faster rendering through explicit geometric primitives without the computational overhead of optimizing a neural network.

As described earlier, 3DGS represents a 3D scene as a collection of ellipsoids, each carrying photometric information (color and opacity). These ellipsoids are initialized from Structure from Motion (SfM) point clouds and subsequently optimized to minimize the photometric error between rasterized ellipsoids and known reference images. During the rasterization process, the 3DGS model projects these ellipsoids onto the image plane as 2D ellipses, computing photometric contributions for each affected pixel. These advances in 3D representation create new challenges for solving classical problems like 6D pose estimation, while simultaneously offering opportunities to leverage the inherent advantages of NVS approaches [169, 171, 172, 214, 255].

As discussed earlier, iNeRF [214] pioneered 6D pose estimation using an NVS model through an iterative analysis-by-synthesis approach, illustrated in the left panel of Fig. 4.7. Building on the unique properties of 3DGS, we present 6DGS, a novel 6DoF pose estimation method that overcomes the fundamental limitations of NeRF-based iterative approaches. Similar to IFFNeRF, 6DGS does not require any initial pose estimation and computes camera parameters without relying on computationally expensive iterative analysis-by-synthesis approaches, enabling near real-time performance (15fps) thanks to the efficient rendering capabilities of 3DGS.

The right panel of Fig. 4.7 illustrates our approach. In standard 3DGS rendering, knowing the camera pose allows projecting ellipsoid centers onto the image plane through rays cast through the camera’s optical center. 6DGS effectively inverts this process to estimate the camera pose. Since the target image camera pose is unknown, we cannot cast single rays from each ellipsoid through the correct target image pixels. Instead, we uniformly radiate rays from each ellipsoid using our novel Ellicell casting procedure. Of all these radiated rays, only one per ellipsoid would accurately render the pixel photometrically by projecting the correct ellipse onto the target image plane.

Therefore, the core challenge in 6DGS becomes selecting the correct bundle of rays that can generate most of the target image pixels with high confidence. Taking inspiration from IFFNeRF, we address this selection challenge through an attention map mechanism that binds pixels and rays. This critical step leverages the known camera poses and images used to compute the 3DGS model, providing pixel-ray pairs for training the attention mechanism. After selecting the optimal ray bundle, their intersection identifies the camera center, which we solve using weighted Least Squares with weights derived from the selection stage scores. Once the optical center is estimated, we can determine the camera rotation from the ray bundle, thus solving the full 6DoF pose.

By design, 6DGS eliminates the need for an initial camera pose—a key limitation of

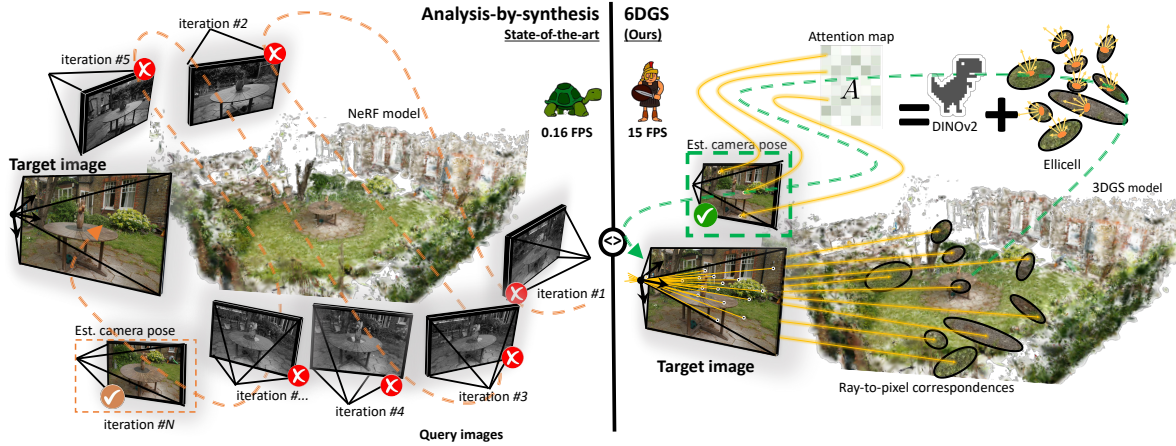


Figure 4.7: Our 6DGS method introduces a novel approach to 6DoF pose estimation, departing from conventional analysis-by-synthesis methodologies. Standard NeRF-based methods (**left**) employ an iterative process, rendering candidate poses and comparing them with the target image before updating the pose, which often results in slow performance and limited precision. In contrast, 6DGS (**right**) estimates the camera pose by selecting a bundle of rays projected from the ellipsoid surface (a *radiant Ellicell*) and learning an attention map to output ray/image pixel correspondences (based on DINOv2). The optimal bundle of rays should intersect the optical center of the camera and then are used to estimate the camera rotation in closed-form. Our 6DGS method offers significantly improved accuracy and speed, enabling the recovery of the pose within a one-shot estimate.

analysis-by-synthesis methods [171, 172, 214]—and avoids the problem of convergence to local minima during iteration, especially when initialization is far from optimal.

We comprehensively evaluate 6DGS on datasets featuring challenging real-world objects and scenes, comparing against state-of-the-art NVS approaches including iNeRF [214], Parallel iNeRF [169], and NeMO + VoGE [172]. Our results demonstrate that 6DGS is particularly effective in scenarios where initial pose is not provided, and achieves near real-time 6DoF pose estimation on consumer hardware—addressing a significant limitation in practical applications of NVS-based camera pose estimation.

To summarize, 6DGS offers three key contributions:

- Our approach eliminates the need for initial camera pose estimates and iterative convergence typically required in analysis-by-synthesis methods;
- 6DGS introduces a novel ray casting pipeline (Ellicell) and an attention-based mechanism that efficiently matches pixel-level image information with 3DGS ellipsoids;
- The proposed method achieves state-of-the-art performance in NVS benchmarks for camera pose estimation in terms of both accuracy and real-time performance.

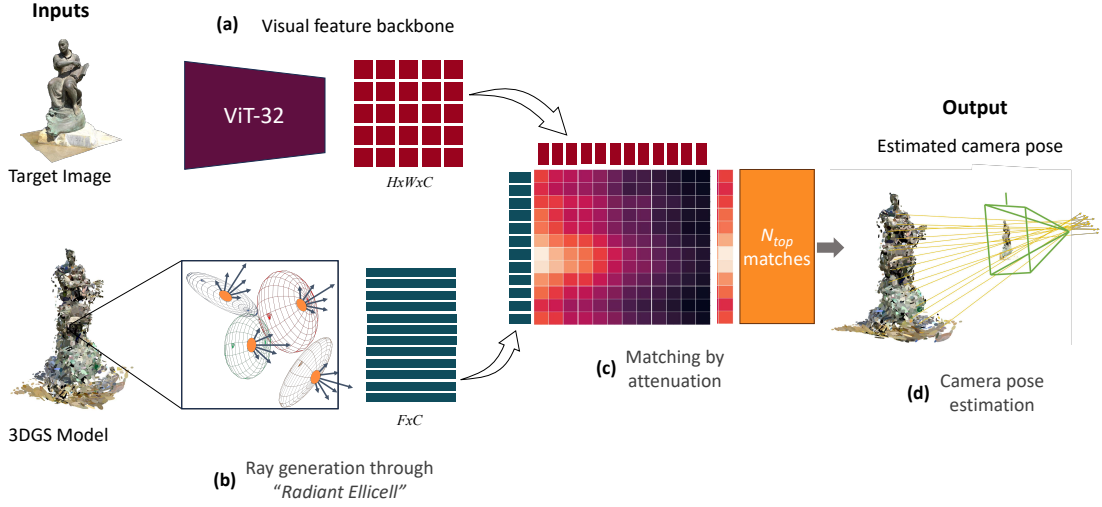


Figure 4.8: The figure illustrates the pipeline of our 6DGS methodology. The image is encoded using a visual backbone (a). Concurrently, rays are uniformly projected from the center of the 3DGS ellipsoids (b), and their corresponding color is estimated. Subsequently, an attention map mechanism is employed to compare the encoded ray and image features (c). Following this comparison, the  $N_{top}$  matches are selected via attenuation, and the camera location is estimated (d) as the solution of a weighted Least Squares problem, resulting in a distinct 6DoF pose for the image.

### 4.2.1 Approach

6DGS estimates the camera pose  $\hat{\mathbf{P}} \in \text{SE}(3)$ , given a target image  $\mathbf{I}_t$  and a set of ellipsoids  $\mathcal{Q}$  from a pre-computed 3DGS model (Fig. 4.8). This section follows the 3DGS notation introduced in Section 2.1.3. For a comprehensive overview of the 3DGS method, we refer the reader to that section. To solve for the camera pose, we propose a casting method from the ellipsoid’s surface, called Ellicell, that divides it in equal area cells (Sec. 4.2.1). The ellipsoids cast a set of  $N$  rays, denoted as  $\mathcal{V} = \{\langle \mathbf{v}_o, \mathbf{v}_d, \mathbf{v}_c \rangle\}_{i=1}^N$ , one for each of the generated cell (Fig. 4.9c). Each ray is identified by *i*) the origin  $\mathbf{v}_o \in \mathbb{R}^3$ , *ii*) the center point of each ellipsoid, *iii*) the direction  $\mathbf{v}_d \in \mathbb{R}^3$  originating from the ellipsoid center to the cell center and through the space, and *iv*) the color information  $\mathbf{v}_c \in \mathbb{R}^3$  as RGB values. We synthesize the rays’ color using the 3DGS rendering function  $\phi$  (Eq. 2.11). A subset of these rays, depending on the view perspective, may intersect the camera’s optical center. For binding the rays to the image pixels we compute the target image pixels features  $\psi(\mathbf{I}_t)$  (Fig. 4.8a) and the rays features  $\psi(\mathcal{V})$  (Fig. 4.8b). These features are used to identify the intersecting rays by using an attention map  $\mathcal{A}$  (Fig. 4.8c), see Sec. 4.2.1. The higher the attention value for a ray-pixel pair is, the more likely the intersection on the image plane is a valid one. Finally, we determine the target camera pose  $\hat{\mathbf{P}}_t$  (Fig. 4.8d) by computing the intersection point of the selected rays using a weighted Least Squares approach (Sec. 4.2.1).



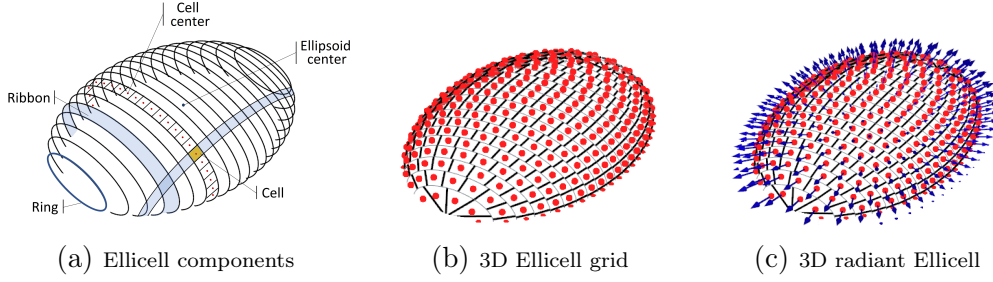


Figure 4.9: The illustration depicts the three primary stages involved in the radiant Ellicell generation. Firstly, **(a)** delineates the formulation of components required to compute the geometric information for each cell. Secondly, **(b)** shows the resulting Ellicell grid positioned on the surface of the ellipsoid along with their respective center points. Finally, **(c)** demonstrates the generation of rays originating from the center point of the ellipsoid going through the Ellicell center.

### Radiant Ellicell

We create rays spanning in every direction allowing 6DGS to recover the camera pose. We introduce the concept of radiant Ellicell for generating rays that uniformly emanate from the ellipsoid surface, as illustrated in Fig. 4.9. Ellicell generation is deterministic [245, 246] and achieves higher precision with fewer rays [248, 249] compared to other sampling methods like Monte-Carlo [247].

First, we compute the area of each Ellicell. This is achieved by calculating the total ellipsoid surface area, using a computationally efficient approach, namely Ramanujan approximation [256]:

$$h = 4\pi \left( \frac{(ab)^{1.6} + (ac)^{1.6} + (bc)^{1.6}}{3} \right)^{\frac{1}{1.6}}, \quad (4.8)$$

where  $a, b, c = \text{diag}(\mathcal{S})$  are the ellipsoid axis scales. Each Ellicell cell's target area equals  $\mu = h/G$ , with  $G$  being the number of cells dividing each ellipsoid. Approximating each cell as a square with side  $z = \sqrt{\mu}$  we slice the ellipsoids along the major axis into ribbons, each as wide as  $z$  (Fig. 4.9a). The extremity of each ribbon is called a ring. The total number of rings is  $e = \lfloor \kappa(a, b)/(2z) \rfloor \in \mathbb{N}$ , where  $\kappa(a, b)$  computes the ring perimeter. Ignoring ellipsoid's rotation, we compute the ring perimeter by treating them as 2D ellipses, thus defining  $\kappa(a, b)$  as:

$$\kappa(a, b) = \pi \left( (a + b) + \frac{3(a - b)^2}{10(a + b) + \sqrt{a^2 + 14ab + b^2}} \right). \quad (4.9)$$

Given the total number of rings  $e$  it is possible to compute the ribbon's centerline geometric parameters. We calculate the scale parameter of each ribbon centerline as:

$$\varrho(n, \Delta r, a, b) = \sqrt{1 - \frac{(0.5\Delta r + n\Delta r - a)^2}{b^2}}, \quad (4.10)$$



where  $\Delta r = a/e$  is the distance between two consecutive rings. This equation derives from the manipulation of the standard ellipse equation. While ribbon size  $z$  should be equal to  $\Delta r$ , these two values will likely differ due to the need for the number of rings being a natural number. Eq. 4.10 is also used to compute the other ribbon scaling parameter by replacing  $b$  with  $c$ . The scaling parameter  $\varrho$  is subsequently used to compute the number of cells inside each ribbon as:

$$\xi(n, e, a, b, c) = \left\lfloor \frac{\kappa(\varrho(n, e, a, b), \varrho(n, e, a, c))}{z} \right\rfloor, \quad (4.11)$$

where  $\xi$  is the number of cells inside the ring. We compute the center of each cell, equally spaced along the ribbon's centerline, by sampling  $\xi$  points along it. This is challenging as the perimeter distance does not linearly correlate with the  $x$  and  $y$  variations. However, we can solve this by using a statistical method. Knowing a distribution's Cumulative Distribution Function (CDF) allows us to sample uniformly between 0 and 1 and then use the CDF inverse to map the sample to the distribution space. This approach applies to our case, where samples are distributed as follows:

$$ds^2 = dx^2 + dy^2, \quad (4.12)$$

and, by taking its inverse, we can retrieve the coordinates of each cell center. To simplify the equations, we define  $r = \varrho(n, e, a, b)$  and  $w = \varrho(n, e, a, c)$  to indicate the scale of the ellipse under consideration. Then we express Eq. 4.12 in polar coordinates to simplify the differentiation:

$$\frac{ds}{d\theta} = \sqrt{r^2 \sin^2 \theta + w^2 \cos^2 \theta}, \quad (4.13)$$

then, we can express the set of points on the perimeter of the ribbon centerline as an angular position in the polar coordinate system as:

$$\theta' = \left( \frac{ds}{d\theta} \right)^{-1} \left( g \cdot \frac{1}{\xi(n, e, a, b, c)} \right), \quad (4.14)$$

with  $g$  being the cell identifier. Given  $\theta'$  we can use it inside the ellipse equation in polar coordinates to obtain the 3D position of each cell center:

$$\mathbf{u} = \begin{pmatrix} w \cos(\theta') \\ g \sin(\theta') \\ -a + n\Delta r \end{pmatrix}. \quad (4.15)$$

## Ray generation

Once we have divided each ellipsoid of the 3DGS model into equidistant cells, we cast the rays originating from the center point of the ellipsoid *i.e.*  $\mathbf{v}_o = \mathbf{x}$  and oriented towards the Ellicell center  $\mathbf{v}_d = \mathbf{u} - \mathbf{x}$ . We reduce the number of potential rays cast from each ellipsoid by considering only the rays oriented in the same hemisphere as the estimated surface normal of the ellipsoid. We obtain the surface normals by treating

the ellipsoid centroids as a point cloud, and the surface normal is estimated using the local neighborhood geometry of nearby points [257].

Finally, each ray has also been associated with the color information  $\mathbf{v}_c$ , which we compute through the same pixel-level approach of 3DGS (Eq. 2.11). We note that the application of the volumetric rendering function of Eq. 2.11 produces a single pixel for each ray. The generated rays represent a collection of potential hypotheses, meaning that a subset of them will intersect the target image  $\mathbf{I}_t$ .

### Binding by attenuation of rays to image

Given all the cast rays  $\mathbf{v}$ , we identify a subset of  $\mathbf{v}$  correlating with the target image  $\mathbf{I}_t$ . A learned attention map  $\mathcal{A}$  assigns scores  $\hat{\mathbf{s}}$  based on the highest correlation to image pixels; higher similarity results in higher scores. Based on the computed scores  $\hat{\mathbf{s}}$ , we select the top candidate rays ( $N_{top}$ ) that exhibit the highest association scores and subsequently use them to recover the camera pose ( $\hat{\mathbf{P}}_t$ ).

To select rays with similar appearance and position, we use a Multi-Layer Perceptron (MLP) defined as  $\mathbf{V} = \psi(\mathbf{v})$ , where  $\mathbf{V} \in \mathbb{R}^{N \times C}$  with  $C$  being the feature size and  $N$  the overall number of rays. The MLP input is enriched by incorporating NeRF-style Positional Encoding that maps the data into the Fourier domain [250] to better distinguish between similar data points.

We generate features from  $\mathbf{I}_t$  using DINOv2 [60] as a pre-trained backbone feature extractor. This results in a set of features  $\mathbf{F}_t \in \mathbb{R}^{M \times C}$ , where  $M = W \times H$ . Both the image and ray features sets are processed by a single attention module  $\mathcal{A}(\mathbf{V}_f, \mathbf{F}_t) \in \mathbb{R}^{M \times N}$  producing a set of scores. Inside the attention module the ray features,  $\mathbf{V}$ , are used as queries and the image features,  $\mathbf{F}_t$ , as a key. We optimize the attention map by summing along the rows and converting it into a per-ray correlation score as follows  $\hat{\mathbf{s}} = \sum_{i=1}^M \mathcal{A}_i$ . The higher the score value given by  $\hat{\mathbf{s}}$ , the better the association between the rays and image pixels. At test-time we select the  $N_{top}$  rays with the highest ranking scores to proceed with pose estimation.

Because a ray and an image pixel should be associated with each other based on the distance between the camera origin and its projection onto the corresponding ray, we supervise the predicted scores  $\hat{\mathbf{s}}$  using the same images and camera poses used to estimate the 3DGS model during the training phase. We compute the projection of the point on the line as  $l = \max((\mathbf{O} - \mathbf{v}_o) \cdot \mathbf{v}_d, 0)$ , where  $\mathbf{O}$  is the camera position,  $\mathbf{v}_o$  the generated ray origin and  $\mathbf{v}_d$  the corresponding direction. Rays are infinite only in one direction, so we restrict  $l \in \mathbb{R}^+$  using the max operator. Then, we can compute the distance between the camera origin and its projection on the ray as  $\mathbf{h} = \|(\mathbf{v}_o + l\mathbf{v}_d) - \mathbf{O}\|_2$ . The value  $\mathbf{h}$  can span from 0 to  $+\infty$ , with 0 indicating a ray that passes through the camera’s optical center. We map distances to the attention map score using:

$$\delta = 1 - \tanh\left(\frac{\mathbf{h}}{\lambda}\right), \quad \mathbf{s} = \delta \frac{M}{\sum \delta}, \quad (4.16)$$

where  $\lambda$  regulates the number of rays to assign to a specific camera. Lastly, the softmax inside the attention map computation requires we normalize the ground truth scores.

We use the  $L2$  loss to minimize the difference between the predicted  $\hat{\mathbf{s}}$  and the computed ground truth  $\mathbf{s}$  scores as:

$$\mathcal{L} = \frac{1}{MN} \sum_{i=1}^M \sum_{j=1}^N \|\hat{\mathbf{s}}_{i,j} - \mathbf{s}_{i,j}\|_2, \quad (4.17)$$

where  $M, N$  are the size of the attention map  $\mathcal{A}$ . During each training iteration, we predict an image and a pose utilized for estimating the 3DGS model.

### Test-time pose estimation

During the test phase, the predicted scores  $\hat{\mathbf{s}}$  are used to select the top  $N_{top}$  rays, identified as the utmost relevant, and constrained to choose at most one ray per ellipsoid. Note that only a small set of rays is sufficient to estimate the camera pose. However, based on an ablation study we set  $N_{top} = 100$ , see Tab. 4.4.

The camera position is found at the intersection of selected rays, solved as a weighted Least Squares problem. Since 3D lines usually do not intersect at a single point due to discretization noise introduced by the Ellicell, we minimize the sum of squared perpendicular distances instead.

For the selected ray  $\mathbf{v}_j$  with  $f = 1 \dots N_{top}$ , the error is given by the square of the distance from the camera position to predict  $\hat{\mathbf{O}}$  to its projection on  $\mathbf{v}_j$ :

$$\sum_{f=1}^{N_{top}} \left( (\hat{\mathbf{O}} - \mathbf{v}_{o,f})^T (\hat{\mathbf{O}} - \mathbf{v}_{o,f}) - ((\hat{\mathbf{O}} - \mathbf{v}_{o,f})^T \mathbf{v}_{d,f})^2 \right), \quad (4.18)$$

where  $\mathbf{v}_{o,f}$  indicating the origin of the  $f$ -th ray and  $\mathbf{v}_{d,f}$  the respective direction. To minimize Eq. 4.18, we differentiate it with respect to  $\hat{\mathbf{O}}$ , resulting in

$$\hat{\mathbf{O}} = \sum_{f=1}^{N_{top}} \hat{\mathbf{s}}_f (\mathbb{I} - \mathbf{v}_{d,f} \mathbf{v}_{d,f}^T) \mathbf{v}_{o,f}, \quad (4.19)$$

where  $\mathbb{I}$  is the identity matrix and  $\hat{\mathbf{s}}_f$  are the predicted ray scores. This expression can be solved as a weighted system of linear equations.

## 4.2.2 Results

### Experimental setup

We evaluate 6DGS and compare with other analysis-by-synthesis methods for 6D pose estimation, including iNeRF [214], Parallel iNeRF [169], and NeMo+VoGE [172, 173]. We reproduce the results using their published code. We follow iNeRF’s evaluation protocol and test on two real-world datasets: Tanks & Temples [3] and Mip-NeRF 360° [5]. For each dataset, we use the predefined training-test splits and evaluate them with two pose initialization pipelines: *i)* the original iNeRF initialization, where the starting

## 4.2. 6DoF using 3D Gaussian Splatting

**Table 4.2** Evaluation of 6DoF pose estimation on the Mip-NeRF 360° [4] dataset. We report results in terms of Mean Angular Error (MAE) and Mean Translation Error (MTE) in terms of degrees and units,  $u$ , respectively. Where  $1u$  is equal to the object’s largest dimension. For both metrics lower is better. Best-performing results are highlighted in **bold** and green, while second best values are underline.

| Scenes  | Fixed pose prior (eval. protocol by [214]) |                   |                          |             |                   |                      | Random pose prior |                   |                      | No pose prior |  |
|---------|--------------------------------------------|-------------------|--------------------------|-------------|-------------------|----------------------|-------------------|-------------------|----------------------|---------------|--|
|         | iNeRF [214]                                | NeMo + VoGE [172] | Parallel iNeRF [169]     | iNeRF [214] | NeMo + VoGE [172] | Parallel iNeRF [169] | iNeRF [214]       | NeMo + VoGE [172] | Parallel iNeRF [169] | 6DGS (Ours)   |  |
|         | MAE ↓ MTE ↓                                | MAE ↓ MTE ↓       | MAE ↓ MTE ↓              | MAE ↓ MTE ↓ | MAE ↓ MTE ↓       | MAE ↓ MTE ↓          | MAE ↓ MTE ↓       | MAE ↓ MTE ↓       | MAE ↓ MTE ↓          | MAE ↓ MTE ↓   |  |
| Bicycle | 39.5 0.116                                 | 43.8 <u>0.015</u> | <u>35.9</u> 0.116        | 76.6 0.217  | 111.8 0.038       | 44.4 0.150           | 12.1 <b>0.010</b> |                   |                      |               |  |
| Bonsai  | 51.3 0.228                                 | 52.5 <b>0.036</b> | <u>41.1</u> 0.223        | 96.7 0.385  | 98.9 0.073        | 58.2 0.298           | 10.5 <u>0.038</u> |                   |                      |               |  |
| Counter | 40.7 0.324                                 | 45.6 <u>0.072</u> | <u>24.7</u> 0.212        | 70.3 0.487  | 98.1 0.139        | 42.1 0.435           | 19.6 <b>0.043</b> |                   |                      |               |  |
| Garden  | <u>31.0</u> 0.121                          | 31.8 <u>0.026</u> | <b>18.2</b> 0.090        | 72.8 0.210  | 89.2 0.038        | 60.0 0.144           | 37.8 <b>0.015</b> |                   |                      |               |  |
| Kitchen | 38.2 0.113                                 | 41.6 <u>0.042</u> | <u>37.3</u> 0.109        | 100.2 0.266 | 122.2 0.082       | 65.0 0.193           | 23.2 <b>0.018</b> |                   |                      |               |  |
| Room    | 38.8 0.274                                 | 44.9 <u>0.045</u> | <b>30.7</b> 0.257        | 91.6 0.444  | 110.0 0.010       | 63.5 0.271           | 38.3 <b>0.019</b> |                   |                      |               |  |
| Stump   | <u>21.4</u> 0.030                          | 26.3 <u>0.016</u> | <b>14.8</b> <u>0.016</u> | 86.9 0.035  | 96.3 0.025        | 72.6 0.033           | 28.3 <b>0.009</b> |                   |                      |               |  |
| Avg.    | 37.3 0.172                                 | 40.9 <u>0.036</u> | <u>28.9</u> 0.146        | 85.0 0.292  | 103.8 0.058       | 58.0 0.218           | 24.3 <b>0.022</b> |                   |                      |               |  |

pose is sampled uniformly between  $[-40^\circ, +40^\circ]$  degrees of errors and  $[-0.1, +0.1]$  units of translation error from the ground-truth target pose; *ii*) by randomly choosing an initialization pose from the ones used to create the 3DGS mode. Although analysis-by-synthesis methods were tested with a prior, in reality it is rarely available, so we present a second scenario to assess them under more realistic conditions. We perform multiple ablation studies to assess the sensitivity of 6DGS to different hyperparameters and settings. We quantify pose estimation results in terms of mean angular (MAE) and translation (MTE) errors (see Tab. 4.2 and Tab. 4.3) and measure the inference time.

**Implementation Details.** 6DGS is implemented in PyTorch and the attention map was trained for 1.5K iterations ( $\sim 45$ mins) with an NVIDIA GeForce RTX 3090. We use the Adafactor optimizer [258] with weight decay of  $10^{-3}$ . For speedup training, we uniformly sample 2000 3DGS ellipsoids at each iteration.

## Datasets

To demonstrate the applicability of 6DGS, we test on two datasets featuring real world challenges. **Tanks&Temples** [3] was created to evaluate 3D reconstruction methods with challenging real-world objects of varying sizes, acquired from human-like view-points and with difficult conditions (illumination, shadows, and reflections). We use the five scenes (Barn, Caterpillar, Family, Ignatius, Truck) and the train test splits given in [77, 251]. The splits are object dependent, having on average  $\approx 247$  training images (87%) and  $\approx 35$  testing images (12%). **Mip-NeRF 360°** [5] consists of seven scenes: two outdoors and four indoors, with a structured scenario and background. We use the original train-test splits [5], at a ratio of 1:8. Following [169], we resize all the objects to fit inside a unit box for standardization. The translation error is relative to the object size, defined as a single unit of measurement.

## Analysis

**Quantitative Analysis:** Tab. 4.2 and Tab. 4.3 present the results obtained across both datasets. 6DGS consistently outperforms baseline methods across all datasets and pose initialization pipelines. Notably, 6DGS achieves lower error rates than the second-best

**Table 4.3** Evaluation of 6DoF pose estimation on the Tanks&Temples [3] dataset. We show the same metrics and analysis as in Table 4.2.

| Objects     | Fixed pose prior (eval. protocol by [214]) |       |                   |       |                      |              | Random pose prior |       |                   |       |                      |       | No pose prior |              |
|-------------|--------------------------------------------|-------|-------------------|-------|----------------------|--------------|-------------------|-------|-------------------|-------|----------------------|-------|---------------|--------------|
|             | iNeRF [214]                                |       | NeMo + VoGE [172] |       | Parallel iNeRF [169] |              | iNeRF [214]       |       | NeMo + VoGE [172] |       | Parallel iNeRF [169] |       | 6DGS (Ours)   |              |
|             | MAE ↓                                      | MTE ↓ | MAE ↓             | MTE ↓ | MAE ↓                | MTE ↓        | MAE ↓             | MTE ↓ | MAE ↓             | MTE ↓ | MAE ↓                | MTE ↓ | MAE ↓         | MTE ↓        |
| Barn        | 26.5                                       | 0.208 | 51.2              | 0.752 | <b>22.9</b>          | <b>0.131</b> | 89.2              | 0.682 | 92.5              | 0.684 | 85.2                 | 0.572 | <u>30.3</u>   | <u>0.162</u> |
| Caterpillar | 42.9                                       | 0.166 | 52.6              | 0.516 | <u>25.2</u>          | <u>0.138</u> | 89.3              | 2.559 | 90.5              | 2.559 | 86.8                 | 0.843 | <b>14.5</b>   | <b>0.027</b> |
| Family      | 42.8                                       | 0.794 | 58.4              | 1.130 | <u>22.9</u>          | <u>0.507</u> | 93.9              | 1.505 | 97.0              | 1.506 | 99.0                 | 2.028 | <b>20.6</b>   | <b>0.468</b> |
| Ignatius    | 31.4                                       | 0.723 | 51.2              | 1.193 | <u>23.4</u>          | <u>0.604</u> | 84.1              | 1.489 | 85.4              | 1.491 | 86.9                 | 1.326 | <b>15.5</b>   | <b>0.441</b> |
| Truck       | 31.6                                       | 0.370 | 54.6              | 1.236 | <u>29.4</u>          | <u>0.351</u> | 94.4              | 1.042 | 97.7              | 1.045 | 97.6                 | 0.883 | <b>27.5</b>   | <b>0.242</b> |
| Avg.        | 35.0                                       | 0.452 | 53.6              | 0.965 | <u>24.7</u>          | <u>0.346</u> | 90.2              | 1.455 | 92.6              | 1.457 | 91.1                 | 1.130 | <b>21.7</b>   | <b>0.268</b> |

results, especially under identical comparison conditions (*i.e.*, random pose prior). Even when initialized from a fixed pose proximal to the known camera, 6DGS still excels over baselines in most scenes. Parallel iNeRF demonstrates improvement over iNeRF across all tested scenarios, consistent with its reported enhancements, but both methods’ performance drops with random initialization. Likewise, NeMo+VoGE performs worst, especially with random pose prior due to the utilization of a smaller number of larger ellipsoids in their approach. In contrast, 6DGS leverages approximately 300,000 ellipsoids of varying sizes obtained via 3DGS, as opposed to their mesh-to-ellipsoid method, which utilizes only about 5,000 larger ellipsoids. This fundamental disparity in ellipsoid size and quantity is a crucial factor contributing to the performance difference. Additionally, 6DGS exhibits faster processing speeds, operating nearly in real-time at 15 *frames per second* (*fps*) compared to the 0.05*fps* of Parallel iNeRF and 0.16*fps* of iNeRF. Please refer to the supplementary material for the complete table on timings.

**Qualitative Analysis:** Fig. 4.10 illustrates qualitative findings revealing notable observations. Particularly, we notice that the estimated poses exhibit proximity to the object relative to ground truth, attributable to the quantization effect introduced by the Ellicell. The qualitative findings verify the quantitative outcomes, albeit occasional inconsistencies in results, such as in the Counter scene, with the analysis-by-synthesis approaches showcasing a total incoherent output in regards to the overall scene (notice how the estimated poses are completely off the target). Moreover, the performance of 6DGS demonstrates consistency across varied scenarios, encompassing single-object instances and indoor settings, despite substantial variations in the models utilized. Fig. 4.12 shows the Tanks&Temples dataset [3] scenes *Barn*, *Caterpillar*, *Ignatius*, and *Truck*. On the other hand, Fig. 4.11 presents views from the *Garden*, *Kitchen*, *Stump*, *Room*, *Bicycle*, and *Counter* scenes of the Mip-NeRF 360° dataset [5]. In each figure, we show the ground truth camera position (green) and our estimated camera pose (red) in addition to the baseline methods (see legend for camera colors and details). In addition to the 3D camera positions estimated by our method and the other baselines, we also show the rendered images using the 3DGS model from ground truth and estimated 6DGS camera pose. From the rendered images, we can assess visually how the synthesized image viewpoint is different from the tested target image.

Looking closer at Fig. 4.12, our method predicts poses with a more accurate rotation, aligning closer to the ground truth. The translation can be slightly less accurate, resulting in either being slightly closer, *e.g.* for the *Caterpillar*, *Ignatius*, and *Truck* scenes, or further, *e.g.* for the *Barn* scene, from the object. A similar estimation of

## 4.2. 6DoF using 3D Gaussian Splatting

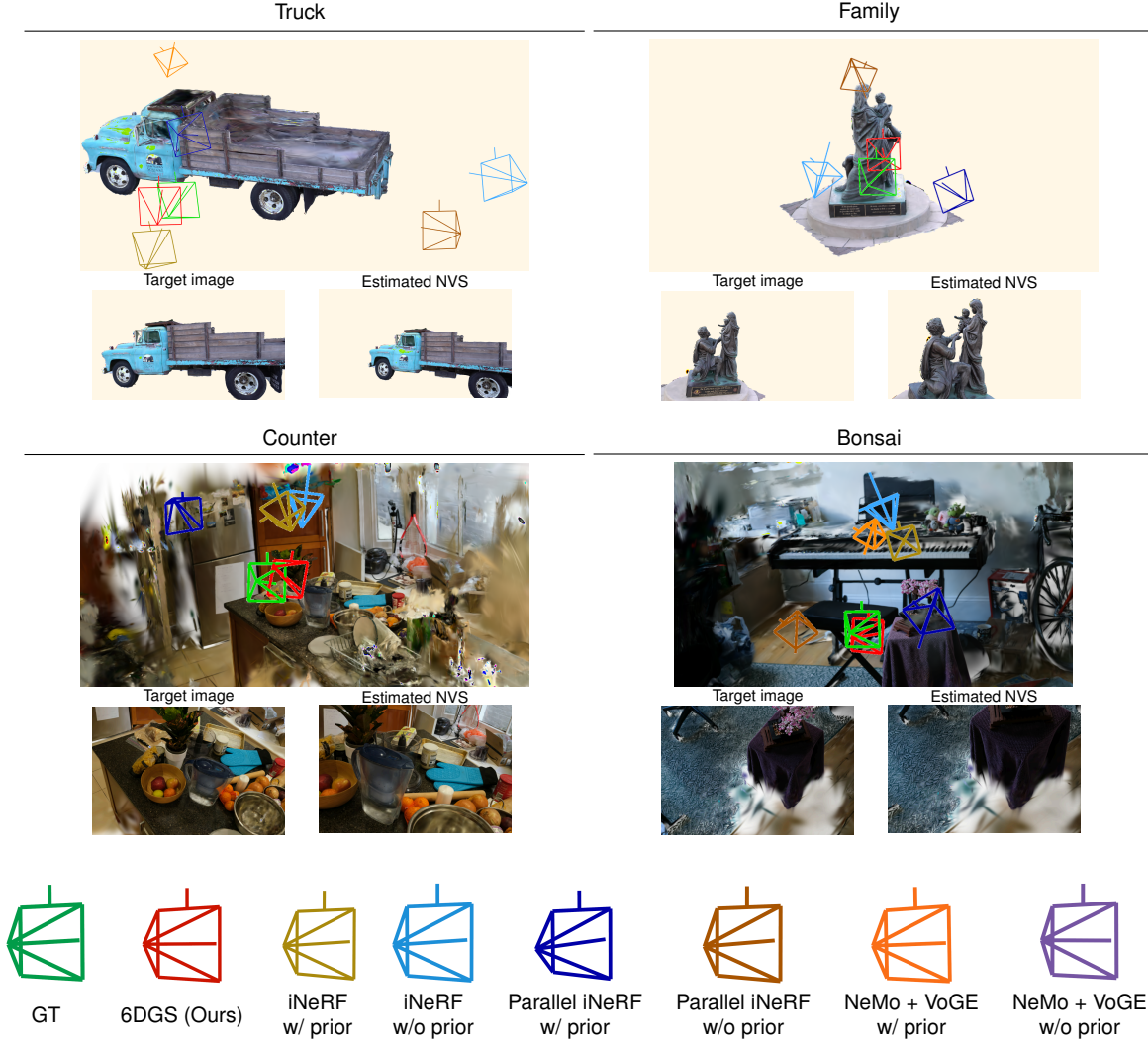


Figure 4.10: The illustration presents qualitative results from Tanks & Temple (upper row) and Mip-NeRF 360° (lower row) datasets. Each scene showcases the target images (bottom left) along with their corresponding Novel View Synthesis (NVS) outputs (bottom right), derived from the camera poses estimated by 6DGS (located on the top). Furthermore, the estimated camera poses from the comparative baselines are visualized, with distinct colors as indicated in the image legend. The NVS of each scene is rendered based on the provided 3DGS model. Please check the supplementary material for more qualitative results.

accuracy in rotation can be seen for the scenes *Kitchen*, *Stump* of the Mip-NeRF 360° dataset, see in Fig. 4.11 while showing a closer camera position though in a more limited amount. Mip-NeRF 360° also showcases a degradation in performance on camera rotation estimation related to the scenes *Room*, *Stump*, *Garden*. As for the baselines, in general, we can see how Parallel iNeRF tends to be more precise compared to the ones from iNeRF, but it can still fail in the complex scenes as shown. Finally, for NeMo

**Table 4.4** Ablation study on the number of rays selected for pose estimation using Mip-NeRF 360° [5]. Underline indicates the default values used.

| $N_{top}$  | MAE (°)↓    | MTE (u)↓      | Time (s)    |
|------------|-------------|---------------|-------------|
| 20         | 29.0        | 0.0235        | 0.03        |
| 50         | 26.3        | 0.0227        | 0.04        |
| <u>100</u> | <u>24.3</u> | <u>0.0217</u> | <u>0.06</u> |
| 150        | 24.4        | 0.0219        | 0.9         |
| 200        | 24.5        | 0.0222        | 0.11        |

**Table 4.5** Ablation study on the number of rays cast from each ellipsoid using Mip-NeRF 360° [5]. Underline indicates the default values used.

| # of cast rays | MAE (°)↓    | MTE (u)↓      | Time (s)    |
|----------------|-------------|---------------|-------------|
| 20             | 29.0        | 0.0235        | 0.04        |
| 35             | 24.7        | 0.0220        | 0.04        |
| <u>50</u>      | <u>24.3</u> | <u>0.0217</u> | <u>0.06</u> |
| 65             | 25.1        | 0.0218        | 0.09        |
| 80             | 25.2        | 0.0221        | 0.15        |

+ VoGE, the qualitative examples are consistent with the quantitative results, which tend to produce poses with a higher error than the other approaches.

### Ablation studies

Our ablation studies involve the analysis of the number of rays selected for the pose estimation (Tab. 4.4), the number of rays that we cast from a Ellicell (Tab. 4.5) as well as the different feature size on the MLP channels (Tab. 4.6). We find that the number of selected rays mainly affects the angular error, while the translation error remains relatively stable. Increasing the number of rays decreases the angular error but slightly increases the translation error, likely due to less confident rays contributing to the pose estimation. The optimal balance between translation and angular errors is achieved between 100 to 150 rays, with 100 being the best. The slight increase in error with more  $N_{top}$  rays is due to introducing rays not pointing precisely to the camera’s optical center. Similar to what we observed in the qualitative examples, the noisy rays make the weighted Least Squares estimating the camera closer to the object than GT.

Regarding the impact of the varying number of rays cast from the Ellicells, the angular error tends to remain relatively constant across different configurations. In contrast, the translation error decreases when 50 cast rays are used, and then increases again. This behavior is connected to network generalization capability. Increasing the number of rays allows the network to fit the training set better, but at test time, it makes the network more prone to noise and consequently selecting the wrong rays, thus increasing the error. We observe this generalization issue when increasing the MLP channels, see Tab. 4.6, particularly given the limited and uneven distribution of training images ( $\approx 150$ ). Moreover, the processing time increases proportionally with the number of rays and the MLP channels; upon exceeding the default values for rays

#### 4.2. 6DoF using 3D Gaussian Splatting

**Table 4.6** Ablation study on the MLP channel feature size using Mip-NeRF 360° [5]. Underline indicates the default values used.

| MLP channels | MAE (°)↓    | MTE (u)↓      | Time (s)    |
|--------------|-------------|---------------|-------------|
| 256          | 29.4        | 0.0273        | 0.04        |
| <u>512</u>   | <u>24.3</u> | <u>0.0217</u> | <u>0.06</u> |
| 1024         | 30.1        | 0.0228        | 0.27        |

**Table 4.7** Ablation study on 6DGS robustness to noise in the Gaussian centers. Results show performance under different noise levels added to the ellipsoid centers on the Bonsai and Bicycle scenes.

| Gauss. noise (‰) | MAE (°) | MTE (u) |
|------------------|---------|---------|
| 0.0‰u (None)     | 11.3    | 0.024   |
| 0.1‰u            | 18.3    | 0.025   |
| 0.5‰u            | 22.5    | 0.025   |

**Table 4.8** Ablation study on 6DGS performance with reduced viewpoints. Results demonstrate pose estimation accuracy when training with fewer camera views on the Bonsai and Bicycle scenes.

| No. views | MAE (°) | MTE (u) |
|-----------|---------|---------|
| All       | 11.3    | 0.024   |
| 12        | 15.5    | 0.024   |
| 6         | 18.6    | 0.026   |



and feature size, a notable surge in processing time is observed, reaching approximately  $10fps$  and  $13fps$ , respectively.

To assess our method’s robustness to 3DGS model quality, we conducted two experiments using the Mip-NeRF 360° dataset’s Bonsai and Bicycle scenes. First, we investigated the effect of 3DGS model accuracy on pose estimation by introducing varying noise levels to the Gaussian centers (Tab. 4.7). Second, we analyzed the performance of our method under sparse viewpoint conditions ( (Tab. 4.8)). The results in Tab. show that 6DGS is resilient for translation and minimal increase for rotational error, both with high noise level and even with only six views ( $\approx 60^\circ$  between viewpoints).

### Inference time

In Tab. 4.9 we detail the timings of each method. It can be noted from the table that our approach is the only one that is able to run under  $1s$  and near to real-time performance at 16FPS. In general, it is possible to notice how time slightly increases in the Mip-NeRF 360° dataset due to the higher complexity of the scenes. The increase in computational complexity can be attributed to several factors, including the size of the neural models required for iNeRF and Parallel iNeRF, as well as the quantity of ellipsoids used in NeMo+VoGE for each of the two datasets (*i.e.*  $\approx 150000$  ellipsoids for Tanks&Temples and  $\approx 300000$  for Mip-NeRF 360°, while in NeRF models Tanks&Temples use  $\approx 50\%$  less parameters than Mip-NeRF 360°). As it can be noticed 6DGS is able to handle both datasets and especially the the Mip-NeRF 360° scenes without issues.

Considering the different methods, we can notice that NeMo+VoGE requires the most time. This derive from the processing algorithm that struggles with large number of ellipsoids. In regards to the analysis-by-synthesis methods, iNeRF represents the fastest method, with Parallel NeRF being the second fastest. However, all methods show a considerable higher running time in contrast to 6DGS.

**Table 4.9** Average computation time to estimate the pose of an image. Comparison between our method and state-of-the-art approaches across the two datasets. Best-performing results are highlighted in **bold** and second-best results are underlined. Time is reported in seconds.

|                                      | Tanks & Temples | Mip-NeRF 360 |
|--------------------------------------|-----------------|--------------|
| iNeRF (pose prior by [214])          | 6.321           | <u>8.214</u> |
| NeMo+VoGE (pose prior by [214])      | 251.4           | 290.4        |
| Parallel iNeRF (pose prior by [214]) | 16.9            | 16.8         |
| iNeRF (random pose)                  | <u>6.2</u>      | 8.5          |
| NeMo+VoGE (random pose)              | 240.5           | 284.4        |
| Parallel iNeRF (random pose)         | 16.5            | 17.1         |
| 6DGS (Ours) (no pose)                | <b>0.05</b>     | <b>0.06</b>  |

## 4.3 Chapter conclusions

This chapter presented two complementary approaches to 6DoF camera pose estimation based on Novel View Synthesis techniques, addressing key limitations of traditional pose estimation methods. Both approaches eliminate the need for explicit depth information or CAD models, offering robust solutions for challenging scenarios involving transparent, reflective, or specular objects. IFFNeRF and 6DGS represent significant advancements over existing methods, each leveraging different NVS frameworks to achieve state-of-the-art performance.

The methods share a common principle: instead of relying on iterative analysis-by-synthesis optimization, they employ ray sampling by attention to efficiently estimate camera poses from a single image. This approach effectively transforms the pose estimation problem from a slow convergence-dependent process to a more direct computation that requires no initialization. By casting rays from scene surfaces and utilizing attention mechanisms to correlate rays with image features, both methods achieve a more robust and efficient estimation pipeline. The attention-based ray selection mechanism proved crucial in both approaches, efficiently mapping between image pixels and generated rays to identify relevant correspondences. This mechanism enables accurate pose estimation without the computational burden of iterative rendering and optimization, representing a shift in how NVS models can be leveraged for pose estimation.

IFFNeRF, built on Neural Radiance Fields, demonstrated substantial improvements over previous NeRF-based methods, reducing angular and translation errors by 80.1% and 67.3% respectively compared to iNeRF. The method achieves real-time performance at 34fps on consumer hardware while eliminating the need for an initial pose guess. 6DGS achieved up to 22% better accuracy while being approximately 94 times faster than traditional methods.

The primary limitation of both approaches is the need for scene-specific training, which could be addressed in future work through meta-learning techniques, especially when considering similar objects or scenes.

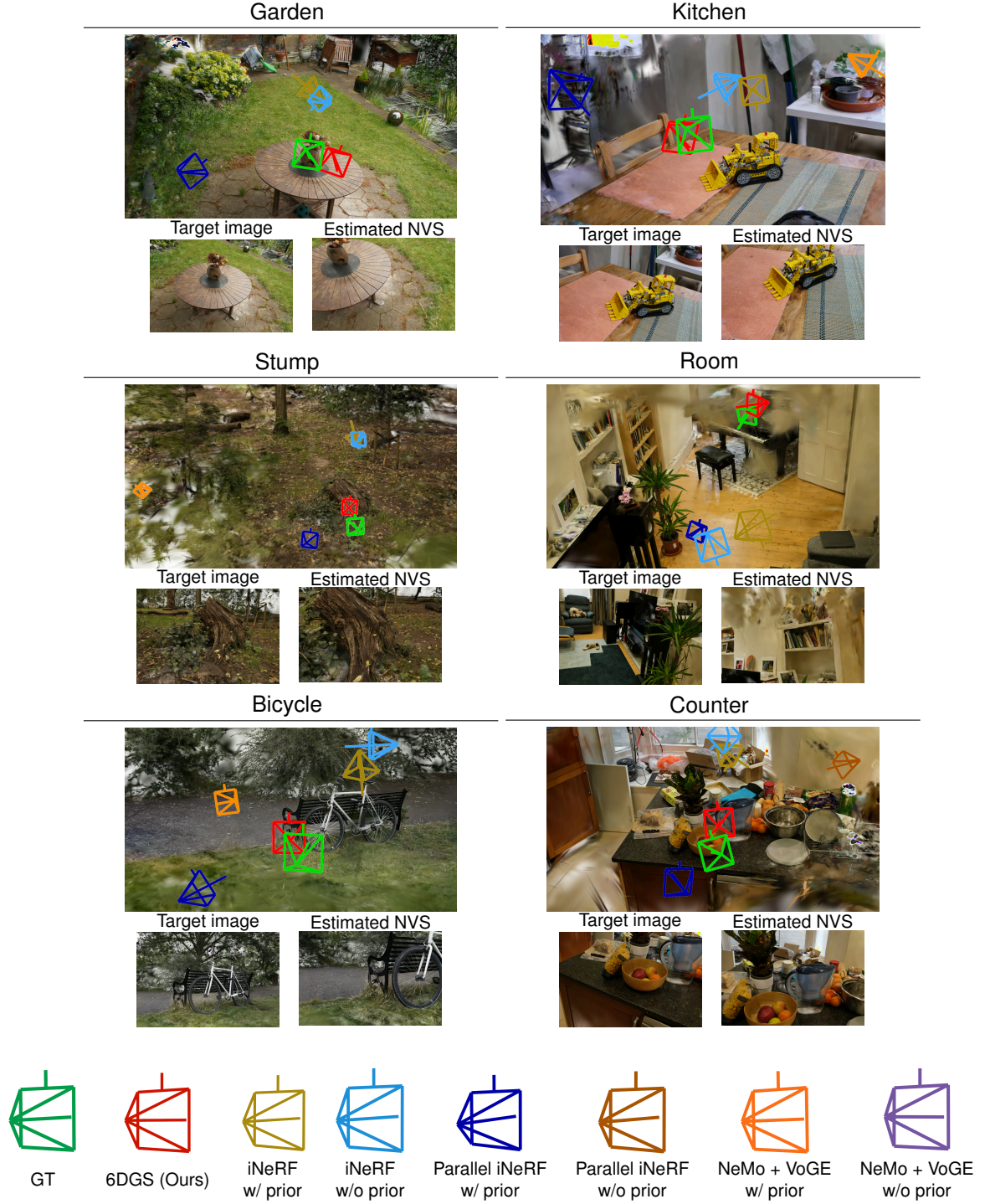


Figure 4.11: Additional scenes from the Mip-NeRF 360° dataset. For each scene, we show a visualization of the camera poses in regards to the model (top) for 6DGS as well as the baselines, which are visualized with different colors as indicated in the image legend. In addition, for each scene, we showcase the target image (bottom left) along with their corresponding 3DGS Novel View Synthesis (NVS) output (bottom right) of the estimated camera pose by 6DGS.

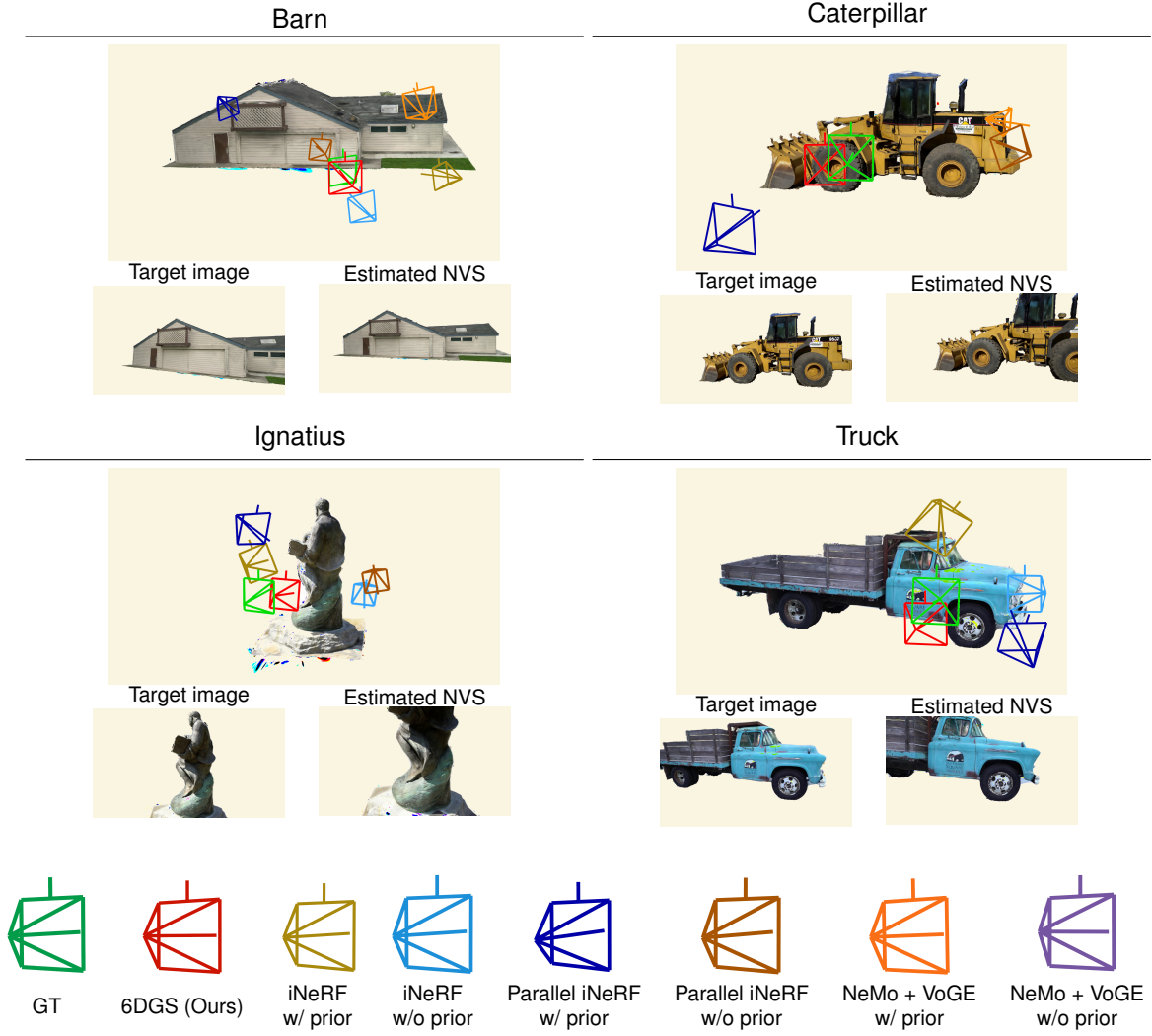


Figure 4.12: Additional scenes from the Tanks&Temple dataset. For each scene, we show a visualization of the camera poses in regards to the model (top) for 6DGS as well as the baselines, which are visualized with different colors as indicated in the image legend. In addition, for each scene, we showcase the target image (bottom left) along with their corresponding Novel View Synthesis (NVS) output (bottom right) of the estimated camera pose by 6DGS.

# Chapter 5

## Robotic grasping

Accurate and reliable grasping is a critical capability for autonomous robots operating in both industrial and domestic environments. As discussed in Chapter 1, robotic manipulation presents significant challenges, particularly when dealing with everyday objects that exhibit diverse shapes, sizes, and surface properties. While Chapters 3 and 4 focused on object reconstruction and pose estimation respectively, this chapter addresses the final crucial component of the robotic manipulation pipeline: grasp pose prediction. Traditional approaches to robotic grasping rely heavily on explicit 3D models and depth information, which limits their applicability when dealing with optically challenging objects such as transparent glasses or reflective metals. Furthermore, dexterous multi-fingered robotic hands add layers of complexity to the grasping problem due to their high degrees of freedom and the need for precise finger placement. This chapter explores how Novel View Synthesis (NVS) techniques can be leveraged to enhance dexterous grasp prediction, allowing robots to better handle a wide range of objects in unstructured environments.

Building upon the NVS foundations established in previous chapters, we introduce GRASPLAT, a novel approach that integrates 3D Gaussian Splatting (3DGS) into the grasp prediction process. This method enables the system to visualize and optimize grasp poses for multi-fingered robotic hands during the training phase, significantly improving both accuracy and adaptability.

### 5.1 Novel View Synthesis for dexterous grasping

The recent emergence of new humanoid platforms underscores the need for efficient solutions to enable grasping with multi-fingered robotic hands. While other grippers, such as parallel jaws, can achieve remarkable accuracy using extensive datasets [208], they still struggle with handling everyday objects and diverse materials [259, 260].

To estimate a successful dexterous grasp, recent approaches transform the target object into a 3D model (such as a mesh or point cloud) and use a neural network to predict one or more potential grasping poses [232, 233, 261, 262]. However, these methods present several challenges:

1. obtaining complete 3D scans often requires multiple viewpoints, which may not be feasible in real-world test scenarios as seen in Chapter 3
2. generating accurate grasp poses can be time-consuming, requiring substantial 3D digitization effort, which limits the number of objects that can be processed
3. depth sensors can struggle with transparent or reflective objects, and the time needed to acquire and process 3D models of novel objects can be prohibitive for real-time applications [1].

To mitigate these challenges, other approaches have explored solutions that can predict hand poses directly from RGB images [49, 234]. However, RGB-based methods often struggle with precision, as they rely solely on visual image data. Without explicit 3D reasoning, object-hand collisions are more likely to occur.

In this work, we propose a dexterous grasping method, GRASPLAT, that leverages 3D information while being trained solely on images. The core idea is: *if we can synthesize a physically plausible image of a hand grasping an object, then we can then regress the hand joints necessary for a successful grasp*. To achieve this, we utilize advanced novel view synthesis techniques, specifically 3DGS [78], which encapsulates a coherent 3D model capable of generating novel images of real hand-object grasps and their corresponding joint parameters. This innovative step enables us to incorporate a photometric loss based on the rendered image of the grasped object, hence to train our model end-to-end using only RGB data.

GRASPLAT follows a similar initial step to other RGB-only approaches: a neural network predicts the 3D parameters of the human hand model, MANO [263]. However, unlike these methods, we leverage 3DGS’s ability to generate high-fidelity scene representations that include both the object and the hand. The hand model is articulated and follows the MANO skeleton, linking the neural network’s predictions to the MANO model. As a result, the appearance of the hand adapts based on the predictions. Inspired by analysis-by-synthesis techniques [214], we introduce an additional photometric loss term during training, which compares the predicted image with the ground truth. This allows the model to continuously refine its grasp predictions by minimizing the discrepancy between the rendered and real images, thereby enhancing both accuracy and robustness in complex scenarios.

We evaluate our method through extensive experiments on synthetic and real-world grasping datasets, demonstrating a significant improvement in grasp success rates over baseline approaches, up to 36.9 % on YCB. Our ablation studies highlight the critical role of the photometric loss term, confirming that it substantially enhances the precision of grasp pose predictions. By leveraging the strengths of 3DGS, our method provides an effective way to improve general dexterous robotic grasping.

In summary, our contributions are as follows:

- We introduce a novel view synthesis solution for grasping based on 3DGS, which refines hand pose predictions through a photometric loss.
- We develop a training pipeline that, despite requiring only an RGB image during inference, leverages the 3D spatial information from the 3DGS model.

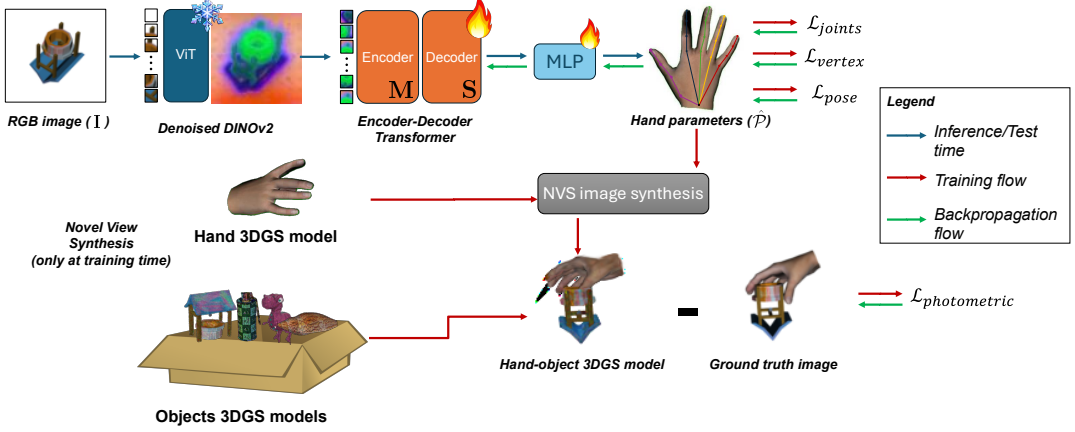


Figure 5.1: GRASPLAT predicts 3D hand poses from a single RGB image using a frozen, denoised DINOv2 backbone. The extracted feature map is processed by a spatial transformer to estimate hand parameters, which are then refined by an MLP. During training, a feedback mechanism based on 3D Gaussian Splatting (3DGS) refines predictions by comparing rendered and real images. To accelerate training, 3DGS models are precomputed before training and then cached. The rendered scene includes both the object and the articulated hand, posed according to the predicted parameters.

- We create a large-scale synthetic dataset based on GraspXL [264], enabling the training of articulated 3DGS models for hand-object interactions.

### 5.1.1 Approach

Our approach estimates a 3D hand pose, denoted as  $\hat{\mathcal{P}}$ , from an RGB image  $\mathbf{I}$ . The hand pose follows the MANO model format [263], i.e.  $\hat{\mathcal{P}} = \hat{\mathbf{Y}}, \hat{\mathbf{t}}, \hat{\mathbf{h}}$ . Here,  $\hat{\mathbf{Y}} \in \text{SO}(3)$  represents the global rotation of the hand at the base,  $\hat{\mathbf{t}} \in \mathbb{R}^3$  denotes the translation, and  $\hat{\mathbf{h}} \in \mathbb{R}^{21 \times \text{SO}(3)}$  indicates the relative rotation of each joint with respect to its parent.

Fig. 5.1 shows the proposed approach block diagram. GRASPLAT processes  $\mathbf{I}$  to obtain a feature map using a Vision Transformer (ViT). This feature map is fed into a transformer-based encoder-decoder architecture that predicts the MANO parameters  $\hat{\mathcal{P}}$  (Sec.5.1.1). During training, our model learns to output the hand parameters while being constrained by additional loss terms (Sec. 5.1.1). We introduce a module that combines a 3DGS model of the object with an articulated hand model in a differentiable manner. The hand model is controlled by parameters predicted by a neural network. The combined model is then rendered from a randomly chosen viewpoint during training. We compare this rendered view with a real scene image taken from the same viewpoint, where the hand has grasped the object. The comparison uses photometric error as the evaluation metric. The differentiability of 3DGS enables us to backpropagate the error to the neural network’s predictions (Sec. 5.1.1). Given the lack of a dataset combining grasping and 3DGS object models, we create our own dataset based on GraspXL [264] (Sec. 5.1.2). At test time, the network generates the hand parameters



for the multi-fingered robotic hand.

### Neural Network Architecture

We compute the feature map  $\mathbf{F}$  from the image  $\mathbf{I}$  using the pretrained DINOv2 Vision Transformer (ViT-Base/14) [60]. We use a variant of DINOv2 called the Denoised Vision Transformer (DVT) [265]. Specifically, DVT adds an extra layer after the DINOv2 output to mitigate noise caused by the positional encoding inherent to transformers.

Since dexterous grasping requires high precision, we enhance the last layers of our neural network. We use an input image size of  $518 \times 518$  as opposed to the typical  $224 \times 224$ . The image is then patchified, with each patch being processed individually. While we retain the standard patch size of  $14 \times 14$ , we reduce the stride from  $14 \times 14$  to  $7 \times 7$  to increase the resolution of the feature map. This modification expands the feature map  $\mathbf{F}$  from its original size of  $C \times 16 \times 16$  to  $\mathbf{F} \in \mathbb{R}^{C \times 73 \times 73}$ , where  $C$  denotes the number of channels. We denote the width and height of the feature map as  $E = 73$  and  $T = 73$ , respectively.

To enhance spatial awareness, we concatenate a 2D positional encoding to each output feature [58]:

$$PE(x, y) = \begin{bmatrix} \sin\left(\frac{x}{\tau^{\frac{2i}{D}}}\right) \\ \cos\left(\frac{x}{\tau^{\frac{2i+1}{D}}}\right) \end{bmatrix}, \quad \forall i \in [0, D/2), \quad (5.1)$$

where  $x$  and  $y$  are the 2D coordinates,  $D = \lfloor C/2 \rfloor$  is positional encoding dimension,  $i$  indexes the encoding dimension, and  $\tau=10000$  is the temperature parameter.

We then fed these features into an encoder-decoder transformer [57] to predict hand coordinates. Due to its structure, the encoder-decoder output a spatial map  $\mathbf{M} \in \mathbb{R}^{21 \times E \times T}$  and a global representation  $\mathbf{S} \in \mathbb{R}^C$ . The representation  $\mathbf{M}$  provides spatial information, where each channel is designed to encode the planned position of a corresponding joint. Instead,  $\mathbf{S}$  contains the overall information regarding the hand, useful to resolve ambiguities inside  $\mathbf{M}$ .

We employ a heatmap-based approach to extract joint coordinates [266]. The heatmap is computed as  $\mathbf{H}_j = \text{softmax}(\beta_j \mathbf{M}_j)$ , where  $j$  indexes the joint, and  $\beta_j$  is a learned parameter that adjusts the weighting of each channel. The final 2D coordinates for each joint are obtained via:

$$\mathbf{p}_j = \sum_u^T \sum_v^E \mathbf{H}_{j,u,v} \begin{bmatrix} u \\ v \end{bmatrix}. \quad (5.2)$$

Each joint position  $\mathbf{p}_j$  is concatenated with  $\mathbf{S}$  then processed by a 4-layer MLP to estimate the global rotation  $\hat{\mathbf{Y}}$ , per-joint rotations  $\hat{\mathbf{h}}$ , and the global translation  $\hat{\mathbf{t}}_n$ . To ensure numerical stability, we normalize the translation with a mean of 0 and a standard deviation of 1. The final translation is then de-normalized as  $\hat{\mathbf{t}} = \hat{\mathbf{t}}_n \sigma + \mu$ , where the standard deviation  $\sigma \in \mathbb{R}^3$  and the mean  $\mu \in \mathbb{R}^3$  are computed from the training dataset.



### Loss functions for hand parameters prediction

During training, multiple loss functions are used to supervise hand prediction. Sec. 5.1.2 provides details on how to efficiently generate the ground truth data for training GRASPLAT. Some loss functions are based on the vertex  $\mathbf{V} \in \mathbb{R}^{768 \times 3}$  and joints  $\mathbf{J} \in \mathbb{R}^{21 \times 3}$  computed using the MANO [263] parametric model. In particular, a 3D vertex loss enforces geometric consistency between predicted  $\hat{\mathbf{V}}$  and ground-truth mesh vertices  $\mathbf{V}$  of MANO:

$$\mathcal{L}_{verts} = \lambda_{verts} \cdot \frac{1}{768} \sum_{i=1}^{768} \|\hat{\mathbf{V}}_i - \mathbf{V}_i\|_2^2, \quad (5.3)$$

where  $\lambda_{verts} = 1 \times 10^{-4}$  is a balance factor. The 3D joint loss aligns skeletal joint positions using:

$$\mathcal{L}_{joints} = \lambda_{joints} \cdot \frac{1}{21} \sum_{i=1}^{21} \|\hat{\mathbf{J}}_i - \mathbf{J}_i\|_2^2, \quad (5.4)$$

where  $\lambda_{joints} = 1 \times 10^{-4}$ . We also directly supervise the MANO parameters, including a loss term for rotation:

$$\mathcal{L}_{pose} = \lambda_{pose} \cdot \frac{1}{21} \sum_{i=1}^{21} \|\hat{\mathbf{Y}}_i - \mathbf{Y}_i\|_2^2, \quad (5.5)$$

where  $\lambda_{pose} = 10.0$  balances the rotation component. We also include a term to supervise translation:

$$\mathcal{L}_{transl} = \lambda_{transl} \cdot \frac{1}{21} \sum_{i=1}^{21} \|\hat{\mathbf{t}}_i - \mathbf{t}_i\|_2^2, \quad (5.6)$$

where  $\lambda_{transl} = 10.0$  weights the translation component. A pose regularization term discourages unnatural hand configurations by penalizing deviations from the zero rotation:

$$\mathcal{L}_{reg} = \lambda_{reg} \cdot \frac{1}{21} \sum_{i=1}^{21} \|\mathbf{I}_i - \hat{\mathbf{Y}}_i\|_2^2, \quad (5.7)$$

where  $\mathbf{I} \in \mathbb{R}^{3 \times 3}$  is the identity matrix, and  $\lambda_{reg} = 1.0$ . Lastly, the MANO part of the loss function is computed as:

$$\mathcal{L}_{mano} = \mathcal{L}_{verts} + \mathcal{L}_{joints} + \mathcal{L}_{pose} + \mathcal{L}_{transl} + \mathcal{L}_{reg}. \quad (5.8)$$

### NVS photometric loss for grasping

GRASPLAT uses a parameterized NVS model for articulated hands based on 3DGS that deforms according to a mesh grid. We deform the 3DGS model using the mesh structure from the MANO parametric model. Such NVS model makes it possible to link the hands parameters to image generation and thus simplifying our training strategy using only images of hand-objects interactions.

*Brief 3DGS preliminaries.* The primary objective of 3DGS is to synthesize novel views

of a scene using a set of 3D Gaussians, denoted as  $\mathcal{Q} = \{\mathbf{Q}_i\}_{i=1}^K$ . Each Gaussian is characterized by its position  $\mathbf{x} \in \mathbb{R}^3$ , orientation  $\mathbf{R} \in SO(3)$ , scale represented as a diagonal matrix  $\mathbf{S} \in \mathbb{R}^{3 \times 3}$ , and a color. A 3DGS model is computed starting from a set of input images and their from different viewpoints. The 3DGS model is optimized through an iterative process in which the Gaussians  $\mathbf{Q}$  are projected onto the image plane and transformed into an image using the differentiable rasterization function  $\phi$  [78]. The resulting image is then compared with the original image by minimizing a photometric error, and the parameters are optimized via backpropagation. For more details on the process, please refer to [78].

*Hand 3DGS model for GRASPLAT.* A standard 3DGS approach can synthesize images for rigid scenes only, so we upgraded 3DGS to generate images of articulated hands for grasp predictions. Specifically, any update to the MANO parameters  $\hat{\mathcal{P}} = \hat{\mathbf{Y}}, \hat{\mathbf{t}}, \hat{\mathbf{h}}$  results in a corresponding deformation of the 3DGS model. To achieve this, we associate each Gaussian  $\mathbf{Q}_i$  to the nearest face  $w$  of the MANO mesh. Consequently, the transformation of each Gaussian is dependent on its corresponding face of the deforming mesh:

$$\hat{\mathbf{x}}_i = \mathbf{T}_w + k_w \mathbf{G}_w \mathbf{x}_i, \quad (5.9)$$

$$\hat{\mathbf{R}}'_i = \mathbf{R}_w \mathbf{R}_i, \quad (5.10)$$

$$\hat{\mathbf{S}}'_i = k_w \mathbf{S}_i^0, \quad (5.11)$$

where  $\mathbf{x}_i$ ,  $\mathbf{R}_i$ , and  $\mathbf{S}_i$  represent the  $i$  Gaussian position, rotation, and scaling respectively in the rest pose of MANO. The transformation parameters  $\mathbf{T}_w \in \mathbb{R}^3$  (translation),  $\mathbf{G}_w \in SO(3)$  (rotation matrix), and  $k_w \in \mathbb{R}^3$  (scaling factor) are derived from the MANO parameters for face  $w$ . The translation parameters for each face  $w$  is calculated as follows:

$$\mathbf{T}_w = \frac{1}{3} \sum_{i=0}^2 \mathbf{V}_{w_i}, \quad (5.12)$$

where  $\mathbf{V}_{w_0}, \mathbf{V}_{w_1}, \mathbf{V}_{w_2}$  are the vertices of face  $w$  in the MANO mesh. The translation  $\mathbf{T}_w$  is computed as the centroid of these vertices. The rotation  $\mathbf{R}_w$  derives from an orthogonal basis  $\langle \cdot \rangle$  from the face's edge directions:

$$\begin{aligned} \mathbf{R}_w = \langle \mathbf{a}_1, \mathbf{a}_2, \mathbf{a}_3 \rangle, \quad \text{where} \quad & \mathbf{a}_1 = \mathbf{V}_{w_2} - \mathbf{V}_{w_1}, \\ & \mathbf{a}_2 = \mathbf{a}_1 \times (\mathbf{V}_{w_0} - \mathbf{V}_{w_1}), \\ & \mathbf{a}_3 = \mathbf{a}_1 \times \mathbf{a}_2. \end{aligned} \quad (5.13)$$

The scaling factor  $k_w$  is given by:

$$k_w = \frac{1}{2} \left( \|\mathbf{a}_1\|_2 + \frac{\|\mathbf{a}_2\|_2}{\|\mathbf{V}_{w_0} - \mathbf{V}_{w_2}\|_2} \right), \quad (5.14)$$

and determined by averaging the face area scaling factor  $\|\mathbf{a}_2\|_2$  and the edge length variations  $\|\mathbf{a}_1\|_2$ , ensuring robustness against significant distortions.

During training, we combine the hand and object 3DGS models using a simple concatenation  $\mathcal{Q}_{scene} = \mathcal{Q}_{hand} \cup \mathcal{Q}_{object}$ , where  $\mathcal{Q}_{hand}$  is given by the articulated hand 3DGS model, and  $\mathcal{Q}_{object}$  represents the rigid object 3DGS model. The combined 3DGS model  $\mathcal{Q}_{scene}$  is rendered using the rendering function  $\phi$ , producing the final image  $\hat{\mathbf{I}}$ . To prevent convergence to local minima, the viewpoint is randomly selected from a predefined set of viewpoints around the object. The corresponding target image is denoted as  $\mathbf{I}_{tgt}$  and the photometric loss is computed as:

$$\mathcal{L}_{img} = \lambda_1 \|\hat{\mathbf{I}} - \mathbf{I}_{tgt}\|_1 + (1 - \lambda_1)(1 - \text{SSIM}(\hat{\mathbf{I}}, \mathbf{I}_{tgt})), \quad (5.15)$$

where  $\lambda_1 = 0.8$  is a balancing factor between the L1 term, which ensures convergence due to its convex nature, and the SSIM metric [239], which, while non-convex, more closely aligns with human visual perception. This formulation enables backpropagation of the visualization error into the pose predictions, resulting in the loss:

$$\mathcal{L} = \alpha \mathcal{L}_{mano} + (1 - \alpha) \mathcal{L}_{img} \quad (5.16)$$

where  $\alpha = 0.5(1 + \cos(\pi o_{curr}/o_{last}))$  gradually activates the photometric error, and  $o_{curr}$ ,  $o_{last}$  are the current and last epoch.

### 5.1.2 GRASPLAT dataset

To train our model, we need to synthesize images of hand-object grasps from multiple viewpoints. We build upon the GraspXL [264] dataset, which contains 3D grasping sequences for 5,849 objects across various hand models. We extend GraspXL in several ways. First, we introduce a NVS pipeline that can generate images from arbitrary camera viewpoints. Then, we expand the object dataset by a factor of eight. Our focus is on grasp sequences using the MANO hand model. Since GraspXL provides only geometric data without the textures needed for image synthesis, we augment the object generation process with Objaverse [267], which allows us to significantly increase the number of textured objects. Similarly, the appearance of the MANO hand is reconstructed using the DART texture model [268]. To train the 3DGS model, we render 200 images per object from multiple viewpoints using Vedo [269]. Each object undergoes five grasping attempts with different orientations, and we simulate the sequence until the object moves. Given GRASPLAT’s focus on humanoid robots, the hand begins from the bottom right of the camera view. In total, the dataset consists of over 4.5 million images. Qualitative examples of the synthesized data are shown in Fig. 5.2.

### 5.1.3 Results

#### Experimental setup

*Datasets.* Novel view synthesis at training time requires a dataset with objects seen from multiple viewpoints. Likewise we need a 3DGS model for the hand as described in Sec. 5.1.1. Current datasets for object grasping such as [270] do not provide this

information, so the need to create the GRASPLAT dataset. Our dataset includes a reference pose to evaluate performance using the Mean Joint Error, which measures the average discrepancy between the predicted MANO joints and their ground truth counterparts. We compare methods based on rotation and translation errors at the base of the hand relative to the ground truth. These evaluations are performed on the test partition, which is obtained through an 80%-20% split ratio. Since our goal is to achieve successful grasping, we further assess performance using a physics simulator. For the test set, we measure the percentage of objects that can be successfully grasped using a QB SoftHand [205] model mapped onto the MANO model. We align both hands relative to their wrists and apply a  $5^\circ$  rotation along the y-axis to compensate for the original MANO’s skewness. Our simplified SoftHand model accurately simulates most of the QBHand’s behavior, including its underactuated joints. A grasp is considered successful when at least two fingers contact the object.

The quantitative results of these experiments are presented in Tab. 5.1. To evaluate generalization capabilities, we also conduct a zero-shot test on the YCB dataset [271] with randomly oriented objects. Since no reference hand poses are provided in YCB, we neither train nor fine-tune the model on YCB objects. Instead, we directly apply the network trained on our dataset to YCB objects, assessing its generalization performance. Due to the lack of ground truth hand poses in YCB, we report only grasping accuracy in Tab. 5.1.

*Comparison methods.* We compare GRASPLAT against GANHand and AffordanceDiffusion, two methods designed to predict a grasping pose from an RGB image. GANHand [49] directly estimate the pose in 3D, whereas AffordanceDiffusion [234] utilizes diffusion models to generate an image of the grasping interaction. To ensure a fair comparison, we retrained both methods on the GRASPLAT dataset using their original repositories. However, during the training of GANHand, we observed catastrophic failures. To stabilize the learning process, we enabled the forward kinematic losses present in the original GANHand implementation [49].

*GRASPLAT implementation details.* Our method is implemented in PyTorch and trained for 1.5K iterations ( $\sim 13$  hours) using a NVIDIA RTX 4500. The translation normalization is based on the objects’ size. We employ the AdamW optimizer [272] with a weight decay of  $10^{-2}$  and a learning rate of  $10^{-5}$ . A step learning rate scheduler is used, decaying every 1,900 iterations with a decay factor of 0.7. For 3D Gaussian Splatting (3DGS) rendering, we utilize gSplat [273]. The grasping simulations are implemented using IsaacGym [274].

## Experimental evaluation

Tab. 5.1 presents the results on GRASPLAT and YCB datasets. No method was fine-tuned for YCB so evaluation is zero-shot for this dataset. The first part of the table shows that GRASPLAT achieves state-of-the-art performance on both datasets, outperforming prior works. We achieve a 64% improvement in grasp accuracy on GRASPLAT and 41% on YCB. The small difference in translation and rotation errors across methods, compared to grasp accuracy, indicates that minor mispredictions can significantly

**Table 5.1** We report the grasp accuracy, mean joint error, relative rotation error (RRE), translation error, where applicable, along with inference time. Best-performing results are highlighted in **bold** and second-best results are underlined.

| Method                                                        | GRASPLAT dataset   |                       |              |                        | YCB [271]          | Time [s]    |
|---------------------------------------------------------------|--------------------|-----------------------|--------------|------------------------|--------------------|-------------|
|                                                               | Grasp accuracy [%] | Mean Joint Error [cm] | RRE [°]      | Translation Error [cm] | Grasp accuracy [%] |             |
| GANHand [49]                                                  | <u>37.1</u>        | 2.22                  | 16.23        | 7.38                   | <u>26.3</u>        | <b>0.26</b> |
| Affordance Diffusion [234]                                    | 34.2               | <u>2.11</u>           | <u>16.03</u> | <u>7.00</u>            | 0.0                | 0.83        |
| GRASPLAT (ours)                                               | <b>61.4</b>        | <b>1.58</b>           | <b>15.15</b> | <b>5.23</b>            | <b>63.2</b>        | <u>0.38</u> |
| $\Delta$ (best - second best)                                 | +24.3              | +0.53                 | +0.88        | +1.77                  | +36.9              | -0.12       |
| GRASPLAT (224×224 image resolution)                           | 60.0               | 1.59                  | 15.43        | 5.36                   | 52.6               | 0.13        |
| GRASPLAT (stride 14)                                          | 58.5               | 1.59                  | 15.19        | 5.26                   | 47.4               | 0.12        |
| GRASPLAT (w/o $\mathcal{L}_{pose}$ & $\mathcal{L}_{transl}$ ) | 60.1               | 1.66                  | 15.18        | 5.60                   | 62.9               | 0.38        |

**Table 5.2** Ablation study on GRASPLAT’s robustness to input image noise. Results show mean joint error and percentage increase compared to the noise-free baseline.

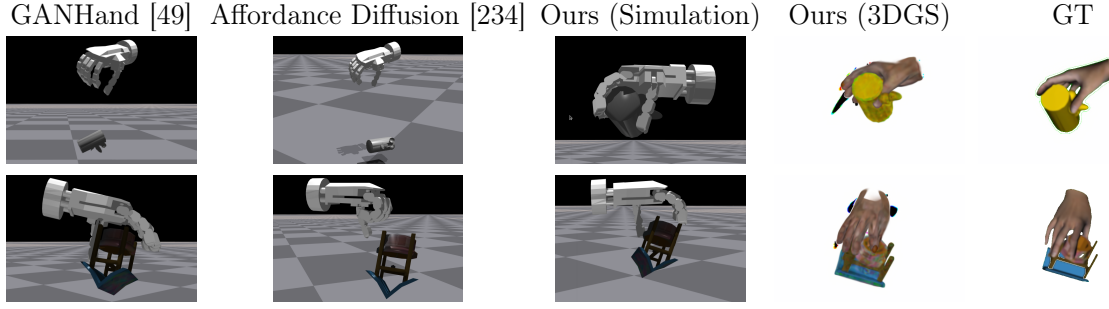
| Setup           | Mean Joint Error [cm] | Delta [%] |
|-----------------|-----------------------|-----------|
| <i>No noise</i> | 1.58                  | 0.0       |
| 0.1             | 1.64                  | 3.6       |
| 0.3             | 1.65                  | 4.6       |
| 0.5             | 1.82                  | 15.0      |

impact grasp success. This effect is evident in the inverted well toy from GRASPLAT (Fig. 5.2), where our method successfully grasps the object between the thumb and index finger despite the minimal contact surface.

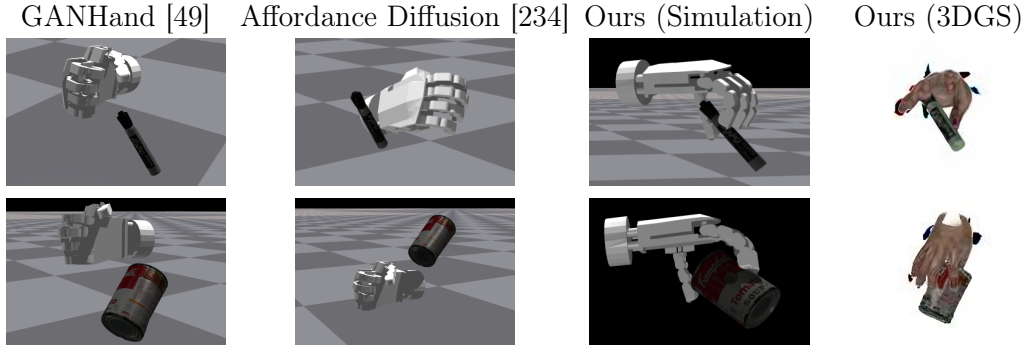
The zero-shot test on YCB further demonstrates GRASPLAT’s ability to generalize to out-of-distribution objects. In the same situation, GANHand has much lower performance due to higher translation errors that impair successful grasps. Fig. 5.2 shows that while Affordance Diffusion can estimate reliably the multi-fingered hands joint poses, but it severely struggles to estimate translation. This problem is due to the method reliance to an off-the-shelf hand pose estimator that is not able to generalise to different scenarios. This affects the estimation of the distance of the hand from the camera, and in turn leads to wrong translation estimates. This also affects accuracy, in many cases the hand collides with the object, which we consider a grasping failure. In our experiments, we found that despite the mapping the hand models differ slightly. For example, QBHand has a flat palm, while MANO’s is slightly convex. Even a small correction of 2 mm can cause a 1–2% performance drop.

The second part of Tab. 5.1 presents the ablation studies. Lowering image resolution (224×224) or stride marginally reduces accuracy on the training set but drastically impacts performance on new objects like those in YCB, likely due to the loss of fine detail reasoning. However, reducing stride or resolution significantly speeds up inference (134 ms), highlighting a performance-efficiency trade-off. Note that even with the fastest inference, GRASPLAT still best all the competing approaches in all the metrics. Additionally, removing direct hand parameter optimization losses ( $\mathcal{L}_{pose}$  and  $\mathcal{L}_{transl}$ ) has minimal impact on accuracy but increases convergence time.

To assess GRASPLAT’s robustness in real-world conditions, we evaluate its performance under different levels of image noise, as shown in Tab. 5.2. The results demon-



(a) GRASPLAT dataset



(b) YCB dataset

Figure 5.2: Comparison of methods on the GRASPLAT (top) and YCB (bottom) datasets. Each row represents a different object. For each of the compared methods, as well as for GRASPLAT, we present the results in the simulated environment. Additionally, for our method, we also show qualitative image synthesis of our 3DGS model during training against the relative ground truth.

strate remarkable resilience, with only 3.6% and 4.6% increases in mean joint error when subjected to noise levels of 0.1 and 0.3 respectively. This minimal degradation underscores the stability of our approach in handling typical imperfections encountered in practical robotics scenarios. Even with substantial noise (0.5), GRASPLAT maintains reasonable performance with a 15% increase in error. Such resilience is crucial for deployment in uncontrolled environments where lighting conditions, camera quality, and visual artifacts may vary significantly, and complements the grasp accuracy improvements shown in Tab. 5.1.

To further assess GRASPLAT’s robustness in realistic scenarios, Tab. 5.3 evaluates performance under varying degrees of occlusion. The results demonstrate exceptional resilience, with only minor performance degradation (2.0% to 7.3% increase in mean joint error) when up to 50% of the input image is occluded with random patches. Even with severe occlusion (70%), the model maintains reasonable performance with an 18.7% increase in error. This robustness is particularly valuable in cluttered environments where partial object visibility is common. Nevertheless, since the network was not trained on scenarios with occluded or partially obstructed objects, it cannot adapt its hand positioning strategy when other objects are present near the target object. This

**Table 5.3** Ablation study on GRASPLAT’s robustness to occlusions. We randomly occlude some part of the input image with patches of maximum 33 pixels, until a certain occlusion area is reached. Results show mean joint error and percentage increase compared to the occlusion-free baseline.

| Percentage of occlusion | Mean Joint Error [cm] | Delta [%] |
|-------------------------|-----------------------|-----------|
| 0%                      | 1.58                  | 0.0       |
| 10%                     | 1.61                  | 2.0       |
| 20%                     | 1.63                  | 3.4       |
| 30%                     | 1.67                  | 5.6       |
| 50%                     | 1.69                  | 7.3       |
| 70%                     | 1.87                  | 18.7      |

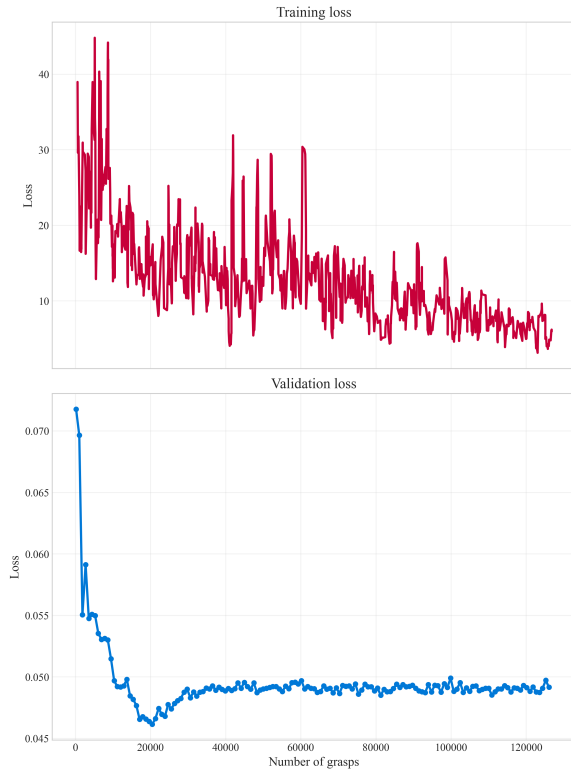
aspect is a potential improvement for future work, as it would allow the model to better handle real-world scenarios where objects are frequently occluded or partially visible.

Significant insight can be also shown by checking the training and validation trends in Fig. 5.3, which demonstrates how dexterous grasping shares fundamental similarities with language generation in LLMs rather than traditional computer vision tasks. Unlike pose estimation (Sec. 4) or classification problems where a single correct answer exists, dexterous grasping—like language completion—admits multiple equally valid solutions. Just as different word choices can appropriately complete a sentence while maintaining its meaning, multiple hand configurations can successfully grasp the same object with equivalent functional outcomes. This many-to-one mapping creates unique challenges for supervision and evaluation metrics. At the same time, sometimes even very near hand poses can lead to considerably different grasping outcomes (as shown in Fig 5.2). The photometric loss help in remove this ambiguities by acting as a supression score (similar to reinforcement learning) for nearby poses that can lead to negative outcomes.

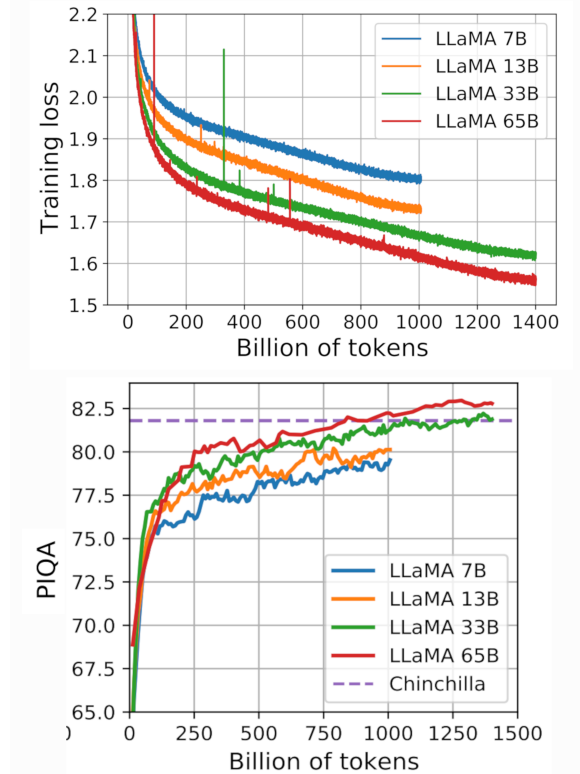
## 5.2 Chapter conclusions

This chapter explored Novel View Synthesis applications for robotic grasping through GRASPLAT, an approach that leverages 3D Gaussian Splatting to bridge geometry understanding and grasp prediction. By incorporating photometric loss during training, our method learns accurate hand poses from RGB images alone, without requiring depth information or complete 3D scans. This addresses key limitations of traditional approaches that struggle with optically challenging objects.

Experimental results show GRASPLAT outperforms existing methods significantly, with improvements up to 64% on our dataset and 36.9% on YCB in zero-shot evaluation. These results confirm that synthesized hand-object interactions provide effective supervision for grasp pose prediction. The integration of 3D Gaussian Splatting (3DGS) enables real-time performance while maintaining high accuracy. GRASPLAT completes our thesis’s robotic manipulation pipeline, complementing the sparse view reconstruction methods from Chapter 3 and pose estimation techniques from Chapter 4.



(a) GRASPLAT losses trends



(b) Llama [13] training and validation trend

Figure 5.3: Comparison of loss trends between GRASPLAT and an LLM, in this case Llama [13], note the similarity in the convergence patterns.



# Chapter 6

## Conclusions

This dissertation explored the application of Novel View Synthesis (NVS) techniques to enhance robotic perception and manipulation capabilities. By addressing key challenges in object reconstruction, pose estimation, and grasp planning, we have developed a comprehensive framework of methods that enables robots to operate more effectively in both industrial and domestic environments. The research presented in this thesis makes significant contributions to advancing the state-of-the-art in autonomous robotics, particularly in handling objects with challenging surface properties such as reflective, transparent, or highly specular surfaces.

We introduced VM-NeRF, a novel method for sparse novel view synthesis that integrates Neural Radiance Fields (NeRF) with View Morphing [6]. View Morphing is a geometry-based technique that does not require prior knowledge of 3D shape, relying instead on fundamental principles of projective geometry to interpolate between views. Our approach leverages the strengths of both paradigms, combining the geometric consistency of View Morphing with the volumetric rendering capabilities of NeRF. By integrating these complementary approaches, VM-NeRF effectively addresses limitations in data-sparse environments where traditional NeRF methods often struggle. Our comprehensive evaluation on standard NeRF-based datasets demonstrates that VM-NeRF effectively learns robust 3D scene representations across various sparse novel view synthesis scenarios. The results indicate significant improvements in reconstruction quality and view synthesis accuracy, particularly when working with limited input views.

Building upon this foundation, we then tackled the problem of 6DoF pose estimation from Novel View Synthesis by proposing IFFNeRF and 6DGS. IFFNeRF introduces a ray-sampling-by-attention method for estimating 6DoF camera poses from a single image and a NeRF model of the scene. This approach eliminates the need for explicit feature matching or iterative pose refinement, which are computationally expensive and often sensitive to initialization. Our experimental results indicate that IFFNeRF achieves high accuracy without requiring initialization, demonstrating robustness while being computationally efficient across diverse environments. The performance gains stem from the innovative ray generation strategy, which efficiently samples a broad range of camera pose hypotheses, and the attention mechanism, which effectively maps rays to image pixel features. Through extensive validation, our method exhibits improved

robustness across both synthetic and real-world datasets and has the potential for real-time applications in robotics and beyond. Extending our research in pose estimation, we developed 6DGS, which builds upon the insights gained from IFFNeRF while leveraging the efficiency and fidelity of 3D Gaussian Splatting representations. 6DGS introduces a unified approach that simultaneously addresses object-centric 6D pose estimation and camera-centric visual relocalization within a single computational framework. The ability to quickly and accurately determine camera poses relative to objects of interest represents a critical capability for robotic systems operating in dynamic environments.

Finally, based on the experience gained with 6DoF pose estimation, we developed a novel dexterous grasping method, GRASPLAT, which overcomes the limitations of existing approaches by leveraging 3D information while being trained solely on RGB images. GRASPLAT introduces a paradigm for predicting successful grasps by synthesizing physically plausible hand-object images, enabling the regression of hand joint configurations necessary for successful grasping. This approach bridges the gap between traditional RGB-based methods, which often lack geometric understanding, and 3D-based methods, which require expensive depth sensing or 3D reconstruction. By utilizing 3D Gaussian Splatting for high-fidelity scene representation, our approach ensures 3D consistency while maintaining the simplicity of RGB-based learning, eliminating the need for explicit 3D scans or dense point clouds. The method’s differentiable rendering pipeline allows for end-to-end training and optimization of grasp configurations directly from image data. Experimental results on both synthetic and real-world grasping datasets validate the effectiveness of GRASPLAT, showcasing significantly improved grasp success rates compared to baseline methods, particularly for objects with complex geometries or challenging surface properties. GRASPLAT offers a scalable and efficient solution for large-scale dexterous robotic grasping, paving the way for future advancements in robotic manipulation in unstructured environments.

In conclusion, this dissertation presents a cohesive body of work that advances the field of robotic perception and manipulation through novel applications of view synthesis techniques. The four primary contributions—VM-NeRF for sparse novel view synthesis, IFFNeRF and 6DGS for robust 6DoF pose estimation, and GRASPLAT for dexterous grasping—form a progressive research pathway that addresses fundamental challenges in robot vision and manipulation. These innovations collectively enable robots to better understand and interact with their environments, particularly when dealing with objects that have traditionally posed difficulties for computer vision systems. The methods developed in this work are characterized by their computational efficiency, robustness to challenging conditions, and ability to generalize across different scenarios, making them well-suited for real-world robotic applications.

## 6.1 Future Work

The research presented in this dissertation opens several promising directions for future investigation, addressing the current limitations of our methods and exploring new applications of NVS in robotics. Sparse novel view synthesis methods, such as VM-NeRF, still rely on accurate camera poses, which constrains their practical application. Future

work should improve VM-NeRF’s robustness when camera poses are unavailable or imprecise. We must also address scenarios with fewer cameras than the four we tested, such as the two cameras available in the DROID dataset [275]. For pose estimation methods (IFFNeRF and 6DGS), future research should increase precision of estimated camera poses, reducing or eliminating quantization artifacts from ray sampling. Work should also explore rendering capabilities of underlying novel view synthesis methods to verify estimation correctness in a manner analogous to human perception. Integration into active perception frameworks—where robots strategically select viewpoints—represents another compelling direction, potentially optimizing information gain during manipulation tasks. GRASPLAT research should incorporate physics-based simulations within the training pipeline to model contact dynamics and material properties more realistically. While currently applied only to grasping, our methodology could extend to general manipulation training once challenges with comprehensive training datasets are resolved. Extensions to diverse manipulation primitives beyond grasping represent a natural evolution of this research. A unifying direction is developing end-to-end systems integrating scene reconstruction, pose estimation, and grasp planning for deployment on real-world robotic platforms.

The advancements presented contribute to developing more capable autonomous robotic systems by bridging the gap between geometric understanding and functional interaction. By advancing these technologies, we approach a future where robots seamlessly perceive and interact with environments, enabling greater autonomy in both industrial and domestic settings.



# Bibliography

- [1] Ali Karami, Roberto Battisti, Fabio Menna, and Fabio Remondino. 3D digitization of transparent and glass surfaces: State of the art and analysis of some methods. *The International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, 2022.
- [2] 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.
- [3] Arno Knapitsch, Jaesik Park, Qian-Yi Zhou, and Vladlen Koltun. Tanks and temples: Benchmarking large-scale scene reconstruction. *ACM Transactions on Graphics*, 36(4), 2017.
- [4] Jonathan T Barron, Ben Mildenhall, Matthew Tancik, Peter Hedman, Ricardo Martin-Brualla, and Pratul P Srinivasan. Mip-NeRF: A multiscale representation for anti-aliasing neural radiance fields. In *ICCV*, 2021.
- [5] Jonathan T. Barron, Ben Mildenhall, Dor Verbin, Pratul P. Srinivasan, and Peter Hedman. Mip-nerf 360: Unbounded anti-aliased neural radiance fields. In *CVPR*, 2022.
- [6] Steven M Seitz and Charles R Dyer. View morphing. In *Conference on Computer graphics and interactive techniques*, 1996.
- [7] Matthias Zwicker, Hanspeter Pfister, Jeroen Van Baar, and Markus Gross. Ewa splatting. *TVCG*, 8:223–238, 2002.
- [8] Shadow Robot Company. Shadow dexterous hand. <https://www.shadowrobot.com/dexterous-hand-series/>. Accessed: 2025-02-28.
- [9] Andrea Fusiello, Emanuele Trucco, and Alessandro Verri. A compact algorithm for rectification of stereo pairs. *Machine Vision and Applications*, 12(1):16–22, July 2000.
- [10] Tianlong Chen, Peihao Wang, Zhiwen Fan, and Zhangyang Wang. Aug-NeRF: Training Stronger Neural Radiance Fields with Triple-Level Physically-Grounded Augmentations. In *CVPR*, 2022.

- [11] Ajay Jain, Matthew Tancik, and Pieter Abbeel. Putting NeRF on a Diet: Semantically Consistent Few-Shot View Synthesis. In *ICCV*, 2021.
- [12] Michael Niemeyer, Jonathan T. Barron, Ben Mildenhall, Mehdi S. M. Sajjadi, Andreas Geiger, and Noha Radwan. RegNeRF: Regularizing Neural Radiance Fields for View Synthesis from Sparse Inputs. In *CVPR*, 2022.
- [13] Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, et al. Llama: Open and efficient foundation language models, 2024.
- [14] Francisco Bonin-Font, Alberto Ortiz, and Gabriel Oliver. Visual navigation for mobile robots: A survey. *Journal of Intelligent and Robotic Systems*, 53, 2008.
- [15] Siting Zhu, Guangming Wang, Xin Kong, Dezhi Kong, and Hesheng Wang. 3d gaussian splatting in robotics: A survey, 2024.
- [16] Guangming Wang, Lei Pan, Songyou Peng, Shaohui Liu, Chenfeng Xu, Yanzi Miao, Wei Zhan, Masayoshi Tomizuka, Marc Pollefeys, and Hesheng Wang. Nerf in robotics: A survey, 2024.
- [17] Yuchuang Tong, Haotian Liu, and Zhengtao Zhang. Advancements in humanoid robots: A comprehensive review and future prospects. *IEEE/CAA Journal of Automatica Sinica*, 11(2):301–328, 2024.
- [18] G.J. Monkman, S. Hesse, R. Steinmann, and H. Schunk. *Robot Grippers*. Wiley, 2007.
- [19] Oliver Heimann and Jan Guhl. Industrial robot programming methods: A scoping review. In *IEEE ETFA*, 2020.
- [20] Giulio Rosati, Giovanni Boschetti, A Biondi, and Aldo Rossi. On-line dimensional measurement of small components on the eyeglasses assembly line. *Optics and Lasers in Engineering*, 47(3-4):320–328, 2009.
- [21] Jessica G Ramsey and Loren Tapp. *Ergonomic Evaluation of Surfacing and Finishing Tasks During Eyeglass Manufacturing-Minnesota*. Citeseer, 2012.
- [22] Hiroyuki Tada. Manufacturing technologies for titanium eyeglass frames. *Titanium for Consumer Applications*, 1(13):235–268, 2019.
- [23] Federica Murmura, Laura Bravi, and Gilberto Santos. Sustainable process and product innovation in the eyewear sector: The role of industry 4.0 enabling technologies. *Sustainability*, 13(1), 2021.
- [24] Aurelio Montalto, Serena Graziosi, Monica Bordegoni, Luca Di Landro, and Michael Johannes Leonardus van Tooren. An approach to design reconfigurable manufacturing tools to manage product variability: the mass customisation of eyewear. *Journal of Intelligent Manufacturing*, 31(1):87–102, 2020.

- 
- [25] Jim J Browne, Didier Dubois, Keith Rathmill, Suresh Sethi, and Kathrin Stecke. Classification of flexible manufacturing systems. *The FMS magazine*, 2(2):114–117, 1984.
  - [26] Longbing Cao. Ai robots and humanoid ai: Review, perspectives and directions, 2024.
  - [27] Yayu Huang, Dongxuan Fan, Haonan Duan, Dashun Yan, Wen Qi, Jia Sun, Qian Liu, and Peng Wang. Human-like dexterous manipulation for anthropomorphic five-fingered hands: A review. *Biomimetic Intelligence and Robotics*, 5(1), 2025.
  - [28] Cassandra A. Adduri and Jonathan J. Marotta. Mental Rotation of Faces in Healthy Aging and Alzheimer's Disease. *PLoS ONE*, 4(7):e6120, July 2009.
  - [29] Antonio Bicchi and Vijay Kumar. Robotic grasping and contact: A review. In *ICRA*, 2000.
  - [30] Zhen Xie, Xinquan Liang, and Canale Roberto. Learning-based robotic grasping: A review. *Frontiers in Robotics and AI*, 10, 2023.
  - [31] Julie Bort. How Chef Robotics found success by turning away its original customers. <https://techcrunch.com/2025/04/10/how-chef-robotics-found-success-by-turning-away-its-original-customers/>. Accessed: 2025-04-15.
  - [32] B Wang, L Jiang, JW Li, HG Cai, and H Liu. Grasping unknown objects based on 3d model reconstruction. In *IEEE/ASME ICAIM*, 2005.
  - [33] Yingzhao Zhu, Man Li, Wensheng Yao, and Chunhua Chen. A review of 6D object pose estimation. In *ITAIC*, 2022.
  - [34] Yasutaka Furukawa and Jean Ponce. Accurate, dense, and robust multiview stereopsis. *TPAMI*, 32(8):1362–1376, 2010.
  - [35] 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 *CoRL*, 2022.
  - [36] Johannes Lutz Schönberger and Jan-Michael Frahm. Structure-from-Motion Revisited. In *CVPR*, 2016.
  - [37] Xian-Feng Han, Hamid Laga, and Mohammed Bennamoun. Image-based 3D object reconstruction: State-of-the-art and trends in the deep learning era. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 43(5):1578–1604, November 2019.
  - [38] Zhaoshuo Li, Thomas Müller, Alex Evans, Russell H Taylor, Mathias Unberath, Ming-Yu Liu, and Chen-Hsuan Lin. Neuralangelo: High-fidelity neural surface reconstruction. In *CVPR*, 2023.

- [39] Binbin Huang, Zehao Yu, Anpei Chen, Andreas Geiger, and Shenghua Gao. 2d gaussian splatting for geometrically accurate radiance fields. In *SIGGRAPH*, 2024.
- [40] Jeffrey Ichnowski, Yahav Avigal, Justin Kerr, and Ken Goldberg. Dex-NeRF: Using a Neural Radiance Field to Grasp Transparent Objects. In *CoRL*, 2022.
- [41] Angela Sirigu and Jean-René Duhamel. Motor and visual imagery as two complementary but neurally dissociable mental processes. *Journal of cognitive neuroscience*, 13(7):910–919, 2001.
- [42] Jian Guan, Yingming Hao, Qingxiao Wu, Sicong Li, and Yingjian Fang. A survey of 6dof object pose estimation methods for different application scenarios. *Sensors*, 24(4):1076, 2024.
- [43] Haonan Duan, Peng Wang, Yayu Huang, Guangyun Xu, Wei Wei, and Xiaofei Shen. Robotics dexterous grasping: The methods based on point cloud and deep learning. *Frontiers in Neurorobotics*, 15:210–271, 2021.
- [44] Pinch grasping robot. <https://www.amazon.science/latest-news/pinch-grasping-robot-handles-items-with-precision>. Accessed: 2025-03-28.
- [45] N.A. Lad, Y.P. Ballal, and P.D. Kulkarni. Study of end effectors - a review. *International Journal of Trend in Research and Development*, 2(5):365–368, 2015.
- [46] Francisco Oliveira, Vítor Tinoco, Sandro Magalhães, Filipe N Santos, and Manuel F Silva. End-effectors for harvesting manipulators-state of the art review. In *ICARSC*, pages 98–103, 2022.
- [47] Dedong Tang, Fengxiao Li, Jingang Jiang, Shichang Song, and Yang Zeng. A review on end-effectors of robotic grinding. *Recent Patents on Engineering*, 17(1):85–101, 2023.
- [48] Francesco Cepolina and Roberto Razzoli. Review of robotic surgery platforms and end effectors. *Journal of Robotic Surgery*, 18(1):74, 2024.
- [49] Enric Corona, Albert Pumarola, Guillem Alenya, Francesc Moreno-Noguer, and Grégory Rogez. Ganhand: Predicting human grasp affordances in multi-object scenes. In *CVPR*, 2020.
- [50] Geoffrey E Hinton, Simon Osindero, and Yee-Whye Teh. A fast learning algorithm for deep belief nets. *Neural computation*, 18(7):1527–1554, 2006.
- [51] Pin Wang, En Fan, and Peng Wang. Comparative analysis of image classification algorithms based on traditional machine learning and deep learning. *Pattern recognition letters*, 141:61–67, 2021.



- 
- [52] Shaveta Dargan, Munish Kumar, Maruthi Rohit Ayyagari, and Gulshan Kumar. A survey of deep learning and its applications: a new paradigm to machine learning. *Archives of computational methods in engineering*, 27:1071–1092, 2020.
  - [53] Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.
  - [54] Mohsen Soori, Behrooz Arezoo, and Roza Dastres. Artificial intelligence, machine learning and deep learning in advanced robotics, a review. *Cognitive Robotics*, 3:54–70, 2023.
  - [55] Daniel W Otter, Julian R Medina, and Jugal K Kalita. A survey of the usages of deep learning for natural language processing. *IEEE transactions on neural networks and learning systems*, 32(2):604–624, 2020.
  - [56] Bharat Singh, Rajesh Kumar, and Vinay Pratap Singh. Reinforcement learning in robotic applications: a comprehensive survey. *Artificial Intelligence Review*, 55(2):945–990, 2022.
  - [57] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *NeurIPS*, 2017.
  - [58] Alexey Dosovitskiy et al. An image is worth 16x16 words: Transformers for image recognition at scale. In *ICLR*, 2021.
  - [59] Mathilde Caron, Hugo Touvron, Ishan Misra, Hervé Jégou, Julien Mairal, Piotr Bojanowski, and Armand Joulin. Emerging properties in self-supervised vision transformers. In *ICCV*, 2021.
  - [60] Maxime Oquab et al. DINOv2: Learning Robust Visual Features without Supervision. *TMLR*, 2023.
  - [61] Charles R Qi, Hao Su, Kaichun Mo, and Leonidas J Guibas. Pointnet: Deep learning on point sets for 3d classification and segmentation. In *CVPR*, pages 652–660, 2017.
  - [62] Charles Ruizhongtai Qi, Li Yi, Hao Su, and Leonidas J Guibas. Pointnet++: Deep hierarchical feature learning on point sets in a metric space. In *NeurIPS*, 2017.
  - [63] Vincent Sitzmann, Julien N.P. Martel, Alexander W. Bergman, David B. Lindell, and Gordon Wetzstein. Implicit neural representations with periodic activation functions. In *NeurIPS*, 2020.
  - [64] Orazio Gallo, Alejandro Troccoli, and Varun Jampani. Novel View Synthesis: From Depth-Based Warping to Multi-Plane Images and Beyond. *CVPR Tutorial*, 2020.

- [65] A. Tewari and et al. Advances in Neural Rendering. *Computer Graphics Forum*, 41(2):703–735, May 2022.
- [66] Anurag Dalal, Daniel Hagen, Kjell G Robbersmyr, and Kristian Muri Knausgård. Gaussian splatting: 3d reconstruction and novel view synthesis, a review. *IEEE Access*, 12:96797–96820, 2024.
- [67] Shimon Ullman. The interpretation of structure from motion. *Proceedings of the Royal Society of London. Series B. Biological Sciences*, 203(1153):405–426, January 1979.
- [68] Yasutaka Furukawa and Carlos Hernández. Multi-view stereo: A tutorial. *Foundations and Trends in Computer Graphics and Vision*, 9(1-2):1–148, 2015.
- [69] Reinhard Koch, Marc Pollefeys, and Luc Van Gool. Multi viewpoint stereo from uncalibrated video sequences. In *ECCV*, 1998.
- [70] Gernot Riegler and Vladlen Koltun. Free view synthesis. In *ECCV*, 2020.
- [71] Neill DF Campbell, George Vogiatzis, Carlos Hernández, and Roberto Cipolla. Using multiple hypotheses to improve depth-maps for multi-view stereo. In *ECCV*, 2008.
- [72] Thomas Schops, Johannes L. Schonberger, Silvano Galliani, Torsten Sattler, Konrad Schindler, Marc Pollefeys, and Andreas Geiger. A multi-view stereo benchmark with high-resolution images and multi-camera videos. In *CVPR*, 2017.
- [73] Tinghui Zhou, Matthew Brown, Noah Snavely, and David G Lowe. Unsupervised learning of depth and ego-motion from video. In *CVPR*, 2017.
- [74] Jonáš Kulháněk, Erik Derner, Torsten Sattler, and Robert Babuška. ViewFormer: NeRF-free Neural Rendering from Few Images Using Transformers. In *ECCV*, 2022.
- [75] Stephen Lombardi, Tomas Simon, Jason Saragih, Gabriel Schwartz, Andreas Lehrmann, and Yaser Sheikh. Neural volumes: Learning dynamic renderable volumes from images. In *SIGGRAPH*, 2019.
- [76] Thomas Müller, Alex Evans, Christoph Schied, and Alexander Keller. Instant neural graphics primitives with a multiresolution hash encoding. *ACM Trans. Graph.*, 41(4), 2022.
- [77] Anpei Chen, Zexiang Xu, Andreas Geiger, Jingyi Yu, and Hao Su. Tensorf: Tensorial radiance fields. In *ECCV*, 2022.
- [78] Bernhard Kerbl, Georgios Kopanas, Thomas Leimkühler, and George Drettakis. 3d gaussian splatting for real-time radiance field rendering. *TOG*, 2023.

- 
- [79] International Federation of Robotics. World robotics 2024 report. Technical report, International Federation of Robotics, 2024.
  - [80] Chao Cheng, Jun Fu, Hang Su, and Luquan Ren. Recent advancements in agriculture robots: Benefits and challenges. *Machines*, 11(1):48, 2023.
  - [81] Dhanya Pramod. Robotic process automation for industry: adoption status, benefits, challenges and research agenda. *Benchmarking: an international journal*, 29(5):1562–1586, 2022.
  - [82] Richard M Murray, Zexiang Li, and S Shankar Sastry. *A mathematical introduction to robotic manipulation*. CRC press, 2017.
  - [83] Charles Chikwendu Okpala, Nkemakonam Chidiebube Igbokwe, and Constance Obiuto Nwankwo. The evolution and impact of jigs and fixtures in modern manufacturing. *International Journal of Engineering Inventions*, 13(7):125–131, 2024.
  - [84] Gayane Kazhoyan, Simon Stelter, Franklin Kenghagho Kenfack, Sebastian Koralewski, and Michael Beetz. The robot household marathon experiment. In *ICRA*, 2021.
  - [85] Matteo Bortolon, Alessio Del Bue, and Fabio Poiesi. VM-NeRF: Tackling Sparsity in NeRF with View Morphing. In *ICIAP*, 2023.
  - [86] Matteo Bortolon, Theodore Tsesmelis, Stuart James, Fabio Poiesi, and Alessio Del Bue. Iffnerf: Initialization free and fast 6dof pose estimation from a single image and a nerf model. In *ICRA*, 2024.
  - [87] Matteo Bortolon, Theodore Tsesmelis, Stuart James, Fabio Poiesi, and Alessio Del Bue. 6dgs: 6d pose estimation from a single image and a 3d gaussian splatting model. In *ECCV*, 2024.
  - [88] Bortolon Matteo, Duarte Nuno Ferreira, Moreno Plinio, Poiesi Fabio, Santos-Victor José, and Del Bue Alessio. Grasplat: Enabling dexterous grasping through novel view synthesis. In *IROS*, 2025.
  - [89] Heiko Hirschmuller. Stereo processing by semiglobal matching and mutual information. *TPAMI*, 30(2):328–341, 2008.
  - [90] Steven M Seitz and Charles R Dyer. Photorealistic scene reconstruction by voxel coloring. *IJCV*, 35:151–173, 1999.
  - [91] Shahram Izadi, David Kim, Otmar Hilliges, David Molyneaux, Richard Newcombe, Pushmeet Kohli, Jamie Shotton, Steve Hodges, Dustin Freeman, Andrew Davison, and Andrew Fitzgibbon. Kinectfusion: real-time 3d reconstruction and interaction using a moving depth camera. In *ACM Symposium on User Interface Software and Technology*, 2011.

- [92] M. Lhuillier and L. Quan. A quasi-dense approach to surface reconstruction from uncalibrated images. *TPAMI*, 27(3):418–433, 2005.
- [93] Ruigang Yang and Marc Pollefeys. Multi-resolution real-time stereo on commodity graphics hardware. In *CVPR*, 2003.
- [94] Silvano Galliani, Katrin Lasinger, and Konrad Schindler. Massively parallel multiview stereopsis by surface normal diffusion. In *ICCV*, 2015.
- [95] Qingshan Xu and Wenbing Tao. Multi-scale geometric consistency guided multi-view stereo. In *CVPR*, 2019.
- [96] Kaixuan Wang and Shaojie Shen. MVDepthNet: Real-time multiview depth estimation neural network. In *3DV*, 2018.
- [97] Arda Duzceker, Silvano Galliani, Christoph Vogel, Pablo Speciale, Mihai Dumanu, and Marc Pollefeys. Deepvideomvs: Multi-view stereo on video with recurrent spatio-temporal fusion. In *CVPR*, 2021.
- [98] Yao Yao, Zixin Luo, Shiwei Li, Tian Fang, and Long Quan. MVSNet: Depth Inference for Unstructured Multi-view Stereo. In *ECCV*, 2018.
- [99] Fangjinhua Wang, Silvano Galliani, Christoph Vogel, Pablo Speciale, and Marc Pollefeys. Patchmatchnet: Learned multi-view patchmatch stereo. In *CVPR*, 2021.
- [100] Hongbin Xu, Zhipeng Zhou, Yu Qiao, Wenxiong Kang, and Qiuxia Wu. Self-supervised multi-view stereo via effective co-segmentation and data-augmentation. In *AAAI*, 2021.
- [101] Marc Levoy and Pat Hanrahan. Light field rendering. In *SIGGRAPH*, 1996.
- [102] Steven J. Gortler, Radek Grzeszczuk, Richard Szeliski, and Michael F. Cohen. The lumigraph. In *SIGGRAPH*, 1996.
- [103] Chris Buehler, Michael Bosse, Leonard McMillan, Steven Gortler, and Michael Cohen. Unstructured lumigraph rendering. In *CGIT*, 2001.
- [104] James R Bergen and Edward H Adelson. The plenoptic function and the elements of early vision. *Computational models of visual processing*, 1(8):3, 1991.
- [105] Ben Mildenhall, Pratul P. Srinivasan, Rodrigo Ortiz-Cayon, Nima Khademi Kalantari, Ravi Ramamoorthi, Ren Ng, and Abhishek Kar. Local Light Field Fusion: Practical View Synthesis with Prescriptive Sampling Guidelines. *ACM Transactions on Graphics*, 38(29):1–14, August 2019.
- [106] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *CVPR*, 2009.

- 
- [107] John Flynn, Ivan Neulander, James Philbin, and Noah Snavely. Deepstereo: Learning to predict new views from the world’s imagery. In *CVPR*, 2016.
  - [108] Tinghui Zhou, Shubham Tulsiani, Weilun Sun, Jitendra Malik, and Alexei A Efros. View synthesis by appearance flow. In *ECCV*, 2016.
  - [109] Maxim Tatarchenko, Alexey Dosovitskiy, and Thomas Brox. Multi-view 3d models from single images with a convolutional network. In *ECCV*, 2016.
  - [110] Eunbyung Park, Jimei Yang, Ersin Yumer, Duygu Ceylan, and Alexander C Berg. Transformation-grounded image generation network for novel 3d view synthesis. In *CVPR*, 2017.
  - [111] Vincent Sitzmann, Justus Thies, Felix Heide, Matthias Nießner, Gordon Wetzstein, and Michael Zollhöfer. Deepvoxels: Learning persistent 3d feature embeddings. In *CVPR*, 2019.
  - [112] Jeong Joon Park, Peter Florence, Julian Straub, Richard Newcombe, and Steven Lovegrove. DeepSDF: Learning continuous signed distance functions for shape representation. In *CVPR*, 2019.
  - [113] Lars Mescheder, Michael Oechsle, Michael Niemeyer, Sebastian Nowozin, and Andreas Geiger. Occupancy networks: Learning 3D reconstruction in function space. In *CVPR*, 2019.
  - [114] C.E. Shannon. Communication in the presence of noise. *Proceedings of the IRE*, 37(1):10–21, 1949.
  - [115] Dor Verbin, Peter Hedman, Ben Mildenhall, Todd Zickler, Jonathan T Barron, and Pratul P Srinivasan. Ref-nerf: Structured view-dependent appearance for neural radiance fields. In *CVPR*, 2022.
  - [116] Kangle Deng, Andrew Liu, Jun-Yan Zhu, and Deva Ramanan. Depth-supervised NeRF: Fewer views and faster training for free. In *CVPR*, 2022.
  - [117] Yi Wei, Shaohui Liu, Yongming Rao, Wang Zhao, Jiwen Lu, and Jie Zhou. Nerfin-gmvs: Guided optimization of neural radiance fields for indoor multi-view stereo. In *ICCV*, 2021.
  - [118] Qiangeng Xu, Zexiang Xu, Julien Philip, Sai Bi, Zhixin Shu, Kalyan Sunkavalli, and Ulrich Neumann. Point-nerf: Point-based neural radiance fields. In *CVPR*, 2022.
  - [119] Alex Yu, Ruilong Li, Matthew Tancik, Hao Li, Ren Ng, and Angjoo Kanazawa. PlenOctrees for real-time rendering of neural radiance fields. In *ICCV*, 2021.
  - [120] Zhiqin Chen, Thomas Funkhouser, Peter Hedman, and Andrea Tagliasacchi. Mobilenerf: Exploiting the polygon rasterization pipeline for efficient neural field rendering on mobile architectures. In *CVPR*, 2023.

- [121] Tomas Akenine-Mo, Eric Haines, Naty Hoffman, et al. *Real-time rendering*. AK Peters/CRC Press, 2018.
- [122] Joo Chan Lee, Daniel Rho, Xiangyu Sun, Jong Hwan Ko, and Eunbyung Park. Compact 3D gaussian representation for radiance field. In *CVPR*, 2024.
- [123] Yihang Chen, Qianyi Wu, Weiyao Lin, Mehrtash Harandi, and Jianfei Cai. Hac: Hash-grid assisted context for 3d gaussian splatting compression. In *ECCV*, 2024.
- [124] Simon Niedermayr, Josef Stumpfegger, and Rüdiger Westermann. Compressed 3D Gaussian Splatting for accelerated novel view synthesis. In *CVPR*, 2024.
- [125] Wieland Morgenstern, Florian Barthel, Anna Hilsmann, and Peter Eisert. Compact 3d scene representation via self-organizing gaussian grids. In *ECCV*, 2024.
- [126] Shakiba Kheradmand, Daniel Rebain, Gopal Sharma, Weiwei Sun, Yang-Che Tseng, Hossam Isack, Abhishek Kar, Andrea Tagliasacchi, and Kwang Moo Yi. 3d gaussian splatting as markov chain monte carlo. In *NeurIPS*, 2024.
- [127] Samuel Rota Bulò, Lorenzo Porzi, and Peter Kotschieder. Revising densification in gaussian splatting. In *ECCV*, 2024.
- [128] Alexander Mai, Peter Hedman, George Kopanas, Dor Verbin, David Futschik, Qiangeng Xu, Falko Kuester, Jonathan T Barron, and Yinda Zhang. EVER: Exact volumetric ellipsoid rendering for real-time view synthesis, 2024.
- [129] Lukas Radl, Michael Steiner, Mathias Parger, Alexander Weinrauch, Bernhard Kerbl, and Markus Steinberger. StopThePop: Sorted Gaussian Splatting for View-Consistent Real-time Rendering. *ACM Transactions on Graphics*, 43(4):1–17, 2024.
- [130] Nicolas Moenne-Loccoz, Ashkan Mirzaei, Or Perel, Riccardo de Lutio, Janick Martinez Esturo, Gavriel State, Sanja Fidler, Nicholas Sharp, and Zan Gojcic. 3d gaussian ray tracing: Fast tracing of particle scenes. *ACM Transactions on Graphics and SIGGRAPH Asia*, 43(6):1–19, 2024.
- [131] Qi Wu, Janick Martinez Esturo, Ashkan Mirzaei, Nicolas Moenne-Loccoz, and Zan Gojcic. 3dgut: Enabling distorted cameras and secondary rays in gaussian splatting. In *CVPR*, 2025.
- [132] Henrik Aanæs, Rasmus Ramsbøl Jensen, George Vogiatzis, Engin Tola, and Anders Bjarholm Dahl. Large-Scale Data for Multiple-View Stereopsis. *International Journal of Computer Vision*, 120(2):153–168, November 2016.
- [133] J. Wang, C. Lan, C. Liu, Y. Ouyang, T. Qin, W. Lu, Y. Chen, W. Zeng, and P. S. Yu. Generalizing to Unseen Domains: A Survey on Domain Generalization. *IEEE Transactions on Knowledge & Data Engineering*, 35(08):8052–8072, August 2023.

- 
- [134] Thomas Müller, Fabrice Rousselle, Jan Novák, and Alexander Keller. Real-time neural radiance caching for path tracing. *ACM Transactions on Graphics*, 40(4):1–16, August 2021.
  - [135] Alex Yu, Vickie Ye, Matthew Tancik, and Angjoo Kanazawa. PixelNeRF: Neural radiance fields from one or few images. In *CVPR*, 2021.
  - [136] Alex Trevithick and Bo Yang. Grf: Learning a general radiance field for 3d scene representation and rendering. In *ICCV*, 2020.
  - [137] Anpei Chen, Zexiang Xu, Fuqiang Zhao, Xiaoshuai Zhang, Fanbo Xiang, Jingyi Yu, and Hao Su. MVSNeRF: Fast Generalizable Radiance Field Reconstruction from Multi-View Stereo. In *CVPR*, 2021.
  - [138] Yuan Liu, Sida Peng, Lingjie Liu, Qianqian Wang, Peng Wang, Theobalt Christian, Xiaowei Zhou, and Wenping Wang. Neural rays for occlusion-aware image-based rendering. In *CVPR*, 2022.
  - [139] Johannes L Schönberger, Enliang Zheng, Jan-Michael Frahm, and Marc Pollefeys. Pixelwise view selection for unstructured multi-view stereo. In *ECCV*, 2016.
  - [140] Mohammad Mahdi Johari, Yann Lepoittevin, and François Fleuret. GeoNeRF: Generalizing nerf with geometry priors. In *CVPR*, 2022.
  - [141] Xiaoshuai Zhang, Sai Bi, Kalyan Sunkavalli, Hao Su, and Zexiang Xu. Nerfusion: Fusing radiance fields for large-scale scene reconstruction. In *CVPR*, 2022.
  - [142] 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. In *NeurIPS*, 2021.
  - [143] Xiaoxiao Long, Cheng Lin, Peng Wang, Taku Komura, and Wenping Wang. SparseNeuS: Fast Generalizable Neural Surface Reconstruction from Sparse views. In *ECCV*, 2022.
  - [144] Konstantinos Rematas, Ricardo Martin-Brualla, and Vittorio Ferrari. ShaRF: Shape-conditioned Radiance Fields from a Single View. In *ICML*, 2021.
  - [145] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, Gretchen Krueger, and Ilya Sutskever. Learning Transferable Visual Models From Natural Language Supervision. In *ICML*, 2021.
  - [146] Laurent Dinh, Jascha Sohl-Dickstein, and Samy Bengio. Density estimation using real nvp. In *ICLR*, 2017.
  - [147] Chen Sun, Abhinav Shrivastava, Saurabh Singh, and Abhinav Gupta. Revisiting unreasonable effectiveness of data in deep learning era. In *ICCV*, pages 843–852, 2017.

- [148] Jiawei Yang, Marco Pavone, and Yue Wang. FreeNeRF: Improving few-shot neural rendering with free frequency regularization. In *CVPR*, 2023.
- [149] Ruoxi Shi, Xinyue Wei, Cheng Wang, and Hao Su. ZeroRF: Fast sparse view 360deg reconstruction with zero pretraining. In *CVPR*, pages 21114–21124, 2024.
- [150] Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. High-resolution image synthesis with latent diffusion models. In *CVPR*, 2022.
- [151] Connor Shorten and Taghi M. Khoshgoftaar. A survey on Image Data Augmentation for Deep Learning. *Journal of Big Data*, 6(1), July 2019.
- [152] Di Chen, Yu Liu, Lianghua Huang, Bin Wang, and Pan Pan. GeoAug: Data Augmentation for Few-Shot NeRF with Geometry Constraints. In *ECCV*, 2022.
- [153] David Charatan, Sizhe Lester Li, Andrea Tagliasacchi, and Vincent Sitzmann. pixelsplat: 3d gaussian splats from image pairs for scalable generalizable 3d reconstruction. In *CVPR*, 2024.
- [154] Yuedong Chen, Haoifei Xu, Chuanxia Zheng, Bohan Zhuang, Marc Pollefeys, Andreas Geiger, Tat-Jen Cham, and Jianfei Cai. Mvsplat: Efficient 3d gaussian splatting from sparse multi-view images. In *ECCV*, 2024.
- [155] Haolin Xiong, Sairisheek Muttukuru, Rishi Upadhyay, Pradyumna Chari, and Achuta Kadambi. SparseSegs: Real-time 360° sparse view synthesis using gaussian splatting, 2024.
- [156] Chen Yang, Sikuang Li, Jiemin Fang, Ruofan Liang, Lingxi Xie, Xiaopeng Zhang, Wei Shen, and Qi Tian. Gaussianobject: High-quality 3d object reconstruction from four views with gaussian splatting. *ACM Transactions on Graphics*, 43(6):1 – 13, 2024.
- [157] Lvmin Zhang, Anyi Rao, and Maneesh Agrawala. Adding conditional control to text-to-image diffusion models. In *ICCV*, 2023.
- [158] Avinash Paliwal, Wei Ye, Jinhui Xiong, Dmytro Kotovenko, Rakesh Ranjan, Vikas Chandra, and Nima Khademi Kalantari. Coherentgs: Sparse novel view synthesis with coherent 3d gaussians. In *ECCV*, 2024.
- [159] Lihe Yang, Bingyi Kang, Zilong Huang, Xiaogang Xu, Jiashi Feng, and Hengshuang Zhao. Depth anything: Unleashing the power of large-scale unlabeled data. In *CVPR*, 2024.
- [160] Jiahe Li, Jiawei Zhang, Xiao Bai, Jin Zheng, Xin Ning, Jun Zhou, and Lin Gu. Dngaussian: Optimizing sparse-view 3d gaussian radiance fields with global-local depth normalization. In *CVPR*, 2024.



- 
- [161] Zehao Zhu, Zhiwen Fan, Yifan Jiang, and Zhangyang Wang. Fsgs: Real-time few-shot view synthesis using gaussian splatting. In *ECCV*, 2024.
- [162] Zhiwen Fan, Wenyan Cong, Kairun Wen, Kevin Wang, Jian Zhang, Xinghao Ding, Danfei Xu, Boris Ivanovic, Marco Pavone, Georgios Pavlakos, Zhangyang Wang, and Yue Wang. Instantsplat: Unbounded sparse-view pose-free gaussian splatting in 40 seconds. In *CoRR*, 2024.
- [163] Zirui Wang, Shangzhe Wu, Weidi Xie, Min Chen, and Victor Adrian Prisacariu. NeRF—: Neural radiance fields without known camera parameters, 2021.
- [164] Chen-Hsuan Lin, Wei-Chiu Ma, Antonio Torralba, and Simon Lucey. Barf: Bundle-adjusting neural radiance fields. In *ICCV*, 2021.
- [165] Shin-Fang Chng, Sameera Ramasinghe, Jamie Sherrah, and Simon Lucey. Gaussian activated neural radiance fields for high fidelity reconstruction and pose estimation. In *ECCV*, 2022.
- [166] Yoonwoo Jeong, Seokjun Ahn, Christopher Bongsoo Choy, Anima Anandkumar, Minsu Cho, and Jaesik Park. Self-calibrating neural radiance fields. In *ICCV*, 2021.
- [167] Fu Li, Shishir Reddy Vutukur, Hao Yu, Ivan Shugurov, Benjamin Busam, Shaowu Yang, and Slobodan Ilic. Nerf-pose: A first-reconstruct-then-regress approach for weakly-supervised 6d object pose estimation. In *ICCV*, 2023.
- [168] 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 *CVPR*, 2019.
- [169] Yunzhi Lin, Thomas Müller, Jonathan Tremblay, Bowen Wen, Stephen Tyree, Alex Evans, Patricio A Vela, and Stan Birchfield. Parallel inversion of neural radiance fields for robust pose estimation. In *ICRA*, 2023.
- [170] Dominic Maggio, Marcus Abate, Jingnan Shi, Courtney Mario, and Luca Carlone. Loc-nerf: Monte carlo localization using neural radiance fields. In *ICRA*, 2023.
- [171] Arthur Moreau, Nathan Piasco, Moussab Bennehar, Dzmitry Tsishkou, Bogdan Stanciulescu, and Arnaud de La Fortelle. Crossfire: Camera relocalization on self-supervised features from an implicit representation. In *ICCV*, 2023.
- [172] Angtian Wang, Peng Wang, Jian Sun, Adam Kortylewski, and Alan Yuille. Voge: a differentiable volume renderer using gaussian ellipsoids for analysis-by-synthesis. In *ICLR*, 2022.
- [173] Angtian Wang, Adam Kortylewski, and Alan Yuille. Nemo: Neural mesh models of contrastive features for robust 3d pose estimation. In *ICLR*, 2020.

- [174] Paul Gay, Cosimo Rubino, Vaibhav Bansal, and Alessio Del Bue. Probabilistic structure from motion with objects (psfmo). In *ICCV*, 2017.
- [175] Marco Crocco, Cosimo Rubino, and Alessio Del Bue. Structure from motion with objects. In *CVPR*, 2016.
- [176] Paul Gay, James Stuart, and Alessio Del Bue. Visual graphs from motion (vgfm): Scene understanding with object geometry reasoning. In *ACCV*, 2019.
- [177] Vincent Gaudillière, Gilles Simon, and Marie-Odile Berger. Perspective-2-ellipsoid: Bridging the gap between object detections and 6-dof camera pose. *RAL*, 5(4):5189–5196, 2020.
- [178] Mo Shan, Qiaojun Feng, You-Yi Jau, and Nikolay Atanasov. Ellipsdf: joint object pose and shape optimization with a bi-level ellipsoid and signed distance function description. In *ICCV*, 2021.
- [179] Shujia Chen, Shuangfu Song, Junqiao Zhao, Tiantian Feng, Chen Ye, Lu Xiong, and Deyi Li. Robust dual quadric initialization for forward-translating camera movements. *RAL*, 6(3):4712–4719, 2021.
- [180] Mehdi Hosseinzadeh, Yasir Latif, Trung Pham, Niko Suenderhauf, and Ian Reid. Structure aware slam using quadrics and planes. In *ACCV*, 2019.
- [181] Tristan Laidlow and Andrew J Davison. Simultaneous localisation and mapping with quadric surfaces. In *3DV*, 2022.
- [182] Ziwei Liao, Yutong Hu, Jiadong Zhang, Xianyu Qi, Xiaoyu Zhang, and Wei Wang. So-slam: Semantic object slam with scale proportional and symmetrical texture constraints. *IEEE RAL*, 7(2):4008–4015, 2022.
- [183] Vincent Gaudillière, Gilles Simon, and Marie-Odile Berger. Camera relocalization with ellipsoidal abstraction of objects. In *ISMAR*, 2019.
- [184] Yongqi Meng and Benchun Zhou. Ellipsoid slam with novel object initialization. In *CASE*, 2022.
- [185] Matthieu Zins, Gilles Simon, and Marie-Odile Berger. Oa-slam: Leveraging objects for camera relocalization in visual slam. In *ISMAR*, 2022.
- [186] David G Lowe. Object recognition from local scale-invariant features. In *ICCV*.
- [187] Paul-Edouard Sarlin, Daniel DeTone, Tomasz Malisiewicz, and Andrew Rabinovich. Superglue: Learning feature matching with graph neural networks. In *CVPR*, 2020.
- [188] Seungwook Kim, Juhong Min, and Minsu Cho. Transmatcher: Match-to-match attention for semantic correspondence. In *CVPR*, 2022.

- 
- [189] Richard Sinkhorn. A Relationship Between Arbitrary Positive Matrices and Doubly Stochastic Matrices. *The Annals of Mathematical Statistics*, 35(2):876 – 879, 1964.
- [190] Xingyi He, Jiaming Sun, Yuang Wang, Di Huang, Hujun Bao, and Xiaowei Zhou. Onepose++: Keypoint-free one-shot object pose estimation without cad models. In *NeurIPS*, 2022.
- [191] Mingyu Ding, Zhe Wang, Jiankai Sun, Jianping Shi, and Ping Luo. Camnet: Coarse-to-fine retrieval for camera re-localization. In *ICCV*, 2019.
- [192] Dingding Cai, Janne Heikkilä, and Esa Rahtu. Gs-pose: Generalizable segmentation-based 6d object pose estimation with 3d gaussian splatting. In *3DV*, 2025.
- [193] Robotiq 2f-85. <https://robotiq.com/products/adaptive-grippers#Two-Finger-Gripper>. Accessed: 2025-03-27.
- [194] Schunk grippers. [https://schunk.com/us\\_en/gripping-systems/grippers/](https://schunk.com/us_en/gripping-systems/grippers/). Accessed: 2025-03-27.
- [195] Franka hand. [https://download.franka.de/documents/220010\\_Product%20Manual\\_Franka%20Hand\\_1.2\\_EN.pdf](https://download.franka.de/documents/220010_Product%20Manual_Franka%20Hand_1.2_EN.pdf). Accessed: 2025-03-27.
- [196] Onrobot gecko. <https://onrobot.com/en/products/gecko-gripper>. Accessed: 2025-03-28.
- [197] Onrobot mg10 gripper. <https://onrobot.com/en/products/mg10>. Accessed: 2025-03-27.
- [198] Robotiq epick. <https://robotiq.com/products/vacuum-grippers#EPick>. Accessed: 2025-03-27.
- [199] Schmalz. Electric needle grippers sngi-ae. <https://www.schmalz.com/en/vacuum-technology-for-automation/vacuum-components/special-grippers/needle-grippers/electric-needle-grippers-sngi-ae-306138/>. Accessed: 2025-04-29.
- [200] Barbara Mazzolai, Alessio Mondini, Francesca Tramacere, Gianluca Riccomi, Ali Sadeghi, Goffredo Giordano, Emanuela Del Dottore, Massimiliano Scaccia, Massimo Zampato, and Stefano Carminati. Octopus-inspired soft arm with suction cups for enhanced grasping tasks in confined environments. *Advanced Intelligent Systems*, 1(6):1900041, 2019.
- [201] Liang Zhou, Lili Ren, You Chen, Shichao Niu, Zhiwu Han, and Luquan Ren. Bio-inspired soft grippers based on impactive gripping. *Advanced Science*, 8(9), 2021.

- [202] Cecilia Laschi and Barbara Mazzolai. Bioinspired materials and approaches for soft robotics. *Mrs Bulletin*, 46(4):345–349, 2021.
- [203] Cecilia Laschi, Barbara Mazzolai, and Matteo Cianchetti. Soft robotics: Technologies and systems pushing the boundaries of robot abilities. *Science robotics*, 1(1), 2016.
- [204] Allegro hand. <https://www.allegrohand.com/>. Accessed: 2025-02-28.
- [205] Manuel G Catalano et al. Adaptive synergies for the design and control of the Pisa/IIT SoftHand. *The International Journal of Robotics Research*, 2014.
- [206] Zhongxu Hu, Runjia Tan, Yanxin Zhou, Junyang Woon, and Chen Lv. Template-based category-agnostic instance detection for robotic manipulation. *IEEE RAL*, 7(4):12451–12458, 2022.
- [207] Linghao Chen, Yunzhou Song, Hujun Bao, and Xiaowei Zhou. Perceiving unseen 3d objects by poking the objects. In *ICRA*, 2023.
- [208] Hao-Shu Fang, Chenxi Wang, Minghao Gou, and Cewu Lu. GraspNet-1Billion: A Large-Scale Benchmark for General Object Grasping. In *CVPR*, 2020.
- [209] Zhenggang Tang, Balakumar Sundaralingam, Jonathan Tremblay, Bowen Wen, Ye Yuan, Stephen Tyree, Charles Loop, Alexander Schwing, and Stan Birchfield. RGB-Only reconstruction of tabletop scenes for collision-free manipulator control. In *ICRA*, 2023.
- [210] Balakumar Sundaralingam, Siva Kumar Sastry Hari, Adam Fishman, Caelan Garrett, Karl Van Wyk, Valts Blukis, Alexander Millane, Helen Oleynikova, Ankur Handa, Fabio Ramos, Nathan Ratliff, and Dieter Fox. Curobo: Parallelized collision-free robot motion generation. In *ICRA*, 2023.
- [211] 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 *ICRA*, 2023.
- [212] 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. In *CVPRW*, 2023.
- [213] Thomas Weng, David Held, Franziska Meier, and Mustafa Mukadam. Neural grasp distance fields for robot manipulation. In *ICRA*, 2023.
- [214] 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 *IROS*, 2021.
- [215] Yunzhu Li, Shuang Li, Vincent Sitzmann, Pulkit Agrawal, and Antonio Torralba. 3d neural scene representations for visuomotor control. In *CoRL*, 2022.

- 
- [216] Yen-Chen Lin, Pete Florence, Andy Zeng, Jonathan T Barron, Yilun Du, Wei-Chiu Ma, Anthony Simeonov, Alberto Rodriguez Garcia, and Phillip Isola. Mira: Mental imagery for robotic affordances. In *CoRL*, 2023.
- [217] Anthony Simeonov, Yilun Du, Andrea Tagliasacchi, Joshua B Tenenbaum, Alberto Rodriguez, Pulkit Agrawal, and Vincent Sitzmann. Neural descriptor fields: Se (3)-equivariant object representations for manipulation. In *ICRA*, 2022.
- [218] Anthony Simeonov, Yilun Du, Yen-Chen Lin, Alberto Rodriguez Garcia, Leslie Pack Kaelbling, Tomás Lozano-Pérez, and Pulkit Agrawal. Se(3)-equivariant relational rearrangement with neural descriptor fields. In *CoRL*, 2023.
- [219] 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 *ICRA*, 2022.
- [220] Allan Zhou, Moo Jin Kim, Lirui Wang, Pete Florence, and Chelsea Finn. Nerf in the palm of your hand: Corrective augmentation for robotics via novel-view synthesis. In *CVPR*, 2023.
- [221] Shaohong Zhong, Alessandro Albini, Oiwi Parker Jones, Perla Maiolino, and Ingmar Posner. Touching a neRF: Leveraging neural radiance fields for tactile sensory data generation. In *CoRL*, 2022.
- [222] Danny Driess, Ingmar Schubert, Pete Florence, Yunzhu Li, and Marc Toussaint. Reinforcement learning with neural radiance fields. In *NeurIPS*, 2022.
- [223] Dongseok Shim, Seungjae Lee, and H Jin Kim. Snerl: Semantic-aware neural radiance fields for reinforcement learning. In *ICML*, 2023.
- [224] Yuhang Zheng, Xiangyu Chen, Yupeng Zheng, Songen Gu, Runyi Yang, Bu Jin, Pengfei Li, Chengliang Zhong, Zengmao Wang, Lina Liu, Chao Yang, Dawei Wang, Zhen Chen, Xiaoxiao Long, and Meiqing Wang. Gaussiangrasper: 3d language gaussian splatting for open-vocabulary robotic grasping. *RAL*, 9(9):7827–7834, 2024.
- [225] Hao-Shu Fang, Chenxi Wang, Hongjie Fang, Minghao Gou, Jirong Liu, Hengxu Yan, Wenhai Liu, Yichen Xie, and Cewu Lu. Anygrasp: Robust and efficient grasp perception in spatial and temporal domains. *IEEE Transactions on Robotics*, 39(5):3929–3945, 2023.
- [226] Mazeyu Ji, Ri-Zhao Qiu, Xueyan Zou, and Xiaolong Wang. Graspplats: Efficient manipulation with 3d feature splatting. In *CoRL*, 2024.
- [227] Guanxing Lu, Shiyi Zhang, Ziwei Wang, Changliu Liu, Jiwen Lu, and Yansong Tang. Manigaussian: Dynamic gaussian splatting for multi-task robotic manipulation. In *ECCV*, 2024.

- [228] Yulong Li and Deepak Pathak. Object-aware gaussian splatting for robotic manipulation. In *ICRA Workshop*, 2024.
- [229] Ola Shorinwa, Johnathan Tucker, Aliyah Smith, Aiden Swann, Timothy Chen, Roya Firoozi, Monroe David Kennedy, and Mac Schwager. Splat-mover: Multi-stage, open-vocabulary robotic manipulation via editable gaussian splatting. In *CoRL*, 2024.
- [230] Junqiu Yu, Xinlin Ren, Yongchong Gu, Haitao Lin, Tianyu Wang, Yi Zhu, Hang Xu, Yu-Gang Jiang, Xiangyang Xue, and Yanwei Fu. SparseGrasp: Robotic Grasping via 3D Semantic Gaussian Splatting from Sparse Multi-View RGB Images, 2024.
- [231] Jad Abou-Chakra, Krishan Rana, Feras Dayoub, and Niko Suenderhauf. Physically embodied gaussian splatting: A realtime correctable world model for robotics. In *CoRL*, 2024.
- [232] Yinzhen Xu, Weikang Wan, Jialiang Zhang, Haoran Liu, Zikang Shan, Hao Shen, Ruicheng Wang, Haoran Geng, Yijia Weng, Jiayi Chen, et al. Unidexgrasp: Universal robotic dexterous grasping via learning diverse proposal generation and goal-conditioned policy. In *CVPR*, 2023.
- [233] Guo-Hao Xu, Yi-Lin Wei, Dian Zheng, Xiao-Ming Wu, and Wei-Shi Zheng. Dexterous grasp transformer. In *CVPR*, 2024.
- [234] Yufei Ye, Xueting Li, Abhinav Gupta, Shalini De Mello, Stan Birchfield, Jiaming Song, Shubham Tulsiani, and Sifei Liu. Affordance diffusion: Synthesizing hand-object interactions. In *CVPR*, 2023.
- [235] Yiheng Xie, Towaki Takikawa, Shunsuke Saito, Or Litany, Shiqin Yan, Numair Khan, Federico Tombari, James Tompkin, Vincent Sitzmann, and Srinath Sridhar. Neural fields in visual computing and beyond. *Computer Graphics Forum*, 41(2):641–676, 2022.
- [236] Richard Hartley and Andrew Zisserman. *Multiple view geometry in computer vision*. Cambridge University Press, 2003.
- [237] Gregory J Chaitin. Register allocation & spilling via graph coloring. *ACM Sigplan Notices*, 17(6):98–101, June 1982.
- [238] Charles R Qi, Or Litany, Kaiming He, and Leonidas J Guibas. Deep hough voting for 3d object detection in point clouds. In *ICCV*, 2019.
- [239] Zhou Wang, Alan C Bovik, Hamid R Sheikh, and Eero P Simoncelli. Image quality assessment: from error visibility to structural similarity. *TPAMI*, 13(4):600–612, 2004.

- 
- [240] Richard Zhang, Phillip Isola, Alexei A Efros, Eli Shechtman, and Oliver Wang. The Unreasonable Effectiveness of Deep Features as a Perceptual Metric. In *CVPR*, 2018.
- [241] Pat Marion, Peter Florence, Lucas Manuelli, and Russ Tedrake. Label fusion: A pipeline for generating ground truth labels for real rgbd data of cluttered scenes. In *ICRA*, 2018.
- [242] Lucas Manuelli, Wei Gao, Peter R. Florence, and Russ Tedrake. kpm: Keypoint affordances for category-level robotic manipulation. In *International Symposium of Robotics Research*, 2019.
- [243] Sriyith Rajeev, Qianwen Wan, Kenny Yau, Karen Panetta, and Sos Agaian. Augmented reality-based vision-aid indoor navigation system in gps denied environment. In *Mobile Multimedia/Image Processing, Security, and Applications*, 2019.
- [244] Nicholas Metropolis, Arianna W Rosenbluth, Marshall N Rosenbluth, Augusta H Teller, and Edward Teller. Equation of state calculations by fast computing machines. *The journal of chemical physics*, 21(6), 1953.
- [245] Luc Masset, Olivier Brls, and Gatan Kerschen. Partition of the circle in cells of equal area and shape. Technical report, Structural Dynamics Research Group, Aerospace and Mechanical Engineering Department, University of Liege, ‘Institut de Mecanique et G ´enie Civil (B52/3), 2011.
- [246] Benoit Beckers and Pierre Beckers. Fast and accurate view factor generation. In *FICUP, An International Conference on Urban Physics*, 2016.
- [247] TJ Malley. A shading method for computer generated images. *Master’s thesis, Dept. of Computer Science, University of Utah*, 1988.
- [248] Lionel Jacques, Luc Masset, and Gatan Kerschen. Direction and surface sampling in ray tracing for spacecraft radiative heat transfer. *Aerospace Science and Technology*, 47, 2015.
- [249] Theodore Tsesmelis, Irtiza Hasan, Marco Cristani, Alessio Del Bue, and Fabio Galasso. Rgbd2lux: Dense light intensity estimation with an rgbd sensor. In *WACV*, 2018.
- [250] Matthew Tancik, Pratul P. Srinivasan, Ben Mildenhall, Sara Fridovich-Keil, Nithin Raghavan, Utkarsh Singhal, Ravi Ramamoorthi, Jonathan T. Barron, and Ren Ng. Fourier features let networks learn high frequency functions in low dimensional domains. In *NeurIPS*, 2020.
- [251] Lingjie Liu, Jiatao Gu, Kyaw Zaw Lin, Tat-Seng Chua, and Christian Theobalt. Neural Sparse Voxel Fields. In *NeurIPS*, 2020.

- [252] Jonathon Luiten, Georgios Kopanas, Bastian Leibe, and Deva Ramanan. Dynamic 3d gaussians: Tracking by persistent dynamic view synthesis. In *3DV*, 2024.
- [253] Tianyi Xie, Zeshun Zong, Yuxing Qiu, Xuan Li, Yutao Feng, Yin Yang, and Chenfanfu Jiang. Physgaussian: Physics-integrated 3d gaussians for generative dynamics. In *CVPR*, 2024.
- [254] Google maps nerf integration. <https://blog.google/products/maps/sustainable-immersive-maps-announcements/>. Accessed: 2023-11-15.
- [255] Dominic Maggio, Courtney Mario, and Luca Carlone. Verf: Runtime monitoring of pose estimation with neural radiance fields. In *ICCV*, 2023.
- [256] Gert Almkvist and Bruce Berndt. Gauss, landen, ramanujan, the arithmetic-geometric mean, ellipses,  $\pi$ , and the ladies diary. *The American Mathematical Monthly*, 95(7):585–608, 1988.
- [257] Federico Tombari, Samuele Salti, and Luigi di Stefano. Unique signatures of histograms for local surface description. In *ECCV*, 2010.
- [258] Noam Shazeer and Mitchell Stern. Adafactor: Adaptive learning rates with sub-linear memory cost. In *ICML*, 2018.
- [259] Suhas Kadalagere Sampath, Ning Wang, Hao Wu, and Chenguang Yang. Review on human-like robot manipulation using dexterous hands. *Cognitive Computation and Systems*, 2023.
- [260] Raymond R Ma and Aaron M Dollar. On dexterity and dexterous manipulation. In *ICAR*, 2011.
- [261] Rhys Newbury et al. Deep learning approaches to grasp synthesis: A review. *TRO*, 2023.
- [262] Seyed S Mohammadi et al. 3dsgrasp: 3d shape-completion for robotic grasp. In *ICRA*, 2023.
- [263] Javier Romero, Dimitrios Tzionas, and Michael J. Black. Embodied Hands: Modeling and Capturing Hands and Bodies Together. *TOG*, 2017.
- [264] Hui Zhang, Sammy Christen, Zicong Fan, Otmar Hilliges, and Jie Song. GraspXL: Generating grasping motions for diverse objects at scale. In *ECCV*, 2024.
- [265] Jiawei Yang, Katie Z Luo, Jiefeng Li, Congyue Deng, Leonidas J. Guibas, Dilip Krishnan, Kilian Q Weinberger, Yonglong Tian, and Yue Wang. Dvt: Denoising vision transformers. In *ECCV*, 2024.
- [266] Qichen Fu, Xingyu Liu, Ran Xu, Juan Carlos Niebles, and Kris M. Kitani. Deformer: Dynamic fusion transformer for robust hand pose estimation. In *ICCV*, 2023.



- 
- [267] Matt Deitke, Dustin Schwenk, Jordi Salvador, Luca Weihs, Oscar Michel, Eli VanderBilt, Ludwig Schmidt, Kiana Ehsani, Aniruddha Kembhavi, and Ali Farhadi. Objaverse: A universe of annotated 3d objects. In *CVPR*, 2023.
- [268] Daiheng Gao et al. DART: Articulated Hand Model with Diverse Accessories and Rich Textures. In *NeurIPS*, 2022.
- [269] M. Musy. vedo, a python module for scientific analysis and visualization of 3D objects and point clouds, 2021.
- [270] Yunze Liu, Yun Liu, Che Jiang, Kangbo Lyu, Weikang Wan, Hao Shen, Boqiang Liang, Zhoujie Fu, He Wang, and Li Yi. Hoi4d: A 4d egocentric dataset for category-level human-object interaction. In *CVPR*, 2022.
- [271] 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 *ICAR*, 2015.
- [272] Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. In *ICLR*, 2019.
- [273] Vickie Ye, Ruilong Li, Justin Kerr, Matias Turkulainen, Brent Yi, Zhuoyang Pan, Otto Seiskari, Jianbo Ye, Jeffrey Hu, Matthew Tancik, and Angjoo Kanazawa. gsplat: An open-source library for Gaussian splatting, 2024.
- [274] Viktor Makoviychuk et al. Isaac Gym: High Performance GPU Based Physics Simulation For Robot Learning. In *NeurIPS*, 2021.
- [275] Alexander Khazatsky, Karl Pertsch, Suraj Nair, Ashwin Balakrishna, Sudeep Dasari, Siddharth Karamcheti, Soroush Nasiriany, Mohan Kumar Srirama, Lawrence Yunliang Chen, Kirsty Ellis, et al. Droid: A large-scale in-the-wild robot manipulation dataset. In *RSS 2024 Workshop: Data Generation for Robotics*, 2024.