

Diploma Thesis

ZEROSHOP - ZERO-SHOT OBJECT POSE ESTIMATION IN UNCONSTRAINED ENVIRONMENTS

Conducted in partial fulfillment of the requirements for the degree of a
Diplom-Ingenieur (Dipl.-Ing.)

supervised by

Ao.Univ.Prof. Dipl.-Ing. Dr.techn. M. Vincze
Dipl.-Ing. P. Ausserlechner


submitted at the

TU Wien

Faculty of Electrical Engineering and Information Technology
Automation and Control Institute

by

Stefan Lechner



Vienna, January 2026

Vision for Robotics Group

A-1040 Wien, Gusshausstr. 27, Internet: <http://www.acin.tuwien.ac.at>

Preamble

During my Bachelor's thesis, I first encountered Professor Markus Vincze's research group Vision for Robotics (V4R), where I was warmly welcomed and well guided. This positive experience led me back to V4R during my Master's degree for another project in the Robot Vision lecture. There, Philipp Ausserlechner introduced me to deep feature extraction from a foundation model for zero-shot 6D object pose estimation, a project that captured my interest in applied machine learning.

As I approached the end of my Master's program, I was actively searching for both a compelling thesis topic and the right research group. Through a project partner, I learned that V4R was offering new computer vision topics. To my surprise, Philipp had posted what seemed like the perfect match for my interests. I immediately reached out to him and, spoiler alert, was accepted for the thesis. This decision proved to be one of the best I have ever made, and I am deeply grateful to Philipp for providing exactly the right balance of guidance and independence, ensuring that I never felt lost while still allowing me to develop my research skills.

I also want to acknowledge all the people who made my time at TU WIEN memorable, starting with my HTL friends and ending with all those Germans who appeared during my Master's program. Special thanks go to my close robotics friends group from the later years of my studies: Julia Kuhn, Benedikt Frey, Dario Spoljaric, Zoltán Varga, and Clemens Hacker. Most importantly, I am forever grateful to my family, whose never-ending support and encouragement carried me through every challenge of my university journey.

To ensure transparency in this era of AI, I note that Grammarly and Writefull were used for grammar checks, and the robots in Figure 2.1 were segmented with Google's Nano Banana. Additionally, the abstract was translated into German with Claude Sonnet 4.5 and manually improved.

Vienna, January 2026

Abstract

Robotic manipulation of unseen objects often relies on zero-shot 6D object pose estimation methods, which typically employ a mesh as a reference representation. However, constructing high-fidelity meshes generally requires specialized hardware-based 3D scanning technologies and manual editing. The development of an automated software solution that requires only an object video as input could eliminate the need for expensive hardware, increase accessibility, and reduce the need for human intervention. Recently proposed Novel View Synthesis (NVS) techniques, including 2D Gaussian Splatting (2DGS) and Sparse Voxels Rasterization (SVRaster), iteratively reconstruct unconstrained environments to generate photorealistic 3D scenes, with accurate surface reconstruction emerging as a byproduct. Consequently, the primary objective of this research was to develop an automated mesh generation pipeline that integrates this high-quality 3D information, comprising Data Acquisition, Camera Registration and Pointcloud Generation, Metric Height Estimation, and subsequent NVS Mesh Generation. Given successful camera registration with MAST3R-SfM or VGGT, 2DGS produces highly accurate meshes in minutes, while SVRaster produces meshes with lower geometric accuracy but achieves training in seconds. For metric scale estimation, grounding near-view object-centric images with far-view scanning scene images using MAST3R yields consistent estimates. Evaluation with BOP YCB-V demonstrates strong performance on the segmentation and pose estimation methods CNOS, SAM-6D, and FoundationPose, with no significant differences between the 2DGS and SVRaster meshes. Finally, empirical robotic grasping experiments with supermarket objects and the best-performing perception pipeline CNOS/FoundationPose indicate robust performance even in the presence of moderate scale errors in the generated meshes.

Kurzzusammenfassung

Die robotische Manipulation unbekannter Objekte stützt sich häufig auf Zero-Shot-Methoden zur 6D-Objektposenschätzung, die typischerweise ein Mesh als Referenzrepräsentation verwenden. Die Erstellung hochauflösender Meshes erfordert jedoch in der Regel spezialisierte hardwarebasierte 3D-Scanning-Technologien und manuelle Nachbearbeitung. Die Entwicklung einer automatisierten Softwarelösung, die lediglich ein Objektvideo als Eingabe benötigt, könnte den Bedarf an teurer Hardware eliminieren, die Zugänglichkeit erhöhen und den manuellen Aufwand reduzieren. Neue Techniken zur Novel View Synthesis (NVS), darunter 2D Gaussian Splatting (2DGS) und Sparse Voxels Rasterization (SVRaster), rekonstruieren iterativ uneingeschränkte Umgebungen, um fotorealistische 3D-Szenen zu erzeugen, wobei eine präzise Oberflächenrekonstruktion als Nebenprodukt entsteht. Folglich bestand das Hauptziel dieser Forschung darin, eine automatisierte Mesh-Generierungspipeline zu entwickeln, die diese hochwertigen 3D-Informationen integriert und aus Datenerfassung, Kameraregistrierung und Punktwolkengenerierung, metrischer Höhenschätzung sowie anschließender NVS-basierter Mesh-Generierung besteht. Bei erfolgreicher Kameraregistrierung durch MAST3R-SfM oder VGGT erzeugt 2DGS hochpräzise Meshes innerhalb von Minuten, während SVRaster Meshes mit geringerer geometrischer Genauigkeit produziert, das Training aber nur Sekunden dauert. Für die metrische Skalierung lieferte die Kombination von nahansichtigen objektzentrierten Bildern und weitwinkligen Scanszenenbildern mittels MAST3R konsistente Schätzungen. Die Evaluierung mit BOP YCB-V zeigt eine starke Leistung bei den Segmentierungs- und Posenschätzungsmethoden CNOS, SAM-6D und FoundationPose, ohne signifikante Unterschiede zwischen den 2DGS- und SVRaster-Meshes. Schließlich deuten empirische robotische Greifexperimente mit Supermarktobjekten und der leistungsstärksten Posenschätzungspipeline CNOS/FoundationPose auf eine robuste Leistung selbst bei moderaten Skalierungsfehlern in den generierten Meshes hin.

Contents

1	Introduction	1
1.1	Motivation	2
1.2	Challenge	3
1.3	Contributions	4
1.4	Outline	6
2	Related Works	8
2.1	Robotic Research Platform	8
2.2	From Visual Features to Object Pose Estimation	9
2.3	3D Reconstruction	12
2.4	Metric Scale Estimation	14
2.5	Data Sources for 3D Reconstruction	15
3	Methods	16
3.1	3D Data Modalities	16
3.1.1	Pointcloud	16
3.1.2	Mesh	17
3.1.3	Object Pose	17
3.2	Benchmark and Datasets	18
3.2.1	Benchmark for 6D Object Pose Estimation	18
3.2.2	MANiBOT	20
3.3	Data Acquisition	21
3.3.1	Imaging Setup	21
3.3.2	Segmentation	22
3.4	Camera Registration and Pointcloud Generation	22
3.4.1	COLMAP and Structure from Motion	23
3.4.2	MASt3R and Multi-View Stereo	24
3.4.3	MASt3R-SfM	24
3.4.4	Visual Geometry Grounded Transformer (VGGT)	25
3.5	Novel View Synthesis for Mesh Generation	26
3.5.1	2D Gaussian Splatting	26
3.5.2	Sparse Voxels Rasterization	28
3.5.3	Generative 3D Reconstruction (Trellis)	29
3.6	Metric Height Estimation	30
3.7	Object Segmentation and Pose Estimation	31
3.7.1	CNOS	31
3.7.2	SAM-6D	32
3.7.3	FoundationPose	34
3.8	Robotic Platform	36
3.9	Mesh Generation Pipeline	37

4 Experiments and Results	40
4.1 Mesh Generation Reliability and Quality	40
4.1.1 Virtual YCB-V Objects	40
4.1.2 Real YCB-V and MANiBOT Objects	44
4.2 Object Scaling	47
4.3 BOP with NVS Meshes	47
4.4 Robotic Object Manipulation	49
4.4.1 Segmentation	50
4.4.2 Pose Estimation	50
4.4.3 Grasp	52
5 Conclusion	57
6 Outlook	61

List of Figures

1.1	A robot extracts environmental information relevant to a target object for grasping. This data, including image location and corresponding pixel values, is then used to infer both a green 2D detection and a blue 3D pose, defined within a virtual coordinate system. Subsequently, this pose information facilitates grasp planning for object manipulation.	2
1.2	Object generation proceeds in four stages: Object Segmentation generates image masks from a video, followed by ML Camera Registration, which estimates camera poses and a sparse pointcloud. Metric Height Estimation then isolates object-dependent 3D points from masked front-view images to estimate the object dimensions. Finally, the pointcloud is densified using the NVS methods 2D Gaussian Splatting and SVRaster. While iterative NVS requires meshing, the GenAI model Trellis generates the mesh directly from images.	5
2.1	Wheeled, legged and humanoid robot: HSR [38], ANYmal [39], Atlas [40].	9
2.2	Examples of common 3D data representations include: Voxels model an object as 3D blocks on a rigid 3D grid; Meshes approximate surfaces using triangular surfaces; Pointclouds store discrete points in 3D space; Octrees are voxel-like but organized hierarchically in a tree data structure; Descriptors encode view-dependent shape information; Multi-view data consists of multiple calibrated cameras and their corresponding images; RGBD data combines continuous depth maps with RGB information; Graphs capture the geometric essence by linking different shape components; Projections map 3D points onto 2D planes; and CAD models describe objects using parametric or functional representations. [46]	10
2.3	Extracted and processed features from foundation models; the left side depicts color-graded object features from DINOv2 [59]. The middle visualizes the features from DUS3R as a depth map [6], which are then used to register camera positions to generate a 3D pointcloud of a bench.	12
2.4	NeRF NVS reconstruction [70]. Input images and associated camera poses are optimized to generate novel views of a sparse scene.	13
2.5	Dataset example of Mip-NeRF 360 [75], DTU Robot Image Dataset [76] and MANiBOT [25].	14
3.1	Extracted pointcloud and mesh depicting the cap of the mustard bottle from Figure 1.2.	17
3.2	Test images from the BOP YCB-V dataset [31].	18

3.3	The BOP subset YCB-V [31] consists of 21 everyday objects typically found in households and supermarkets. The objects are numbered sequentially, beginning with the first object in the upper-left corner and proceeding from left to right and from top to bottom. Under this scheme, the object located in the lower-right corner corresponds to 21.	19
3.4	Subset of the supermarket object dataset from the MANiBOT project [25], featuring semi-transparent items, diverse geometries, and deformable properties.	20
3.5	Camera trajectory circling an object two times to generate a diverse set of camera poses.	21
3.6	The first frame of the object video is segmented using Grounded-SAM with the prompt "object in the middle." This generates an initial segmentation mask that starts the mask-tracking process, which is then used to remove the scene background from the object.	22
3.7	The COLMAP SfM pipeline consists of two primary stages: Correspondence Search and Incremental Reconstruction. Initially, SIFT features are detected and matched across images and subsequently validated using projective geometric constraints. Then, an initial image pair is selected to start the scene reconstruction. Additional images are iteratively registered using PnP, new 3D points are triangulated, and BA refines the reconstruction by minimizing an error term, after which outliers are removed. This procedure is repeated until all images have been successfully registered to the reconstructed scene. [22]	23
3.8	Overview of the MAST3R-SfM pipeline, which integrates MAST3R and COLMAP to improve SfM. The MAST3R encoder is employed as a feature retrieval backbone to construct a sparse scene graph from an image set. Next, the MAST3R decoder performs pairwise matching to produce a set of point maps. These outputs are then fed into the COLMAP pipeline to estimate camera poses and reconstruct a pointcloud. [30]	24
3.9	Visual comparison of an object-centric pointcloud generated by MAST3R-SfM (left) and VGGT (right).	25
3.10	Left: Illustration of how the 2D gaussian surface ensures view consistency between registered image planes compared to 3DGS. Right: Detailed visualization of the elliptical disk in both the tangent and image frames, parameterized by its center point \mathbf{p}_k , tangential vectors \mathbf{t}_u and \mathbf{t}_v , and scaling factors s_u and s_v . [10]	26
3.11	The left side illustrates the octree data structure of SVRaster to ensure correct rendering order and adaptive detail representation, where each voxel holds its own density values. These densities are trilinearly interpolated inside the voxels, based on shared corner grid points, and are used to compute the opacity value α through volume integration of K samples along the ray-voxel intersections. [12]	28

- 3.12 Overview on how Trellis infers a mesh from an image. First, a SLAT representation is learned from a voxelized 3D asset and its DINOv2 features, inhibiting both geometry and appearance information, to decode this representation into different 3D modalities. Followed by two-step 3D asset generation utilizing 3D noise and image conditions as an input to generate a sparse structure, which is then similarly refined with local latents, to use the trained decoder on the generated SLAT to decode it into a mesh. [11] 29
- 3.13 To generate an accurate metric scene pointcloud, four scene images (excluding the object) are registered with two images of the object using MAST3R. Subsequently, the first registered image and its corresponding object mask are utilized to extract object-specific points, which are then employed to estimate the height of the scanned object. 31
- 3.14 The three main stages of CNOS. Onboarding stage: Renders different viewpoints of the CAD models into templates, which are then processed with DINOv2 to create reference descriptors. Proposal stage: Employs SAM [89] or FastSAM [90] to generate proposal RGB masks from an RGB image, after which DINOv2 is applied again to extract proposal descriptors. Matching stage: All generated features are processed using cosine similarity to produce scores, which are then aggregated to identify the object ID with the highest confidence value. [26] 32
- 3.15 Overview of the SAM-6D PEM pipeline: In the Feature Extraction stage, a ViT is applied to both the rendered template images and the scene image to obtain features. These features, along with a background token, are subsequently used as input for a Geometric Transformer in the Coarse Point Matching stage, where a coarse pose is inferred for each template. The resulting hypotheses are then ranked according to a corresponding score. Subsequently, each sparse pose, the ViT-derived features, a pointcloud representation of the mesh, a depth map, and the background token are provided as input to an SDPT in the Fine Point Matching stage. The resulting outputs per coarse pose are then combined and weighted using singular value decomposition to determine the final fine pose. [8] 33
- 3.16 In the context of mesh-based object pose estimation, FoundationPose can be decomposed into three stages. To train its ML model, FoundationPose leveraged Language-aided Synthetic Data Generation, which modifies object textures using a diffusion model conditioned on diverse prompts obtained from ChatGPT. This process yields a varied synthetic dataset that is subsequently rendered within a physics engine. For pose estimation, the method first produces pose hypotheses by rendering the mesh and corresponding depth map from multiple viewpoints. Each of these candidates is then processed together with an RGBD pose conditioned input crop by a CNN and a transformer decoder, which iteratively updates rotation and translation until a refined pose is obtained. After this refinement, the resulting set of K hypotheses is evaluated in the Pose Selection stage by a Pose Ranking Encoder, and linearly projected, where the hypotheses are compared to one another to determine the best object pose. [9] 35

3.17	Overview of how the Grasping Pipeline from V4R interacts with the implemented Pose Estimation and Instance Segmentation ROS servers. The pipeline initiates by transmitting image data from the HSR to either SAM-6D ISM or CNOS for segmentation mask inference and associated object ID identification. Once an object is detected, the newly acquired data, combined with the previous image data, is sent to either SAM-6D PEM or FoundationPose for object pose estimation. Annotations on each server show the object-specific data needed at inference time.	37
3.18	Detailed overview of the mesh generation pipeline, comprising four stages: Data Acquisition, Camera Registration and Pointcloud Generation, Metric Height Estimation, and Novel View Synthesis for Mesh Generation. Data Acquisition yields an object-specific dataset captured by a camera and Grounded-SAM. The data consists of object/scene images and object masks, which are subsequently used for mesh generation. Camera Registration and Pointcloud Generation utilizes MAST3R-SfM or VGGT to generate camera poses and a pointcloud from the object images. Independent of this, Metric Height Estimation reconstructs a metric scene pointcloud from object and scene images using MAST3R, refining it by projecting onto the first registered camera plane and utilizing an object mask to infer the height of the object. Finally, Novel View Synthesis for Mesh Generation creates and post-processes the mesh with texture and pose, optionally scaling it to correct dimensions. 2DGS and SVRaster are the primary NVS methods analyzed in this thesis, while Trellis provides a baseline without camera pose information.	38
4.1	YCB-V object rendering with high/low feature surface and extracted segmented RGB masks; rendering angles of 0° , 45° , and -45° to the horizontal plane.	41
4.2	The first two objects represent qualitative examples of the surface of Trellis meshes, whereas the colored objects serve as representative instances of the inaccurate texture.	42
4.3	The first mesh shows a sparse SVRaster reconstruction, followed by the 2DGS mesh and its textured post-processed counterpart, and finally the YCB-V GT mesh.	42
4.4	Overview of the reconstruction quality of three geometrically distinct YCB-V objects. Of each pair, the left object is from 2DGS and the right from SVRaster.	44
4.5	Mesh quality comparison between 2DGS and SVRaster on real-world data, showing top and bottom views, with 2DGS yielding more accurate surfaces.	44
4.6	Reconstructed YCB-V objects, with the left/upper object illustrating reconstruction using 2DGS, followed by SVRaster.	45
4.7	Reconstructed MANiBOT objects, with the left/upper object illustrating reconstruction using 2DGS, followed by SVRaster. Upon application of the texture, no apparent differences become visible.	46

4.8	The relationship between the real object height and the absolute measurement error in percent for objects from the YCB-V and MANiBOT datasets. Individual measurements are represented as blue points, with a red line indicating the trend of decreasing error as object height increases.	48
4.9	Representative test scenes, with the second column illustrating CNOS segmentations and the third column presenting SAM-6D ISM segmentations. The segmentations in the first and second rows are generated using FastSAM, whereas those in the final row are produced using SAM.	51
4.10	Representative segmentation outcome obtained using the most robust CNOS/SAM configuration on MANiBOT objects.	51
4.11	Preliminary pose evaluation test scenes to quantify relative differences among GT, 2DGS, and SVRaster meshes, with FoundationPose/CNOS and SAM-6D.	52
4.12	Overview of the eight object poses used for grasping experiments. If an object cannot be stably inverted, it is positioned on its side, as illustrated with the "mustard_bottle" object.	53
4.13	Example of a successfully executed robotic grasp on the smallest test object, the "toothbrush".	55
4.14	Overview of the multi-object grasp evaluation scenes, including object segmentations and corresponding 6-DoF poses projected onto the image plane. Two recurring issues are illustrated. In the first row, the deformable object was generated within a compressed configuration; consequently, FoundationPose estimates its pose with a systematic offset. Additionally, the "toothbrush" is incorrectly segmented, resulting in the estimated pose being wrongly associated with an incorrect object instance.	55
4.15	Illustrative example of one of the six exploratory multi-object scenes with overlapping object boundaries, including object segmentations and their corresponding 6-DoF poses projected onto the image plane. The visualization highlights recurrent failure modes: mutual misclassification between "soap" and "razors", duplicate segmentation of a single object with two distinct class labels, as exemplified by "ahorn_sirup", and missing segmentations for objects with more strongly overlapping boundaries.	56

List of Tables

4.1	2DGS YCB-V mesh reconstruction success initiated by VGGT (quarter precision) and MAST3R-SfM: \times = none, \circ = sparse, \bullet = good	43
4.2	SVRaster YCB-V mesh reconstruction success initiated by MAST3R-SfM and VGGT (quarter precision): \times = none, \circ = sparse, \bullet = good. The lower part depicts 2DGS and VGGT (full precision).	43
4.3	Comparison between the actual heights of YCB-V objects and those estimated using the MAST3R registration method.	47
4.4	Comparison between the actual heights of MANiBOT objects and those estimated using the MAST3R registration method.	47
4.5	Official BOP AP (segmentation) and AR (pose estimation) scores compared to reproduced scores across different mesh datasets from the BOP objects. FoundationPose was additionally benchmarked with GT masks because the official filtering of the CNOS masks was not accessible.	49
4.6	Robotic manipulation performance comparison using 2DGS and SVRaster meshes. For each object, the first row reports segmentation success, and the second row the grasping success for each evaluated pose with: \times = failure, \circ = multi-shot, \bullet = single-shot.	54

1 Introduction

Automation is an integral part of our modern society, allowing humanity to eliminate repetitive tasks. However, it frequently remains unseen in daily life, primarily being deployed in industrial settings. Until recently, it was largely static, exemplified by systems such as motorway vignette checks via cameras or high-bay warehouses managed by location-dependent robotic systems. A new wave of automation is now emerging, driven by the increasing availability of mobile robotic platforms. For example, e-commerce companies are deploying hundreds of collaborative mobile robots to sort and prepare goods for delivery [1], or Waymo’s push for self-driving taxi cars [2]. However, these systems still operate in controlled, laboratory-like environments, relying on physical markers and predefined floor plans such as street grids.

The rise of Machine Learning (ML) and Generative Artificial Intelligence (GenAI) has significantly enhanced the ability of robotic systems to navigate complex and dynamic environments, moving beyond the constraints of controlled settings [3]. This is evident in the development of bipedal robots, which are evolving beyond traditional mobile platforms to incorporate human-like arms and limbs, enabling operation in a broader range of real-world settings where humans live and work. This shift opens up new possibilities for automation that were previously unattainable [4].

However, perfect locomotion and grasping in robots are of limited utility without reliable object detection and interaction capabilities. Consequently, robots utilize sensors and cameras to collect information from their surroundings. Robot vision, a subset of Computer Vision (CV), is benefiting exponentially from advances in ML [3]. Furthermore, the emergence of generalizable ML models, specifically foundation models, enables broader applicability. For example, models capable of understanding 2D or 3D structures can be applied to various downstream tasks, such as object pose estimation [5] or stereo 3D reconstruction [6]. Foundation models are trained unsupervised on extensive datasets, enabling them to learn patterns and structures directly from unlabeled data, without requiring human annotation. Therefore, this allows the development of more generalizable systems compared to specialized ML models, which have long been the industry standard [7]. Subsequently, when a model learns highly generalizable features that are present across diverse domains, it can be applied in a zero-shot manner, eliminating the need for task-specific training and offering adaptability to new scenarios.

The focus of this thesis is a subtopic within robotic automation, robotic vision, with a specific emphasis on object generation and perception to facilitate object manipulation. To enable manipulation, an object should be describable with Six Degrees of Freedom (6-DoF), which includes the position within a three-dimensional (3D) coordinate system and three angles (roll, pitch, yaw) to describe the rotation of the object. This is typically referred to as the 6D pose of the object. Estimating the pose of a real object can be done by

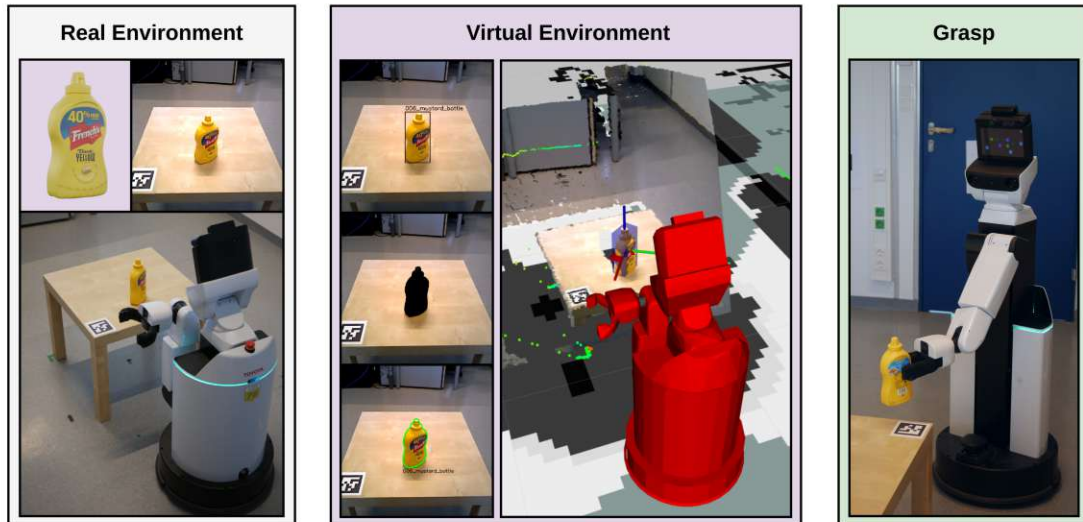


Figure 1.1: A robot extracts environmental information relevant to a target object for grasping. This data, including image location and corresponding pixel values, is then used to infer both a green 2D detection and a blue 3D pose, defined within a virtual coordinate system. Subsequently, this pose information facilitates grasp planning for object manipulation.

comparing it with a virtual pendant [8]. For this to be achievable, a virtual representation of the object, commonly referred to as a 3D mesh, is generated and detected through a dedicated pose estimation pipeline. The generation, detection, segmentation, and pose estimation of objects benefit from new ML concepts such as foundation models [9] or Novel View Synthesis (NVS) [10] [11] [12]. Therefore, the interaction between automated mesh generation, enabled by novel mesh generation techniques, and the deployment of a pose estimation pipeline constitutes the core of this research. Figure 1.1 illustrates a robot operating within a real-world environment to locate an object from its captured scene image. The robot detects and segments the object, estimating its pose to project this information into a virtual scene, thereby enabling grasp planning and manipulation.

1.1 Motivation

Robotic manipulation of objects requires an understanding of object identity, location, and pose; intrinsically linked to spatial context. Although Vision-Language Models (VLMs) are increasingly embedding object representations within their parameters [13], current object detection and 6D pose estimation pipelines still rely on reference objects, typically represented as virtual 3D meshes for feature matching [14]. This matching increasingly benefits from CV foundation models that utilize deep features to improve generalization and enable zero-shot capabilities [15].

Object detection and pose estimation pipelines rely on accurate mesh generation, traditionally achieved through hardware scanning followed by manual refinement in Computer-Aided Design (CAD) software. However, recent advances in Artificial Intelli-

gence (AI) are enabling the automation of both scanning and post-processing. Systems such as Trellis [11] generate consistent 3D meshes directly from images, while approaches such as MAST3R [16] and Visual Geometry Grounded Transformer (VGGT) [17] produce camera positions and pointclouds suitable for refinement using ML techniques. Integrating real-world metric scales can be facilitated by ML models [18] or standard depth cameras.

Developing an automated software pipeline could substantially reduce costs by mitigating the need for specialized hardware. However, accuracy may be affected, necessitating a feasibility assessment and testing using a project-specific unseen dataset. Therefore, the effectiveness of the developed system will be evaluated on object-centric datasets and assessed through real-world experiments on the Toyota Human Support Robot (HSR) [19].

1.2 Challenge

Automation of initial pipeline steps without supervision is prone to propagating errors throughout the process. In the context of mesh-based 6D object pose estimation, the success or failure of the entire system hinges on accurate mesh generation. Consequently, factors such as mesh completeness, texture resolution, mesh size, and mesh alignment substantially impact performance. Choosing the appropriate mesh generation method to balance accuracy and complexity in an automated setting remains a critical challenge. However, this influence may be mitigated by identifying and implementing suitable mesh-based segmentation and pose estimation methods that leverage diversely trained zero-shot ML model pipelines. Nevertheless, ML models generally perform effectively only on trained data and underperform when faced with out-of-domain scenarios. Real-world scenes are often cluttered and difficult to fully describe within the constraints of any single dataset.

Therefore, implementing such methods in a robotic system raises further concerns. The capacity of a computer system to handle computational demands and its ability to generalize effectively to real-world conditions remain unclear. These challenges underscore the need for robust, adaptable, and resource-efficient approaches that can bridge the gap between simulated environments and real-world deployment. To achieve this, the following research questions were addressed:

How do mesh generation techniques, employed as automatic model generation procedures, affect the accuracy and robustness of zero-shot object pose estimation?

Conventional mesh generation methods, such as Marching Cubes [20] and Poisson Reconstruction [21], typically rely on pointcloud data as an initial input, generated from Structure from Motion (SfM) [22]. A frequent challenge with these algorithms is their tendency to produce meshes with topological inconsistencies, often requiring adjustment of parameters specific to the object being reconstructed [23]. An ML alternative uses dense reconstruction and point alignment to enhance pointcloud detail with NVS before

applying established mesh generation algorithms [10]. The NVS GenAI approach Trellis offers the prospect of direct mesh generation, bypassing intermediate pointcloud processing steps. However, their performance and suitability required a comparative analysis.

What are the optimal cost-effective depth map modalities for scaling 3D meshes to preserve pose estimation performance?

Accurate object pose estimation requires a correctly scaled 3D mesh [9]. For example, meshes generated by Trellis are scaled to 1 m [11], and pointclouds reconstructed with SfM algorithms exhibit an ambiguous metric scale [22]. To address this, depth map information provides the correct scale. This thesis investigated methods for deriving this scale using ML models [18] to reduce dependence on costly and specialized hardware.

To what extent can novel mesh generation techniques enable accurate robotic manipulation in real-world scenarios?

For example, FoundationPose [9] represents a unified foundation model for pose estimation, which was tested on a comprehensive set of public datasets. To validate its performance within the project scope, several NVS-generated objects [24] [25] were explored. Given that pose estimation relies on accurate instance segmentation methods, such as CNOS [26], this component was investigated as a potential influence on FoundationPose. Additionally, the methods were compared to SAM-6D [8], a combined instance segmentation and pose estimation method based on template rendering. Testing extended beyond benchmarking to include practical implementation and evaluation on a robotic platform using the Robot Operation System (ROS) [27], demonstrating real-world utility.

1.3 Contributions

Building on the prerequisite of avoiding specialized scanning hardware, the developed meshing pipeline relies solely on a generic camera, provided that it captures sharp images with a standard focal length that is neither fisheye nor telephoto. Consequently, the only input data that is required are RGB images; all other information is inferred using ML models. This resulted in a modular mesh generation pipeline that eliminates the need for expensive hardware, increases accessibility, and reduces the need for human intervention. Figure 1.2 illustrates the four main stages of object generation, enabling accurate pose estimation: Object Segmentation, Camera Registration and Pointcloud generation, Metric Height Estimation, and meshing supported by Novel View Synthesis. Each step is described in detail in the following paragraphs.

Beginning with data aggregation, two distinct categories of data sources were used. The first consisted of a virtual BlenderProc [28] scene designed to provide controllable and reproducible data, as illustrated in the upper left corner of Figure 1.2. The second comprised data acquired using a camera from a real-world scanning scene. Following correct capture, Grounded-SAM [29] was employed to segment the object, creating a mask

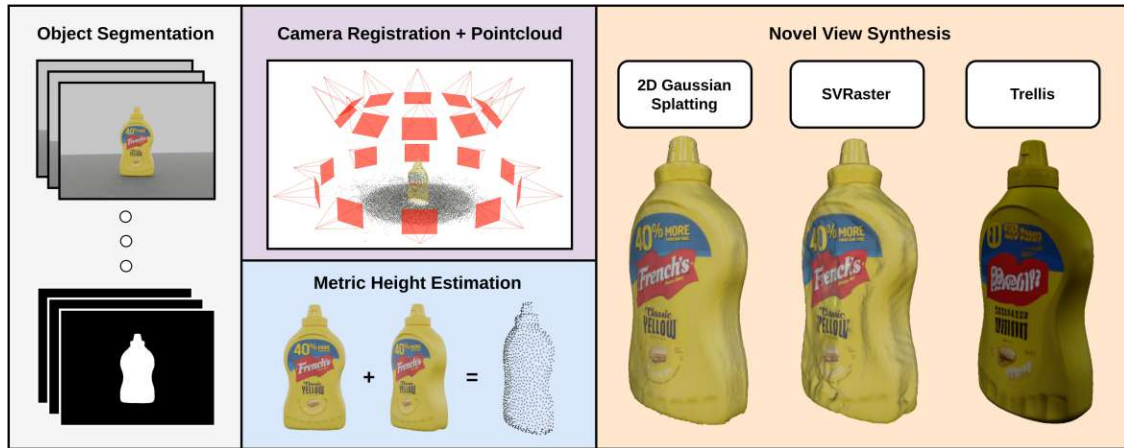


Figure 1.2: Object generation proceeds in four stages: Object Segmentation generates image masks from a video, followed by ML Camera Registration, which estimates camera poses and a sparse pointcloud. Metric Height Estimation then isolates object-dependent 3D points from masked front-view images to estimate the object dimensions. Finally, the pointcloud is densified using the NVS methods 2D Gaussian Splatting and SVRaster. While iterative NVS requires meshing, the GenAI model Trellis generates the mesh directly from images.

to guide mesh creation and extracting object-specific pointcloud parts for height estimation.

The unsegmented object images are then registered to each other using ML approaches, such as VGGT [17] or MAST3R-SfM [30]. The generated camera poses and a sparse pointcloud are depicted in the violet section of Figure 1.2. VGGT employs a brute-force approach, scaling an ML model until it requires server-grade hardware, processing all data at once to infer poses, a pointcloud, and depth maps in a single shot. In contrast, MAST3R-SfM is built around a multi-shot approach, iteratively performing stereo-matching on pairs of images and employing algorithms to extract 3D data.

Estimating the correct scale of an object from images alone using ML models proved to be challenging. Preliminary tests with State-of-the-Art (SOTA) depth estimation models showed inconsistent estimates and scale ambiguity for near-view object-centric images. Therefore, the MAST3R metric scene reconstruction approach was adapted to combine two masked near-view images of the object, as seen in Figure 1.2 in the blue section, with four images from the scanning scene (depicting far views), minimizing the influence of incorrect near-view estimations. Projecting the generated pointclouds onto the masked camera planes yields an object-only pointcloud, from which the height is obtained using a min/max point comparison.

After data generation, three NVS methods were applied and tested to generate object meshes; examples are shown in Figure 1.2. The first two methods are training-based and the third is an instant GenAI approach. The created pointcloud and camera poses are used to train 2D Gaussian Splatting (2DGS) [10] and Sparse Voxels Rasterization (SVRaster)

[12] on the scene; additionally, incorporating object masks allows them to free the object from the background. Trellis infers an object mesh with texture directly from images, without relying on camera registration. It should be noted that the mesh represented by 2DGS and SVRaster in Figure 1.2 underwent an automatic post-processing step to triangulate the correct texture with the camera poses and utilize the scale estimation to ensure the correct height. This is a step not applicable to the Trellis object because it does not employ a camera registration approach.

Following the automatic mesh generation pipeline, the resulting 3D objects are ready for use in segmentation and pose estimation ML pipelines. The objects were evaluated using the Benchmark for 6D Object Pose Estimation (BOP) [31], specifically the YCB-V video dataset. To generate each mesh type, the virtual YCB-V Ground-Truth (GT) objects were rendered with BlenderProc and then processed with the pipeline to create 2DGS, SVRaster, and Trellis meshes. Subsequently, CNOS, SAM-6D, and FoundationPose were evaluated for their respective tasks using all mesh configurations.

To evaluate the meshes under realistic conditions, they were tested with an established robotic object-grasping pipeline developed by the V4R research group. This pipeline controls the HSR [32] for object detection, segmentation, pose estimation, and grasping tasks. CNOS, SAM-6D, and FoundationPose were modularly implemented as ROS nodes, allowing direct integration into the existing codebase. Grasp tests performed on a real-world subset of the BOP YCB-V dataset and a subset of objects from the MANiBOT [25] project demonstrated that automating the mesh generation step does not significantly impact the accuracy of the best-performing perception pipeline CNOS/FoundationPose and subsequent robot manipulation. Nevertheless, pose estimation remains unreliable for thin objects with unobserved undersides, as well as in scenes characterized by a high degree of overlap or occluded object boundaries.

1.4 Outline

Beginning with a review of the literature, Chapter 2 analyzes existing methods and studies to identify knowledge gaps in the scientific landscape in robotics, image features, ML models, NVS methods, and datasets. In particular, with a focus on 3D reconstruction, object detection, segmentation, and pose estimation to enable robust object manipulation in real-world environments. Building on this foundation, Chapter 3 elaborates on the key concepts from the previous chapter in detail, introduces the data acquisition procedure, and describes how the height of an object can be estimated without the use of additional scanning hardware. The final two sections subsequently describe how the identified methods are integrated with the HSR platform and the automated mesh generation pipeline.

Performance evaluation is presented in Chapter 4, where the experimental setup and corresponding results are described in detail. First, the reliability of the mesh generation step is assessed, initially from data generated in a virtual environment and subsequently from data acquired in a real-world scanning scenario. Second, the proposed height estimation method is evaluated to determine the extent to which the estimated object height corresponds to the actual object height. Third, the segmentation and pose estimation

approaches are systematically narrowed down to the most effective methods through benchmarking with the BOP framework, preliminary experiments in real-world object scenes, and grasping experiments with the HSR robot using the best-performing methods. Finally, Chapter 5 concludes the thesis, and Chapter 6 outlines future research directions.

2 Related Works

This chapter reviews related work in the domain of robotic object manipulation, with a particular focus on mesh-based object pose estimation and closely related research topics. Section 2.1 provides a high-level overview of robotic perception and robot platforms. This is followed by Section 2.2, which provides an in-depth analysis of features in the CV domain, feature matching methodologies, and their respective roles in the context of 6-DoF object pose estimation. Subsequently, Section 2.3 compares 3D reconstruction methodologies for scenes and objects. Finally, Section 2.4 addresses the estimation of the metric scale, while Section 2.5 presents a review of diverse data sources that can be employed beyond their original intended application domains.

2.1 Robotic Research Platform

The challenge in enabling robots to interact with environments is accurately perceiving spatial relationships between objects and the robot coordinate system. Early pioneers such as Shakey the Robot [33] introduced foundational approaches to environmental navigation, employing ultrasound sensors and rudimentary cameras to detect boxes and surfaces. Similarly, Stanford Cart [34] advanced mobility with depth perception, demonstrating the integration of sensory systems for task-oriented interaction. These efforts enabled subsequent developments, such as the Stanford Arm, which demonstrated rudimentary object detection and manipulation by assembling a Ford Model T water pump. As research progressed, the complexity of perception systems grew, particularly in human-centric environments. Khatib et al. [35] analyzed this evolution, highlighting the need for robots to interpret dynamic scenes.

Modern robotic systems, such as Toyota HSR [19], exemplify this advancement, leveraging integrated perception capabilities to navigate cluttered spaces, perform everyday tasks, and collaborate seamlessly with humans. Based on the ROS framework [27], HSR provides a robust platform for robotic research and development, allowing systematic validation of perception pipelines in realistic domestic environments. However, other robots, such as ANYmal from ANYbotics [36] and Boston Dynamics Atlas [37] surpass the HSR in locomotion, autonomy, and system integration. Figure 2.1 shows the structural differences between the wheeled care robot HSR, the legged ANYmal, and the humanoid Atlas. Although HSR is modular and research-friendly, its mobility is limited compared to dynamic, closed, and highly capable systems from ANYbotics and Boston Dynamics. These newer robots are designed for challenging environments and complex tasks, reflecting the trend toward more integrated systems with superior locomotion and modularity. However, for many studies, the ability to traverse difficult terrains is not a critical requirement, allowing researchers to focus on other aspects of robotic perception and interaction [19]. Consequently, robots such as the HSR remain valuable tools for specific research needs.



Figure 2.1: Wheeled, legged and humanoid robot: HSR [38], ANYmal [39], Atlas [40].

2.2 From Visual Features to Object Pose Estimation

Early 2D detection methodologies, such as Scale-Invariant Feature Transform (SIFT) [41] and Histogram of Oriented Gradients (HOG) [42], are widely used for object localization in images. However, these approaches exhibited notable limitations in terms of accuracy and robustness, particularly in scenarios that involve lighting variations, featureless surfaces, or partial occlusions [42]. Object detection with 3D pose estimation takes it a step further by additionally determining the orientation and position in a three-dimensional space, often represented as a 6-DoF pose. This 3D problem is reversible to the 2D domain through multi-view representations, which depict objects from a set of predefined poses. The task then reduces to identifying the most accurate match, through SIFT or HOG features, between the observed 2D image and the generated multi-view representation.

Alternatively, objects can also be represented as pointclouds, which collectively define the object boundaries directly in 3D space. Pose estimation in such a context often leverages Iterative Closest Point (ICP) [43], which iteratively aligns pointcloud data by minimizing an error term. However, the effectiveness of ICP is highly dependent on the quality of the initial alignment, which requires robust initialization strategies. A complementary initialization approach involves establishing 2D-3D correspondences using Perspective-n-Point (PnP) [44]. In addition to multi-view and pointcloud data, several other 3D data modalities are available. Figure 2.2 illustrates a wide range of the most significant ones in CV. Among these, textured CAD models and meshes are regarded as the most precise and versatile due to the extensive 3D information they contain [45]. A key advantage of these representations is their ready convertibility to other modalities, enhancing the versatility of 3D object workflows, as demonstrated by their wide adoption across various domains, from video games to medical applications. However, meshes are computationally intensive when they have a high degree of fine detail [46].

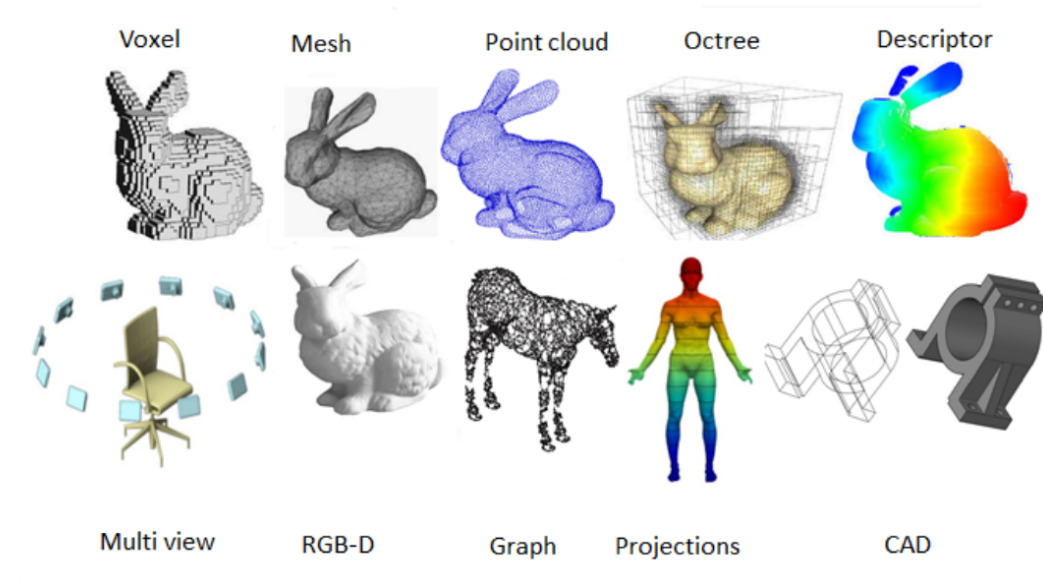


Figure 2.2: Examples of common 3D data representations include: Voxels model an object as 3D blocks on a rigid 3D grid; Meshes approximate surfaces using triangular surfaces; Pointclouds store discrete points in 3D space; Octrees are voxel-like but organized hierarchically in a tree data structure; Descriptors encode view-dependent shape information; Multi-view data consists of multiple calibrated cameras and their corresponding images; RGBD data combines continuous depth maps with RGB information; Graphs capture the geometric essence by linking different shape components; Projections map 3D points onto 2D planes; and CAD models describe objects using parametric or functional representations. [46]

Classical methods for 6-DoF pose estimation rely on hand-crafted features and geometric algorithms; however, recent advances are mainly driven by deep learning (DL). These modern techniques replace manual feature engineering with networks that learn to extract relevant information for pose estimation from various data modalities [47]. Kendall et al. [48] presented a robust real-time monocular 6-DoF camera localization system called PoseNet, based on Convolutional Neural Networks (CNN) [49]. PoseNet is capable of localizing high-level features and remains robust to challenging lighting conditions, motion blur, and various camera properties, where point-based SIFT registration normally fails. Within object-centric approaches, PoseCNN [50] has become a highly influential model for estimating 6-DoF poses of objects within a scene. However, the drawback of these methods and others [51] [52] is that they are trained on domain-specific datasets and are unable to generalize to other objects without retraining.

Training regression models, such as PoseNet or PoseCNN, on a comprehensive dataset of every existing object is infeasible due to the resulting data volume and the limitations of ML model architectures. A solution to this challenge is zero-shot object detection (ZSD), originally proposed by Bansal et al. [53], which leverages the semantic relationships between seen and unseen classes to detect the 2D position of a new object. In object

pose estimation, methods are typically categorized as instance-level or category-level [47]. Liu et al. [54] further differentiate approaches involving unseen objects, characterizing CAD model-based methods as falling within category-level; however, these methods still rely on instance-specific data and are therefore appropriately classified as instance-level. Instance-level methods require a specific object modality [47], whereas category-level methods generalize pose estimation for any object within a known category, such as "bottle" or "camera", without instance-specific object models [55]. When identifying a specific novel object is a priority, instance-level zero-shot implementations utilize features, templates, RGBD, pointclouds, and specialized modalities such as object masks, all of which can be generated from a mesh or CAD file. Consequently, detection pipelines employing 3D models benefit from a wide range of input data. This allows these methods to achieve high detection accuracy when precise 3D representations are available [56].

He et al. [57] presented a challenging few-shot open-set problem, introducing a dense prototype matching framework for 6D pose estimation of unknown objects from limited RGBD views. MegaPose [58] uses a render-and-compare strategy and a large-scale synthetic dataset to train a pose classifier and refinement network, highlighting the importance of data diversity for novel object generalization. Extending this model-based approach, ZS6D [5] implements a pre-trained ML model DINOv2 [59] as a zero-shot descriptor generator, matching rendered templates to query images for pose estimation without task-specific fine-tuning. Example DINOv2 deep features of an eagle and a plane are depicted in Figure 2.3 on the left side. Conceptually similar to ZS6D in its 2D-2D correspondence matching strategy, FreeZe [60] achieves higher accuracy through a 3D-3D matching approach; however, its implementation is not publicly available. Lin et al. extended this line of work with SAM-6D [8], which first uses a segmentation model [61] for robust novel object detection and segmentation, then applies a two-stage point matching model for pose estimation. These efforts converge in unified foundation models such as FoundationPose [9], which offers a single framework for the estimation and tracking of 6D poses of novel objects, supporting model-based and model-free scenarios, leveraging LLM-aided synthetic training and a neural implicit representation for performance gains. Controversially, the synthetic training data is subject to copyright issues, and the version trained without it has been released. This may result in a degraded accuracy, potentially reducing implementation success.

Motivated by these challenges, recent work has increasingly focused on "generate-then-refine" or "generate-then-align" paradigms, reducing the need for pre-existing 3D models and instead generating meshes at inference time. For example, HIPPo [62] and Any6D [63] leverage meshes derived from 3D reconstruction ML models at test time, employing either subsequent refinement or alignment to the real-world object. However, these approaches do not achieve higher accuracy than mesh-based systems such as SAM-6D [8] or FoundationPose [9]. Given robust mesh-based pose estimation pipelines, using pre-generated high-resolution meshes is still an effective and practical solution.

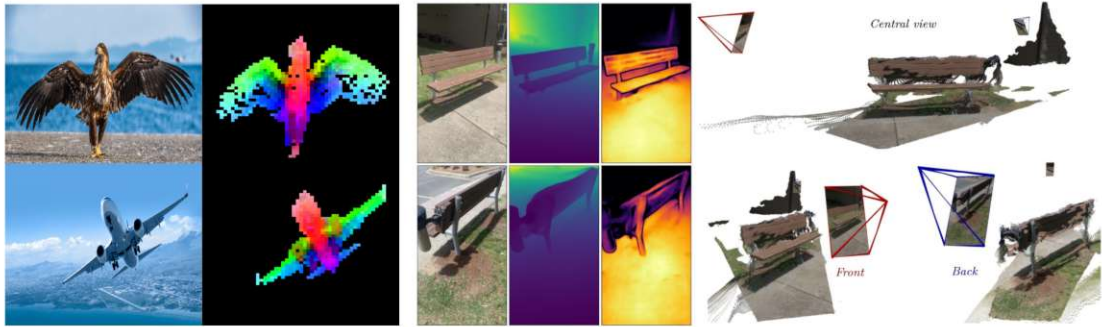


Figure 2.3: Extracted and processed features from foundation models; the left side depicts color-graded object features from DINOv2 [59]. The middle visualizes the features from DUST3R as a depth map [6], which are then used to register camera positions to generate a 3D pointcloud of a bench.

2.3 3D Reconstruction

Generating precise meshes for object pose estimation is resource-intensive, as high-quality methods require specialized hardware and environmental knowledge, such as the location of signal sources and sensor placements. Examples of such methods include Structure from Light, LiDAR, Laser Triangulation, and RGBD cameras. These techniques generate precise and dense pointclouds, which are then transformed into meshes using algorithms such as Poisson Reconstruction [21] or Marching Cubes [20]. Recent direct mesh generation models that leverage GenAI, such as Trellis [11] and Hunyuan3D-2 [64], offer a potential alternative to simplify the creation of 3D objects. Although Hunyuan3D-2 is slightly better, its computational demands are higher. These models are trained on extensive datasets of 3D objects to generate meshes directly from single-image input, bypassing intermediate processing steps. However, due to the generative nature of these models and the limitations of their training data, the resulting meshes do not precisely replicate the real objects, but provide a reasonable approximation.

Established 3D reconstruction software pipelines typically involve stepwise processing rather than a single-shot approach. An industry standard software solution is COLMAP [22], which employs SfM to register camera positions using an iterative algorithm, which allows the triangulation of SIFT features and therefore the generation of a pointcloud. Following an additional densification step and normal vector calculation, the pointcloud is then transformed into a high-quality mesh using Poisson Reconstruction. This approach performs best with feature-rich or large objects, such as buildings. However, the method suffers from lengthy processing times and relies on a good initialization for camera registration to ensure the successful completion of the pipeline. Although there is a successor, GLOMAP [65], that addresses processing time and achieves comparable or superior reconstruction quality, both methods rely on handcrafted features, which are increasingly being replaced by newer ML alternatives for camera registration [30].

The emergence of foundation models such as DINOv2 [59] and the zero-shot correspondence estimator CroCo [66] represents a shift towards more generalizable solutions

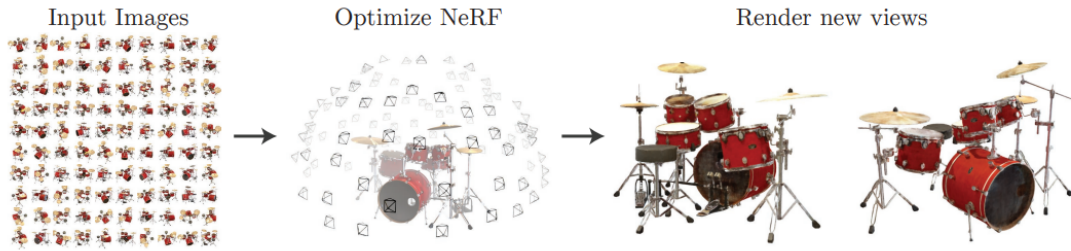


Figure 2.4: NeRF NVS reconstruction [70]. Input images and associated camera poses are optimized to generate novel views of a sparse scene.

for downstream tasks without requiring retraining. Both learned robust visual features facilitated by their training in an unsupervised fashion. Subsequently, the intrinsic 3D understanding of CroCo is utilized for depth perception, camera registration, and subsequent pointcloud creation, as seen in Figure 2.3 with images of a bench. Examples of its improved integrations include DUST3R [6], MAST3R [16], and MAST3R-SfM [30]. DL features that enable these capabilities are more robust than hand-made feature extractors [16], and therefore are increasingly used for feature matching, as demonstrated in ZS6D [5], CNOS [26] for CAD-based segmentation, or several spin-offs from MAST3R [67] [30]. Other approaches include specifically trained options such as VGGT [17] and VGGSfM [68], which facilitate camera registration and subsequent pointcloud creation. However, depending on the feature density of a scanned object, the resulting pointcloud varies in completeness. Currently, the MAST3R-SfM pipeline and the VGGT model generate the most viable initial pointclouds and camera positions, but may remain inconsistent for direct mesh reconstruction. When comparing the methods on the RealEstate10K benchmark [69], MAST3R-SfM and VGGT (Feed-Forward) score about the same. VGGT with Bundler Adjustment (BA) demonstrates SOTA performance; however, in the paper [17], it was mentioned that reconstruction performance drops under conditions involving extreme input rotations.

Building on an initial set of points and registered camera positions, NVS methods enable enhanced reconstruction of 3D scenes from sparse image inputs. In turn, the reconstructed scene can then be used to densify the pointcloud with the novel viewing angles. For example, Neural Radiance Fields (NeRF) [70] [71] represent a 3D scene as a continuous volumetric field learned by a multilayer perceptron (MLP), which, given a 3D coordinate and viewing direction, predicts color and density values. Figure 2.4 visualizes the NeRF NVS reconstruction; the input consists of a set of images with associated camera positions, which are then optimized through a training scheme with the MLP, enabling the generation of novel views. Despite its photorealistic reconstruction, the application is limited by its prolonged training time. Therefore, 3D Gaussian Splatting (3DGS) [72] was developed to accelerate scene reconstruction by representing scenes with 3D gaussians aligned through guiding input views. However, the inherent overlapping nature of 3D gaussians in all x-y-z directions hinders the extraction of consistently surface-aligned pointclouds. This issue has been addressed by 2DGS [10], which applies 2D gaussian surfaces placed in a 3D space, ensuring alignment with the surfaces of a scene. This results in dense, on-surface-aligned



Figure 2.5: Dataset example of Mip-NeRF 360 [75], DTU Robot Image Dataset [76] and MANiBOT [25].

points, where each point represents the midpoint of a gaussian surface. Alternatively, SVRaster [12] adopts a different approach, initiating reconstruction from registered camera frames and directly generating view-consistent voxels, subsequently fused into a mesh. Sun et al. [12] also presented comparative evaluations of NVS methods, demonstrating a favorable balance between accuracy and inference time for both SVRaster and 2DGS.

Integrated loss functions go beyond aligning radiance fields, gaussians, or voxels for accurate rendering; they also leverage alignment to object surfaces guided by additional modalities, such as sparse pointclouds, depth maps, normal maps, or segmentation masks. For example, mask modalities are extracted using specialized methods such as the Segment Anything Model (SAM) [61], Grounded SAM [29], or Grounding DINO [73]. After applying an NVS method and extracting a dense pointcloud, the post-processing to generate a mesh typically involves Truncated Distance Functions (TDF) [74] followed by standard meshing algorithms. Although mesh texturing is increasingly performed using novel diffusion models [64], texture registration from source images remains a well-established alternative. However, all data sources are derived solely from images; therefore, NVS remains sensitive to lighting conditions [10]. Specular reflections and specular light frequently introduce artifacts during meshing [10].

2.4 Metric Scale Estimation

Accurate metric scaling of objects is a prerequisite for robust integration into a pose estimation pipeline. Cameras and photogrammetry alone provide an incomplete representation of the 3D world, lacking metric scale. Hardware-based depth perception offers a readily available solution; however, it introduces additional costs. Although resolution can decrease around edges, potentially introducing artifacts [77], it is accurate for scaling purposes. Alternatively, MAST3R, trained on metric data [16], is able to estimate scene scale; however, its performance is validated primarily for large-scale environments and has received limited testing in small-scale scenarios. Other ML models, such as monocular depth estimation, offer continuous depth perception, but typically generate only disparity maps [78]. More novel monocular methods, such as metric DepthAnythingv2 [78], and Depth-Pro [18], attempt to refine depth by estimating camera intrinsic and with metric

training data. Combined approaches leverage GT sparse depth data from a low-cost LiDAR to fuse the strengths of both modalities [79]. FoundationStereo [80] similarly integrates a low-cost stereo vision setup with ML-based depth estimation for a more robust solution. The emerging trend favors consistent models that improve frame-to-frame depth coherence in video [81]. Depth maps not only facilitate model scaling, but also help alignment in NVS. Importantly, depth ML models are rarely tested on near-view images of objects, as the scale is often ambiguous in such scenarios and may therefore be unusable for downstream object scaling.

2.5 Data Sources for 3D Reconstruction

Although originally designed for distinct tasks, many existing benchmarks are based on object-centric 3D datasets, making them suitable for comparing meshes generated by a wide range of methods. NVS methods generally focus on large or unbounded scene datasets, such as Mip-NeRF 360 [75], ScanNet++ [82], or LLFF [83] for scene reconstruction, and Tanks and Temples [84] is often used for subsequently evaluating the meshing. Small object-centric scenes are primarily represented with the DTU Robot Image Dataset [76]. Alternatively, CO3D [85] and OmniObject3D [86] offer datasets focused on common household items, while the project-specific MANiBOT [25] dataset includes supermarket objects. An example of Mip-NeRF 360, DTU Robot Image Dataset, and MANiBOT is depicted in Figure 2.5. For subsequent evaluation of pose estimation, the de facto benchmark is BOP [31], which offers both 3D object models, as illustrated in Figure 3.3, for reconstruction, as well as real-world counterpart objects from its YCB-V subset, thereby constituting a comprehensive resource.

3 Methods

This chapter details the necessary components for 3D reconstruction of real-world objects and presents subsequent methods for zero-shot object pose estimation in unconstrained environments. Beginning with a description of the 3D data modalities in Section 3.1, the representation of the virtual object is dissected into its important attributes. Section 3.2 then introduces the object datasets used in this thesis and the metrics for evaluation within the BOP benchmark. Turning to real-world data acquisition, Section 3.3 describes how to correctly capture data from supermarket objects. Following this, Camera Registration and Pointcloud Generation 3.4 describes methods for converting generated 2D data into 3D data. The process of using these 2D and 3D data to generate a mesh with NVS is then described in Section 3.5. Since the meshes still have an undefined scale at this point, Section 3.6 describes the scaling procedure developed specifically for this thesis. Methods for inferring object pose are subsequently listed in Section 3.7. Finally, the last two sections, Robotic Platform 3.8 and Mesh Generation Pipeline 3.9, detail the thesis-specific implementation.

3.1 3D Data Modalities

Given an unordered collection of images of a scene, each provides 2D RGB information from a distinct viewpoint. By combining these camera perspectives and estimating their 3D relationships through feature matching, a 3D scene reconstruction is achieved. These 3D features can then be described and visualized as a pointcloud or as a combined mesh. Figure 3.1 shows a zoomed-in top view of the corresponding mustard object from Figure 1.2. The left side illustrates the extracted pointcloud, and the right side shows the resulting mesh. These representations are fundamental in robotic vision to describe objects and their poses in 3D environments.

3.1.1 Pointcloud

Geometric representation relies on fundamental elements such as points, lines, and surfaces, each defining a spatial characteristic. A pointcloud utilizes the condition that lines and surfaces are describable with points, by accepting loss of intrinsic information, such as line direction or surface normal vectors. As shown in Figure 3.1, transforming the mesh into a pointcloud results in a loss of information. Consequently, the raw representation of the pointcloud data is a collection of points, each with x , y , and z coordinates. Additional attributes, such as normal vectors or RGB values, are not part of a pointcloud. Furthermore, points can be described within a fixed, discrete 3D grid, in which they are commonly referred to as voxels. When a pointcloud is not generated by down-sampling a mesh, it is typically generated using the camera intrinsic matrix and a depth map.

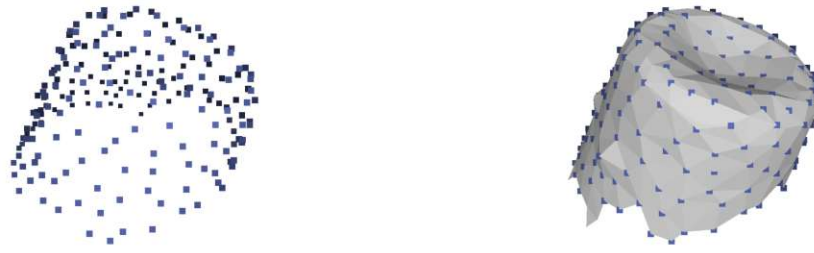


Figure 3.1: Extracted pointcloud and mesh depicting the cap of the mustard bottle from Figure 1.2.

3.1.2 Mesh

Compared to a pointcloud, a mesh provides a superior representation of a 3D object through the explicit incorporation of edges and faces, which define its surface and result in a more complex data structure. This structure transforms a collection of unconnected points into a network of vertices. As illustrated in Figure 3.1, the vertices can be interpreted as corners (corresponding to the blue points), and the faces are the surfaces defined by the edges that connect them. This structure is comparable to a chain of connected triangles. The density of the mesh determines the smoothness of the resulting object surface, but higher densities introduce greater computational demands. Faces also have a perpendicular normal vector, the direction of which is used to render shading, and can hold simple color information. Detailed texture information is commonly represented as a UV map. It represents the 3D mesh unfolded onto a 2D plane, ensuring that each face corresponds to a unique coordinate without distortion. It allows mapping of high-resolution image data onto the faces, resulting in a better representation than only saving one RGB value per face. If registered camera positions are available for the mesh, UV maps can be generated by triangulating the camera RGB values.

3.1.3 Object Pose

The pose of an object in 3D space is defined by its translation vector and rotation matrix, referenced to both an object-centric and a world coordinate system. Establishing a suitable object-centric coordinate system depends on the specific application. Common choices include alignment along a symmetry axis, the geometric center, or the center of gravity. For this thesis, focusing primarily on supermarket objects, the coordinate system is defined as follows: the z-axis is aligned upwards away from the base surface upon which it rests, the x-axis is defined in relation to the primary viewing direction of the object within a supermarket shelf display, and the y-axis is defined by adhering to the cartesian coordinates system, oriented perpendicular to both the z-axis and the x-axis.



Figure 3.2: Test images from the BOP YCB-V dataset [31].

3.2 Benchmark and Datasets

A project that encompasses data aggregation and real-world implementation requires a diverse dataset for comprehensive testing, utilizing both synthetic and real data. Given the importance of robust meshing, scaling, texturing, and object pose estimation, particularly when leveraging high-quality 3D meshes of real-world objects, access to physical objects is invaluable. As TU Wien has objects from the BOP dataset, a leading benchmark for pose estimation in the CV community, this dataset presents a natural choice. Furthermore, the MANiBOT project generated a dataset of supermarket objects with corresponding real-world representations, allowing testing of real-world challenges.

3.2.1 Benchmark for 6D Object Pose Estimation

The stated mission of BOP is "The goal of BOP is to capture the state of the art in 6-DoF object pose estimation and related tasks such as 2D object detection and segmentation. An accurate, fast, robust, scalable, and easy-to-train method that solves this task will have a big impact in application fields such as robotics or augmented reality." [31]. Since 2018, BOP has been an ongoing competition to find the best model-free or model-based object detection, segmentation, and pose estimation on seen or unseen objects. CNOS, SAM-6D, and FoundationPose were evaluated on model-based 6-DoF object pose estimation on unseen objects, using the 3D objects of BOP [26] [8] [9]. Consequently, these evaluations serve as a robust baseline for comparing the performance of meshes generated from BOP objects with NVS methods.

The BOP dataset comprises three groups, with the BOP-Classic group, specifically the BOP-Classic-Core YCB-V [50] dataset, being the primary focus of this work. This dataset contains RGBD images of object instances annotated with labels, 6D poses, 2D bounding boxes, and 2D binary masks. The test images are captured from real-world scenes, depicted in Figure 3.2, while the training images are generated from real data or synthetically using BlenderProc [28]. Figure 3.3 illustrates the 21 YCB-V 3D models representing a diverse set of household and supermarket objects.



Figure 3.3: The BOP subset YCB-V [31] consists of 21 everyday objects typically found in households and supermarkets. The objects are numbered sequentially, beginning with the first object in the upper-left corner and proceeding from left to right and from top to bottom. Under this scheme, the object located in the lower-right corner corresponds to 21.

Training, testing, and evaluation of this dataset are defined as follows. For 2D segmentation of unseen objects, methods are trained on images with object-specific 2D binary masks and 3D meshes. At test time, they take unseen RGBD images depicting an arbitrary number of objects and predict a list of object segmentations with confidence values. The accuracy of a method is then evaluated using an Average Precision (AP) score. To compute the score per-object AP_O , the precision of the intersection-over-union between the predicted and GT masks is averaged over multiple thresholds. Finally, correct predictions with less than 10% visibility are filtered, and the AP per dataset is defined as AP_D [24].

The 6D pose estimation methods are trained on RGBD images with annotated GT 6D poses and 3D meshes. During training, the method learns to estimate the rigid transformation matrix $P = [R|t]$ from the coordinate system of the object to the camera coordinate system. At test time, the method receives unseen RGBD images and predicts a list of pose estimates $E = [E_1, \dots, E_m]$ with confidences for each object instance. The



Figure 3.4: Subset of the supermarket object dataset from the MANiBOT project [25], featuring semi-transparent items, diverse geometries, and deformable properties.

accuracy is evaluated using Average Recall (AR), which is calculated based on three pose-error functions (see Section 2.2 in [24]): VSD (Visible Surface Discrepancy), Maximum Symmetry-Aware Surface Distance (MSSD), and Maximum Symmetry-Aware Projection Distance (MSPD). Recall is the fraction of annotated object instances for which a correct pose is estimated. For each pose-error function e , a pose is considered correct if $e < \theta_e$, where $e \in \{VSD, MSSD, MSPD\}$, the AR_e is calculated by averaging recall rates at multiple thresholds θ_e and misalignment tolerances τ of VSD. Finally, the overall accuracy per dataset D , AR_D , is defined as [24]:

$$AR_D = \frac{AR_{VSD} + AR_{MSSD} + AR_{MSPD}}{3}$$

3.2.2 MANiBOT

The supermarket objects in the YCB-V dataset generally feature simple geometries, such as cylinders or box-shaped items, with prominent text. To evaluate reconstruction quality and robotic manipulation performance on more complex, realistic shapes, a more diverse set of objects is required. Therefore, this thesis incorporates a subset of objects from the MANiBOT project. As shown in Figure 3.4, these objects additionally depict semi-transparent, deformable geometries with various shapes and fine detailed inscriptions, offering greater complexity compared to the YCB-V supermarket objects.

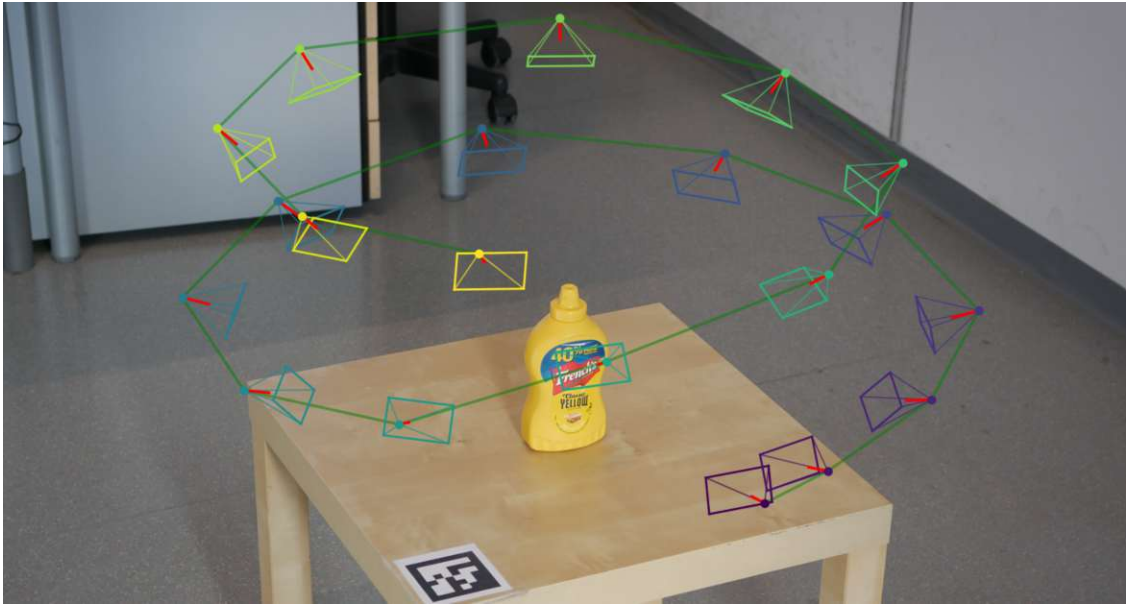


Figure 3.5: Camera trajectory circling an object two times to generate a diverse set of camera poses.

3.3 Data Acquisition

Technical systems rely on data as input and are, consequently, affected by its quality. For instance, sharp images contain more discernible features than blurred images, leading to improved output. Therefore, a robust data capture strategy enhances data quality and promotes process repeatability. Specifically, in unconstrained environments, adhering to defined guidelines introduces necessary structure. However, the following sections should be understood as presenting structured approaches for data acquisition to facilitate reproducible testing, rather than as rigid protocols. Section 3.3.1 details the data capture procedures employed in this thesis, followed by a description of the post-processing segmentation approach.

3.3.1 Imaging Setup

To acquire a diverse image set, a camera trajectory should encircle the object at varying heights, as shown in Figure 3.5, to ensure comprehensive coverage of the surfaces. Additionally, the trajectory should maintain focus on the center of the object to guarantee a robust center point initialization of the object pose. Beginning with the initial image, it needs to be captured from the front of the object, as this establishes the coordinate axis. Specifically, the x-axis is aligned perpendicular to the camera sensor, the z-axis points upward, and the y-axis extends to the right.

Taking into account a realistic environment, as exemplified in Figure 3.5, the object is placed on a surface and the trajectory of the camera follows a continuous smooth path that combines the view of the middle and top during video capture. Recording a video speeds up data capture, ensures consistent lighting changes between images, and the

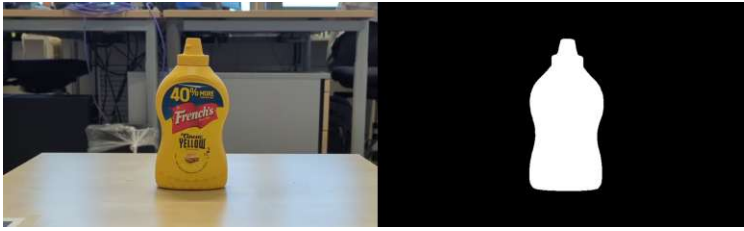


Figure 3.6: The first frame of the object video is segmented using Grounded-SAM with the prompt "object in the middle." This generates an initial segmentation mask that starts the mask-tracking process, which is then used to remove the scene background from the object.

number of extracted image frames can be adjusted later on. Importantly, when capturing the object solely from a top-down perspective, the resulting mesh will exhibit a hole at the bottom that cannot be filled correctly without additional information. A potential workaround involves incorporating a bottom view, feasible only if the object is stable in an inverted position, or manually rotating the object during filming. However, this approach results in camera registration failure with the entire scene, necessitating the use of segmented object extractions. Furthermore, image capture could be performed under diffused light to minimize the impact of shadows or reflections. Failure to address these lighting conditions hinders camera registration and may result in artifacts within the mesh generation process.

3.3.2 Segmentation

Following image capture, the object must be extracted from the scene to remove the background, allowing 3D object reconstruction. Consequently, a zero-shot segmentation model, Grounded-SAM [29], generates a segmentation mask that is used to differentiate between the scene and the object, as demonstrated in Figure 3.6. Grounded-SAM leverages video input to track object masks across all frames, with the initial mask defined in the first frame by the prompt "object in the middle.". This open-vocabulary segmentation approach accurately captures masks for diverse objects.

3.4 Camera Registration and Pointcloud Generation

Starting from an unordered collection of scene images without known camera intrinsics or extrinsics, it is possible to estimate these camera parameters and subsequently reconstruct 3D scene geometry, represented as a pointcloud, by performing image feature detection and matching. Established methods, including COLMAP and MAST3R-SfM, leverage SfM, Multi-View Stereo (MVS), and BA. MAST3R-SfM specifically extends this approach with a learning-based MVS technique derived from MAST3R [30]. Although these methods are effective, recent SOTA approaches, such as VGGT, based on a feed-forward neural network performing 3D reconstruction from an arbitrary number of input views, demonstrate superior performance [17]. The following sections elaborate on important pipeline stages and the usage of each method.

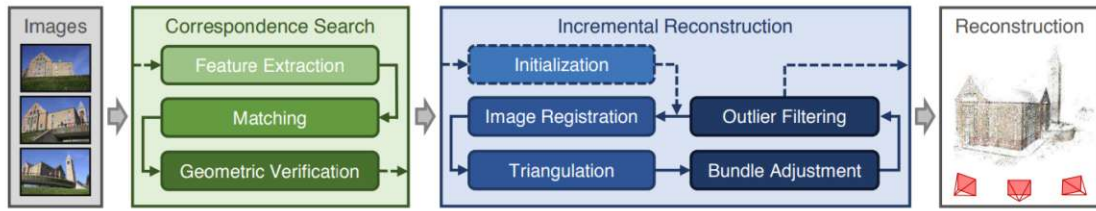


Figure 3.7: The COLMAP SfM pipeline consists of two primary stages: Correspondence Search and Incremental Reconstruction. Initially, SIFT features are detected and matched across images and subsequently validated using projective geometric constraints. Then, an initial image pair is selected to start the scene reconstruction. Additional images are iteratively registered using PnP, new 3D points are triangulated, and BA refines the reconstruction by minimizing an error term, after which outliers are removed. This procedure is repeated until all images have been successfully registered to the reconstructed scene. [22]

3.4.1 COLMAP and Structure from Motion

COLMAP represents the most popular framework based on traditional SfM methods, positioning it as the go-to industry standard. As illustrated in Figure 3.7, its incremental SfM pipeline consists of two main parts: a Correspondence Search applied to an arbitrary number of images, followed by an Incremental Reconstruction step generating a sparse pointcloud with camera poses [22]. The initial stage aims to identify geometrically verified image pairs and construct a graph of image projections for each point. Specifically, SIFT features are employed as appearance descriptors, matched to corresponding scene parts, and subsequently geometrically verified by estimating a transformation that maps feature points between images using projective geometry [22].

The second stage takes the verified scene graph as input and outputs a reconstructed scene consisting of 3D pose estimations for the cameras and a sparse pointcloud. It begins with the critical step of initializing an image pair from a preferred dense seed location. New images are then registered to the current model using PnP to estimate both poses and intrinsic parameters, leveraging already triangulated points. Following image registration, new points are triangulated to extend the model. Finally, BA is employed as a core refinement step to mitigate accumulated error and prevent a non-recoverable state. Specifically, the reprojection error between predicted and observed image features is minimized by utilizing a non-linear least-squares algorithm. Points that do not fit the model are removed after BA, and this incremental reconstruction process is repeated until a stable state is reached, that is, until all registerable images have been registered. Finally, the pointcloud can be densified using multi-view stereo to generate new points and normal vectors, aiding subsequent meshing with Poisson Reconstruction.

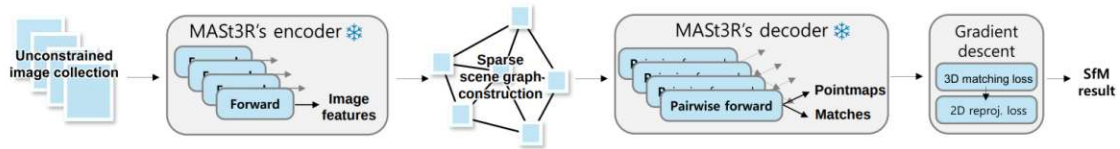


Figure 3.8: Overview of the MAST3R-SfM pipeline, which integrates MAST3R and COLMAP to improve SfM. The MAST3R encoder is employed as a feature retrieval backbone to construct a sparse scene graph from an image set. Next, the MAST3R decoder performs pairwise matching to produce a set of point maps. These outputs are then fed into the COLMAP pipeline to estimate camera poses and reconstruct a pointcloud. [30]

3.4.2 MAST3R and Multi-View Stereo

Unlike COLMAP, which relies on handcrafted features, MAST3R [16] employs a DL-based MVS approach to estimate camera poses, camera intrinsics, and metric pointclouds, thus reducing the complexity of the reconstruction pipeline. MAST3R adapts the DUST3R 3D point map regression matching methodology [6] and incorporates a second head to produce dense local features, which enhances matching performance while maintaining robustness even with extreme viewpoint disparities. Using the underlying siamese ViT architecture of CroCo [66], the system facilitates information sharing between two images, regresses metric point maps and confidence scores, and learns local features. Subsequently, at inference time, fast nearest-neighbor search, utilizing sub-sampling, is applied for both geometrical and feature-based reciprocal matching.

Robust 2D–2D descriptor learning enables relative camera pose estimation, while direct point map regression predicts metric depth, resolving scale ambiguity, and producing detailed 3D reconstructions. The intrinsics of the camera are implicitly encoded and can be recovered from the predicted 3D geometry, allowing the method to serve as a standalone camera calibration solution. However, as an MVS approach operating on image pairs, its processing time grows quadratically with scene complexity.

3.4.3 MAST3R-SfM

Building on the foundation laid by MAST3R and aiming to substantially streamline SfM approaches, such as COLMAP, MAST3R-SfM [30] introduced a new framework that is simpler, more scalable, faster, and capable of processing unconstrained image sets. It has been demonstrated that this method is capable of registering scenes from purely rotational image sequences without actual motion, and it operates without RANSAC [87], two characteristics that are typically required for standard SfM methods [30]. Unlike MAST3R, MAST3R-SfM provides a memory-efficient strategy to accurately align local reconstructions and serves as an effective image retrieval component within SfM, reducing computational complexity from quadratic to linear. This is accomplished via a two-stage cascade of image matching: a rapid initial comparison, usually relying on global image descriptor similarity, first selects candidate image pairs, which are then passed to a more computationally intensive keypoint matching stage if they appear sufficiently similar. Using the frozen

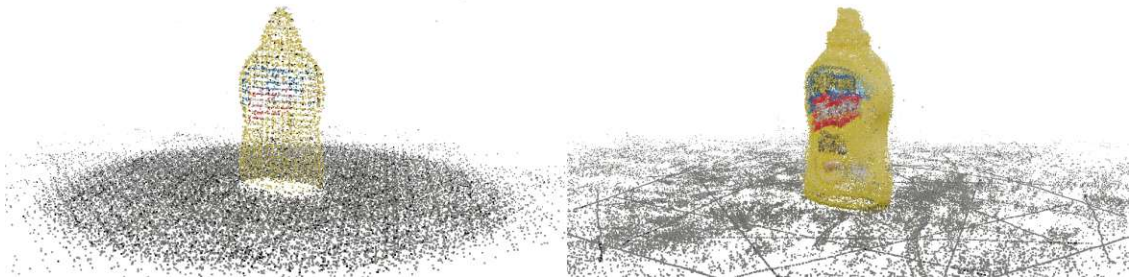


Figure 3.9: Visual comparison of an object-centric pointcloud generated by MAST3R-SfM (left) and VGGT (right).

encoder of MAST3R, the token features are directly applied as local features, which are subsequently used for the retrieval using Aggregated Selective Match Kernels (ASMK) [88].

As depicted in Figure 3.8, MAST3R-SfM initiates the process by registering an unconstrained image collection and constructing a sparse scene graph. The pipeline leverages the MAST3R encoder as a feature retrieval model to identify dominant feature matches, avoiding the need to match every single image feature with each other, resulting in faster processing times compared to COLMAP. Subsequently, pairwise processing is performed using the MAST3R decoder to infer point maps between the identified matches. Finally, COLMAP SfM employs gradient descent to calculate camera poses and reconstruct a pointcloud from the images. Notably, this turns the method into a hybrid of MAST3R and COLMAP, aiming to substitute the correspondence search that COLMAP usually performs in order to boost performance. However, a limitation is that the method is not fully standalone and depends on a separate installation of the COLMAP codebase.

3.4.4 Visual Geometry Grounded Transformer (VGGT)

In contrast to MAST3R, which operates on pairwise comparisons, VGGT simultaneously processes multiple images and jointly estimates camera poses, point maps, depth maps, and 3D point tracks [17]. As a result, this feedforward network is substantially faster than comparable SOTA systems such as MAST3R-SfM; inference typically takes less than one second, allowing real-time use. Conceptually, VGGT aligns with the contemporary large-scale AI paradigm: employ a simple, scalable ViT-based architecture and learn 3D geometry directly from big data without relying on intricate algorithmic components. However, it has two main limitations relative to MAST3R-SfM: processing all images simultaneously leads to significantly higher GPU memory consumption, and reconstruction accuracy deteriorates under strong input rotations [17]. A VGGT variant augmented with BA is also available, which substantially surpasses current SOTA, but only under modest viewpoint changes. A visual comparison with MAST3R-SfM is presented in Figure 3.9. The left illustrates the object-centric pointcloud generated by MAST3R-SfM, whereas the right depicts the corresponding reconstruction produced by VGGT.

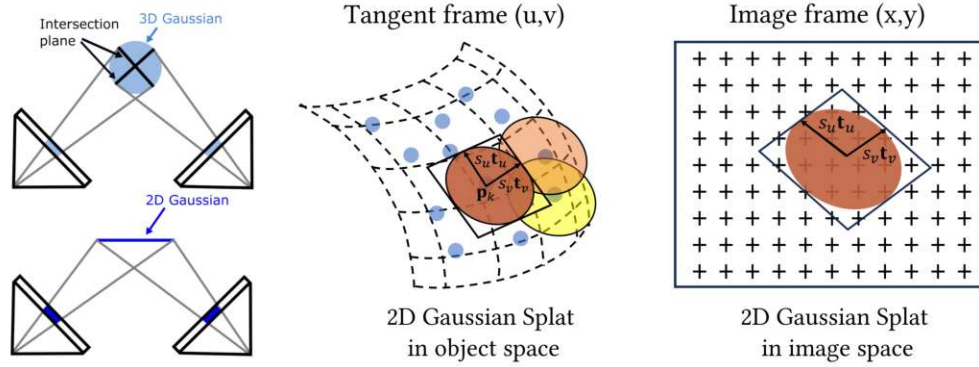


Figure 3.10: Left: Illustration of how the 2D gaussian surface ensures view consistency between registered image planes compared to 3DGS. Right: Detailed visualization of the elliptical disk in both the tangent and image frames, parameterized by its center point \mathbf{p}_k , tangential vectors \mathbf{t}_u and \mathbf{t}_v , and scaling factors s_u and s_v . [10]

3.5 Novel View Synthesis for Mesh Generation

Established NVS methods typically start with registered cameras as the basis for applying ML techniques to represent a scene in a specific data modality. When supported, corresponding pointclouds can further improve scene reconstruction. For example, NeRF characterizes a scene by mapping the 3D position and 2D viewing direction to color and density [70], while 3DGS generates 3D gaussians within 3D space [72]. This data can subsequently be employed to generate a mesh that represents the reconstructed scene. However, while NeRF produces view-consistent geometry, it requires substantial training time. Compared to this, 3DGS employs a parallelizable rasterization approach to reduce training time, but lacks view-consistent geometry [72]. SOTA NVS methods, including 2DGS and SVRaster, address this trade-off by mitigating training time while maintaining view-consistent geometry, rendering them suitable for a rapid and reliable mesh generation pipeline. Finally, a growing number of GenAI image-to-mesh models, such as Trellis, skip these stages altogether, along with the associated preprocessing steps [64] [11].

3.5.1 2D Gaussian Splatting

As noted previously, 3DGS does not generate view-consistent geometry, limiting its effectiveness as a densification step prior to mesh extraction. To mitigate this, 2DGS reduces the 3D gaussians to a set of 2D oriented planar gaussian disks as seen on the left side of Figure 3.10. These 2D disks exhibit improved surface alignment because, unlike 3DGS, their intersection planes are view-consistent with rays originating from the camera plane, thereby reducing the surface overlapping observed with 3DGS. This approach yields detailed geometry reconstruction while maintaining fast training speeds, leveraging the 3DGS rasterization technique. Furthermore, to refine alignment with thin structures, two regularization terms, one penalizing depth distortion and the other encouraging normal consistency, were incorporated.

The elliptical gaussian disk is visualized in Figure 3.10 in a tangent frame and image frame, described with a center point \mathbf{p}_k , two tangential vectors \mathbf{t}_u and \mathbf{t}_v , and two scaling terms s_u and s_v . Since the disk is defined in a local tangent plane in world space, it is parameterized as:

$$P(u,v) = \mathbf{p}_k + s_u \mathbf{t}_u u + s_v \mathbf{t}_v v = \mathbf{H}(u,v,1,1)^T \quad (3.1)$$

with $\mathbf{H} \in 4 \times 4$ being a homogeneous transformation matrix. The 2D gaussian value of a point $\mathbf{g} = (u,v)$ is then evaluated by a standard Gaussian:

$$\mathcal{G}(\mathbf{g}) = \exp\left(-\frac{u^2 + v^2}{2}\right) \quad (3.2)$$

The learnable parameters are the center \mathbf{p}_k , scaling (s_u, s_v) , and rotation $(\mathbf{t}_u, \mathbf{t}_v)$, additionally each gaussian has an opacity α and a view-dependent appearance c parameter for volumetric alpha blending. This allows the scene to be represented by stacking overlapping semi-transparent disk layers, while filtering out disks with low opacity, which do not contribute to the final reconstruction.

Considering \mathbf{x} being a homogeneous ray emitted from the camera passing through pixel coordinates (x,y) and intersecting with a gaussian disk at depth z , \mathbf{x} is defined as:

$$\mathbf{x} = \begin{pmatrix} xz, yz, z, z \end{pmatrix}^T = \mathbf{W}P(u,v) = \mathbf{W}\mathbf{H}(u,v,1,1)^T \quad (3.3)$$

With $\mathbf{W} \in 4 \times 4$ being a transformation matrix from world space to screen space to model ray-splat intersections to generate view-consistency between the gaussian scene and the scene images. To filter out noise and improve geometric modeling 2DGS applies two additional loss functions in the training process. One that mitigates depth distortion by minimizing the distance between the ray-splat intersections:

$$\mathcal{L}_d = \sum_{i,j} \omega_i \omega_j |z_i - z_j| \quad (3.4)$$

with ω_i as blending weight and z_i the depth of the intersection points. Followed by a normal consistency loss function:

$$\mathcal{L}_n = \sum_i \omega_i (1 - \mathbf{n}_i^T \mathbf{N}) \quad (3.5)$$

with \mathbf{n}_i denoting the normal of the disk towards the camera and \mathbf{N} the normal estimated by a calculated depth map. The final loss is described by adding the two 2DGS introduced losses with the RGB reconstruction loss \mathcal{L}_c of 3DGS [72] as:

$$\mathcal{L} = \mathcal{L}_c + \mathcal{L}_d + \mathcal{L}_n \quad (3.6)$$

While training, the Gaussian disk can separate into smaller gaussian disks, which leads to a denser 3D reconstruction than the initial pointcloud. The mesh is then extracted using the rendered depth maps of the training views to project the depth values onto the image pixels. Additionally, a Truncated Signed Distance Function (TSDF) is used to fuse the depth maps of the reconstruction.

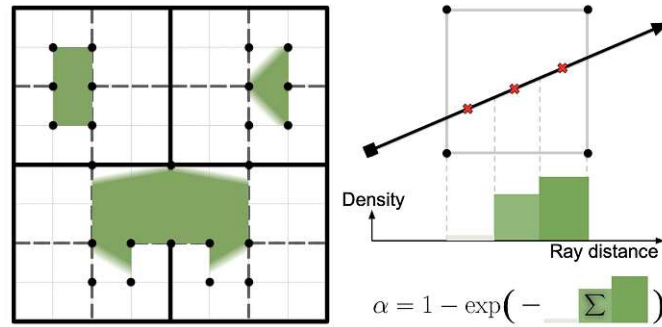


Figure 3.11: The left side illustrates the octree data structure of SVRaster to ensure correct rendering order and adaptive detail representation, where each voxel holds its own density values. These densities are trilinearly interpolated inside the voxels, based on shared corner grid points, and are used to compute the opacity value α through volume integration of K samples along the ray–voxel intersections. [12]

3.5.2 Sparse Voxels Rasterization

Representing a scene with 2D Gaussian disks does not yield a well-defined grid pattern, rendering surface reconstruction non-trivial. Nevertheless, the rendering speed outperforms previous grid-based rasterization methods [12]. To decrease the rendering time of grid-based approaches and make them competitive with 2DGS, SVRaster adaptively and explicitly allocates sparse voxels to represent varying levels of detail within a scene. This results in a method that trains and renders efficiently while preserving NVS quality comparable to the SOTA.

To facilitate correct rendering order and adaptive detail representation, SVRaster voxels are described using an octree data structure. Figure 3.11 visualizes this structure on the left, with each voxel possessing its own density values. The density is trilinearly varied within the voxels, defined by shared corner grid points, and used for α computation via volume integration of K samples along the intersections of rays. As in NeRF [70] and 2DGS [10], α is used for alpha composition to render the color of an image pixel. Crucially, the summation of the α function incorporates an activation function to prevent negative density values. The view-dependent appearance is modeled with spherical harmonic coefficients per voxel.

Scene optimization begins by initializing the octree grid and setting all parameters to constant values. Subsequently, as in 2DGS [10], rays emitted from camera pixels optimize the scene based on a loss function \mathcal{L} to generate view-consistent geometry. This includes, for mesh extraction, the depth-normal loss 3.5. Because the initial grid only coarsely captures the scene, voxels are pruned and subdivided every h training iteration. Pruning discards low-opacity voxels, and further subdividing them is hypothesized to especially benefit those with larger training loss gradients [12]. Thereafter, the spherical harmonics and the grid point density are updated. Once training is complete, the resulting voxel structure directly supports the Marching Cubes algorithm for mesh generation.

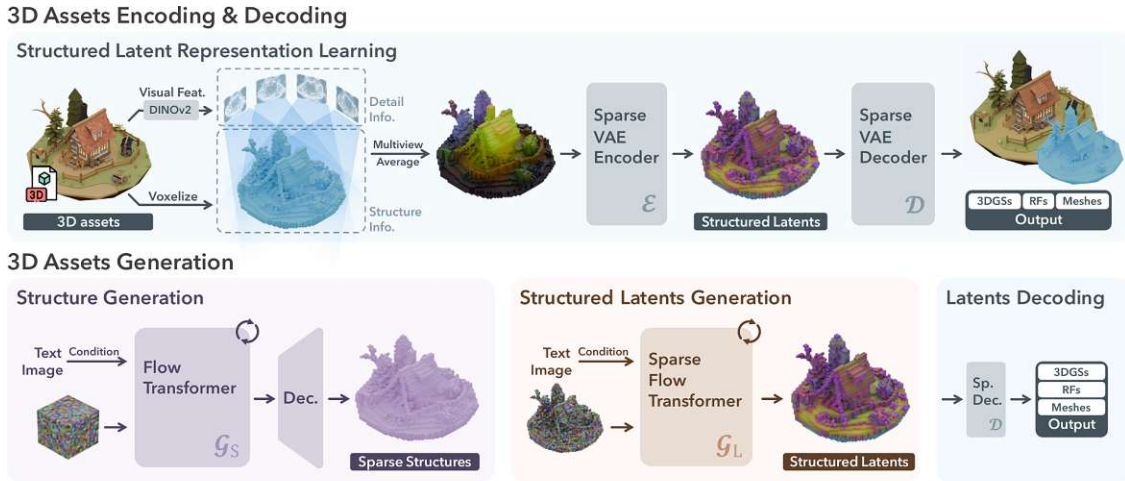


Figure 3.12: Overview on how Trellis infers a mesh from an image. First, a SLAT representation is learned from a voxelized 3D asset and its DINOv2 features, inhibiting both geometry and appearance information, to decode this representation into different 3D modalities. Followed by two-step 3D asset generation utilizing 3D noise and image conditions as an input to generate a sparse structure, which is then similarly refined with local latents, to use the trained decoder on the generated SLAT to decode it into a mesh. [11]

3.5.3 Generative 3D Reconstruction (Trellis)

Registering cameras, deriving a pointcloud, densifying the sparse representation with an NVS method, and subsequent meshing generate high-quality meshes; however, this workflow involves multiple stages. All of these stages can be bypassed by SOTA generative 3D reconstruction methods such as Hunyuan3D-2 [64] or Trellis [11]. Among these, Trellis offers a comparatively simpler local deployment than Hunyuan3D-2. Trellis builds on Structured Latent (SLAT) representations to decode encoded image data directly into 3DGS, radiance fields, or meshes. Figure 3.12 illustrates how Trellis learned SLAT representation by training a sparse Variable Auto Encoder (VAE) for the subsequent generation of 3D assets. Similarly to several other ML methods [26] [29], Trellis starts from DINOv2 deep features computed from multi-view renderings of 3D assets and projects their averaged features into a voxel grid representation of the corresponding assets. This representation is then used to train the encoder and decoder of the VAE. As its name indicates, the VAE learns the intermediate SLAT state in an unsupervised manner by decoding this latent representation into various 3D modalities.

As illustrated in the lower part of Figure 3.12, the 3D asset generation pipeline is divided into two distinct stages: Structure Generation and SLAT Generation. Both stages employ a transformer architecture, conditioned on images or text, and incorporate an additional 3D noise input. The initial stage generates a sparse structure from unstructured noise and its condition; this structure is subsequently used as a structured noise input for the second stage, allowing the generation of SLAT. Finally, the SLAT is decoded into the desired 3D modality using the VAE decoder. Collectively, this system represents an SOTA

image-to-mesh generator, capable of producing textured, high-quality meshes in seconds [11]. However, due to its reliance on GenAI, the system does not consistently produce accurate results at all viewing angles and generates meshes scaled to 1 m. An illustrative example is provided in Figure 1.2.

3.6 Metric Height Estimation

Generated meshes from NVS methods lack a defined scale. To enable their application in robotic manipulation, a scaling process is therefore necessary. The exclusion of hardware-based depth estimation, which physically measures the environment, leaves only the option of comparing known object scales within a scene to understand the overall scene scale. However, without a known object for reference, accurate scale estimation becomes ambiguous. A potential workaround involves using ML models trained on a metric scale, as these models retain implicit scale information within their parameters.

Preliminary testing focused on extracting scale from the pointcloud generated by MAST3R/MASt3R-SfM or utilizing registered images in combination with depth maps created by Depth-Pro [18]. However, both methods revealed ambiguity in scene scale when analyzing near-view images of objects. Taking into account this drawback, a consistent method such as Video-Depth-Anything [81] was applied, combining several video frames to maintain scale stability for near-view analysis. However, this again yielded inconsistent depth estimation results.

After a reevaluation of MAST3R, the solution involved utilizing its multiview 3D reconstruction to integrate not only near-view images but also an additional sample of far-view images from the scanning scene. Given that the scale of each depth estimation between two images and their registration with other pairs is an optimization parameter [16], allowing the 3D reconstruction to generate a matching pointcloud, the scene images need to exceed the number of near-view object images. A good compromise was found by registering four scene and two object images to each other, as shown in Figure 3.13, to generate a pointcloud depicting the full scene and the front side of the object registered into the scene. Two object images are the minimum required, as MAST3R builds upon a stereo matching backbone. This approach maintains metric scaling and is therefore used to scale real objects, without relying on specialized hardware, by extracting the object-specific points. Importantly, the scanning scene images should not be captured with the object present, as the object-specific pointcloud is registrable using only two images per scanned object. This yields the advantage that the scene images only need to be captured once per scanning scene environment.

Estimating the height of an object is achieved by projecting the pointcloud onto the first camera plane of the object video. The object mask is then applied to retain only the points that describe the object, as shown in Figure 3.13. Subsequently, the height is estimated by subtracting the lowest point coordinates from the upper bound.

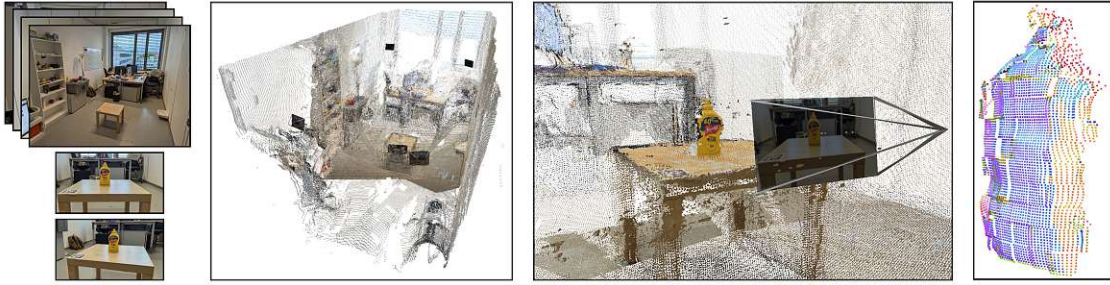


Figure 3.13: To generate an accurate metric scene pointcloud, four scene images (excluding the object) are registered with two images of the object using MAST3R. Subsequently, the first registered image and its corresponding object mask are utilized to extract object-specific points, which are then employed to estimate the height of the scanned object.

3.7 Object Segmentation and Pose Estimation

The process of estimating the pose of a known object can be achieved through a multitude of specialized stages utilizing different 3D modalities. A popular approach is to divide it into two phases: an object detection and segmentation stage, followed by pose estimation using the prepared data from the previous stage [24]. The first stage reduces the feature space and narrows the area of interest by segmenting the object in the image and linking a confidence value to filter segmentation hypotheses. The following pose estimation stage then generates one or more pose hypotheses from these segmentations. This thesis utilized three open-source SOTA methods to validate whether the generated meshes can be used for pose estimation. First, the CNOS stand-alone segmentation approach is described, followed by the hybrid SAM-6D, which combines both stages into one system. Lastly, FoundationPose, a unified foundation model with built-in tracking and pose estimation utilizing CNOS or SAM-6D segmentation.

3.7.1 CNOS

Segmentation of an object from an RGB image can be formulated as a matching task, where feature matching is performed using an available GT mesh. Therefore, robust visual features are essential to achieve SOTA performance, which motivated the use of DINOv2 within CNOS. Here, features are obtained from a ViT model trained in a self-supervised manner to produce representations that generalize across diverse image domains and tasks without requiring finetuning [59].

As illustrated in Figure 3.14, CNOS comprises three stages: Onboarding, Proposal, and Matching. The Onboarding stage renders CAD models from different viewpoints using BlenderProc [28] or Pyrender before inference. This involves rotating the objects, performing offline rendering of template images, and extracting descriptors to create a database of object-specific reference descriptors. Critically, these descriptors correspond to transferable features derived from the pre-trained DINOv2 ViT. As a result, CNOS does not require additional training when applied to new objects. The Proposal stage is used during inference, beginning with an RGB image from which a segmentation model,

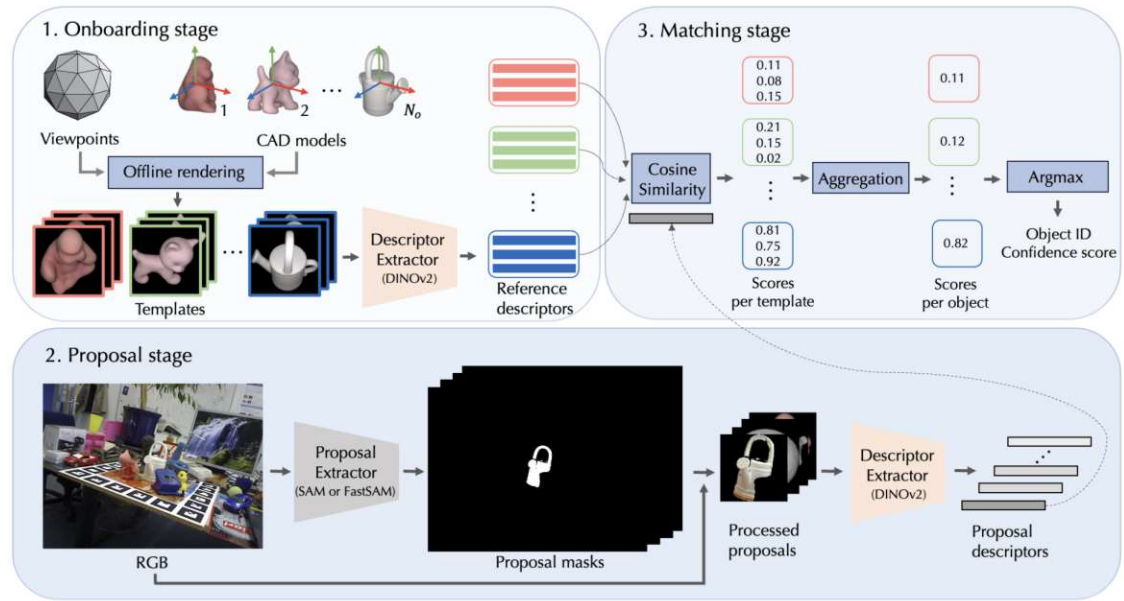


Figure 3.14: The three main stages of CNOS. Onboarding stage: Renders different viewpoints of the CAD models into templates, which are then processed with DINOv2 to create reference descriptors. Proposal stage: Employs SAM [89] or FastSAM [90] to generate proposal RGB masks from an RGB image, after which DINOv2 is applied again to extract proposal descriptors. Matching stage: All generated features are processed using cosine similarity to produce scores, which are then aggregated to identify the object ID with the highest confidence value. [26]

specifically SAM [89] or FastSAM [90], generates the proposed masks. These masks are then colored, and processed with DINOv2 to extract proposal descriptors. As an additional note, both SAM and FastSAM function on a grid-based scheme, treating each grid point as an initialization seed for new segmentations that detect corresponding surfaces, thereby dividing an image into multiple segmentation masks. Finally, features from the Onboarding and Proposal stages are compared in the Matching stage via cosine similarity, aggregated by score per template, and used to generate a set of segmentation masks associated with per-object confidences. Consequently, by leveraging a foundation model, CNOS enables training-free novel object detection and segmentation in a simple yet effective way.

3.7.2 SAM-6D

Instance segmentation and pose estimation are typically treated as separate tasks, which can result in the loss of complementary information. To address this, SAM-6D integrates an Instance Segmentation Model (ISM) and a Pose Estimation Model (PEM) into a unified two-stage framework, where the final pose estimation score is conditioned on the instance segmentation score to recover missed detections. At its core, the SAM-6D ISM operates analogously to CNOS, but additionally exploits the mesh to compute a geometric score, which is subsequently modulated by a visibility score to integrate the shape and size of

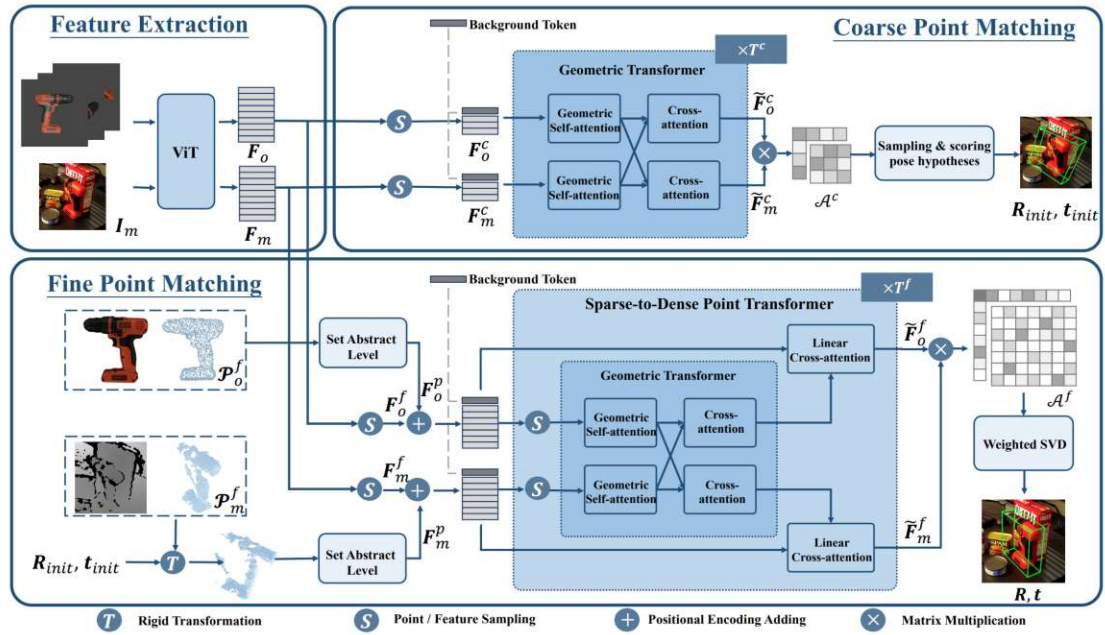


Figure 3.15: Overview of the SAM-6D PEM pipeline: In the Feature Extraction stage, a ViT is applied to both the rendered template images and the scene image to obtain features. These features, along with a background token, are subsequently used as input for a Geometric Transformer in the Coarse Point Matching stage, where a coarse pose is inferred for each template. The resulting hypotheses are then ranked according to a corresponding score. Subsequently, each sparse pose, the ViT-derived features, a pointcloud representation of the mesh, a depth map, and the background token are provided as input to an SDPT in the Fine Point Matching stage. The resulting outputs per coarse pose are then combined and weighted using singular value decomposition to determine the final fine pose. [8]

the object into the overall matching score. Consequently, this section concentrates on the PEM, which introduces the novel concepts of a background token and a Sparse-to-Dense Point Transformer (SDPT).

As shown on the left side of Figure 3.14, the PEM estimates the pose of an object using rendered object templates, a cropped scene image, a 3D model, and a depth map. This information is processed through a three-stage pipeline: Feature Extraction, Coarse Point Matching, and Fine Point Matching. Beginning with Feature Extraction, it utilizes a ViT to extract point-wise features F_o and F_m from the rendered templates and the image crop I_m . These features are then sampled and used as input for the Coarse Point Matching stage, which concatenates a learnable background token to solve the alignment problem of non-overlapping points and partial-to-partial correspondence, before processing the features with a trained Geometric Transformer [91]. The processed features \tilde{F}_o^c and \tilde{F}_m^c are then used to compute a soft assignment matrix A^c [8] where each row indicates the matching probabilities. Importantly, the included background token features are used

to identify invalid correspondence. The rotation R_{init} and translation t_{init} of the best hypothesis are then used to initialize the coarse poses.

Thereafter, the Fine Point Matching stage is applied, utilizing finer correspondences, to improve the coarse poses. The fine correspondences are sampled from the features of the first stage, denoted as F_o^f and F_m^f , and combined with positional encodings F_o^p and F_m^p . These encodings were learned from the object pointcloud P_o^f , and the transformed depth map P_m^f by applying a multi-scale set abstract level [92]. Similarly to coarse matching, the features are concatenated with a background token and processed with the introduced SDPT. By employing the proposed SDPT, the computational expense of learning dense point features remains low, yet the method still preserves the ability to model point-wise interactions. Subsequently, the soft assignment matrix A^f is calculated to find the best pose with weighted singular value decomposition. This approach achieves competitive pose estimation results on BOP [56] by integrating an ISM with a coarse-to-fine matching PEM.

3.7.3 FoundationPose

Considering the complex pipeline of the SAM-6D PEM, FoundationPose takes a more unified approach, eliminating the need for pre-rendered template matching. It combines model-based and model-free 6D object pose estimation and tracking by utilizing a neural implicit representation for NVS [9]. An additional novel idea that was implemented in the training scheme was LLM-aided synthetic data generation to train a generalizable transformer-based network. Considering that the thesis examined generated meshes, the Neural Unknown Object Modeling stage is not discussed.

Therefore, this thesis details the FoundationPose pipeline, as illustrated in Figure 3.16, with a focus on Language-aided Synthetic Data Generation at Scale, Pose Hypothesis Generation, and Pose Selection. Beginning with the initial stage, the training of generalizable ML models typically requires diverse datasets. Therefore, FoundationPose leverages generic objects sourced from Objaverse [93] and GSO [94]. These objects are then described using ChatGPT to generate varied texture prompts. The resulting prompts, alongside shape and texture noise, are subsequently used as input for a diffusion model to generate novel textures for object augmentation. A physics engine then produces a diverse synthetic training dataset of objects with known GT poses. However, due to legal restrictions regarding the diffusion model, the released pre-trained weights do not incorporate diffusion-augmented data, and performance degradation is thus anticipated [9].

Pose estimation begins by generating multiple pose hypotheses through iterative refinement of a coarse pose initially estimated from a segmentation and detection method, such as CNOS. The initial translation is determined from the 2D bounding box of the segmentation mask, using depth information from an RGBD image. To initialize rotations, a series of viewpoints are sampled by rotating the 3D model and rendering corresponding RGBD images. As the generated poses are initially inaccurate, they are fine-tuned using a trained ML network. Following the process depicted on the left side of the Pose Hypothesis Generation stage in Figure 3.16, the rendered RGBD image for each pose

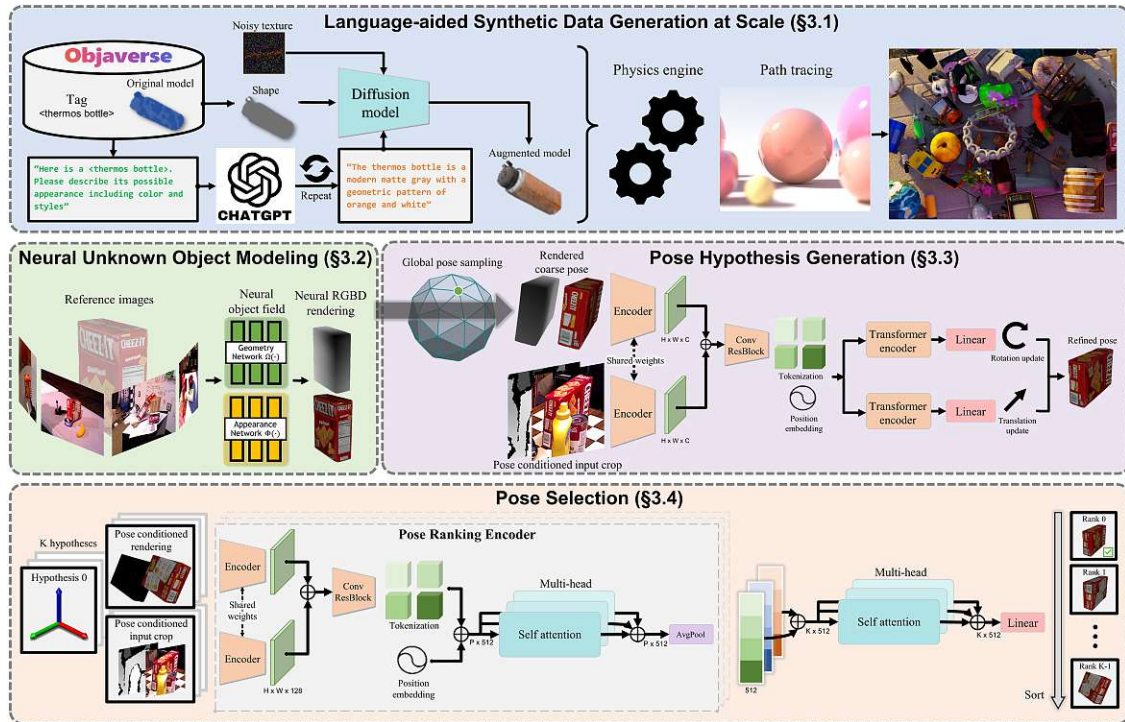


Figure 3.16: In the context of mesh-based object pose estimation, FoundationPose can be decomposed into three stages. To train its ML model, FoundationPose leveraged Language-aided Synthetic Data Generation, which modifies object textures using a diffusion model conditioned on diverse prompts obtained from ChatGPT. This process yields a varied synthetic dataset that is subsequently rendered within a physics engine. For pose estimation, the method first produces pose hypotheses by rendering the mesh and corresponding depth map from multiple viewpoints. Each of these candidates is then processed together with an RGBD pose conditioned input crop by a CNN and a transformer decoder, which iteratively updates rotation and translation until a refined pose is obtained. After this refinement, the resulting set of K hypotheses is evaluated in the Pose Selection stage by a Pose Ranking Encoder, and linearly projected, where the hypotheses are compared to one another to determine the best object pose. [9]

hypothesis is encoded, along with the pose-conditioned RGBD input crop, by a single CNN. The resulting feature maps are then concatenated and processed with a second CNN before being tokenized into patches with pose embeddings. Two ViTs then process these tokens to generate separate updates for translation and rotation. This allows for iteratively updating the initial pose of each hypothesis, refining it until an optimum is reached.

The optimal pose is then selected using a two-level comparison strategy in the Pose Selection stage, from a set of K hypotheses. Beginning with the Pose Ranking Encoder, this module utilizes the same strategy and backbone architecture as the hypothesis generation stage. However, to better leverage the global context, the resulting features are forwarded into a multi-head self-attention module. FoundationPose was specifically designed to avoid outputting a single score after the Pose Ranking Encoder to incorporate context from other pose hypotheses. Therefore, the features for each pose are downsampled using average pooling and subsequently used in the next step to apply a second multi-head self-attention module. Finally, the features are linearly projected to generate scores that rank the object pose hypotheses in descending order, leading to an SOTA pose estimation method that was also awarded as best open-source project at BOP 2024 [56].

3.8 Robotic Platform

The object segmentation and pose estimation models were initially engineered to function within a dedicated Python script for benchmarking or offline testing. However, this approach lacks integration with a real-world robotics system. Therefore, the code was adapted to work with the HSR platform to address this limitation. It comprises a mobile base, a robotic arm, and a suite of sensors, including cameras. Control is facilitated by ROS, which provides access to implemented controllers for navigation and object grasping within an environment. The modular architecture of ROS allows for the seamless integration of additional subsystems, as exemplified by the Grasping Pipeline developed by the V4R research group. This pipeline operates as a state machine running on an external PC and communicates with the HSR platform via ROS.

As shown in Figure 3.17, the Grasping Pipeline handles various data types and communicates with modular ROS servers. Because estimating the pose of an object is a two-step approach, the servers are split into two types: Pose Estimation and Instance Segmentation, in which each method from Section 3.7 is implemented to work within ROS. Before using the Grasping Pipeline, SAM-6D and CNOS require the generation of pre-rendered templates of the objects to be detected. In contrast, the server data modalities in Figure 3.17 show which data are used during inference, with SAM-6D using two modalities.

To successfully grasp an object using the HSR, the Grasping Pipeline is initiated by transmitting image data from the HSR to either the SAM-6D ISM or the CNOS server. These servers process the image data to identify objects, mapping each detected instance to a segmentation mask, and returning this information to the Grasping Pipeline. When a known object is detected, the newly inferred data is combined with the original image data and sent to SAM-6D PEM or FoundationPose for object pose estimation. Subsequently, this estimated pose is used to drive the robot near the location of the object, enabling

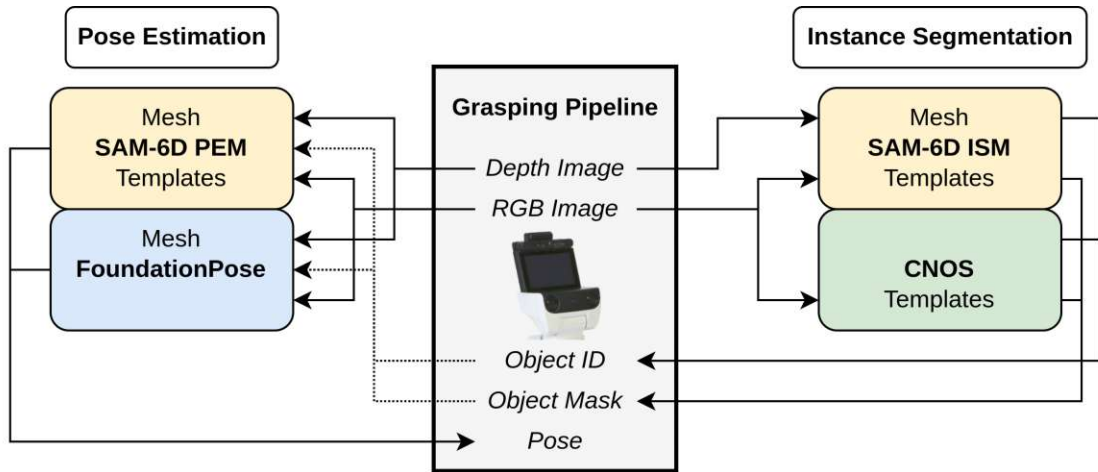


Figure 3.17: Overview of how the Grasping Pipeline from V4R interacts with the implemented Pose Estimation and Instance Segmentation ROS servers. The pipeline initiates by transmitting image data from the HSR to either SAM-6D ISM or CNOS for segmentation mask inference and associated object ID identification. Once an object is detected, the newly acquired data, combined with the previous image data, is sent to either SAM-6D PEM or FoundationPose for object pose estimation. Annotations on each server show the object-specific data needed at inference time.

the robot arm to grasp it. The servers utilized within the pipeline are modular and can be exchanged for alternative methods. Critically, the SAM-6D PEM operates exclusively with the SAM-6D ISM. The Code is published at <https://github.com/St333fan/DOPE>.

3.9 Mesh Generation Pipeline

Bringing everything together, the mesh generation pipeline functions as follows. Whereas Figure 1.2 provided a simplified overview, Figure 3.18 breaks down the system into its data flows between the four main stages: Data Acquisition, Camera Registration and Pointcloud Generation, Metric Height Estimation, and Novel View Synthesis for Mesh Generation. As described in Section 3.3, Data Acquisition has guidelines in place to ensure the correct initialization of the coordinate system to infer a consistent pose for supermarket objects. The first step is to create an object video from which the object is segmented with Grounded-SAM to generate object masks. Additionally, if object scaling is applied to the generated mesh, a subset of scene images of the scanning scene should be captured as described in Section 3.6. This dataset, comprising object and scene images alongside corresponding generated masks, contains all the necessary data for generating accurately scaled meshes.

Subsequently, the object images are registered using MAST3R-SfM or VGGT to generate camera poses and a pointcloud during the Camera Registration and Pointcloud Generation step. Independently, a metric scene pointcloud is generated from the object and scene images using MAST3R in the Metric Height Estimation step. Points not belonging to the

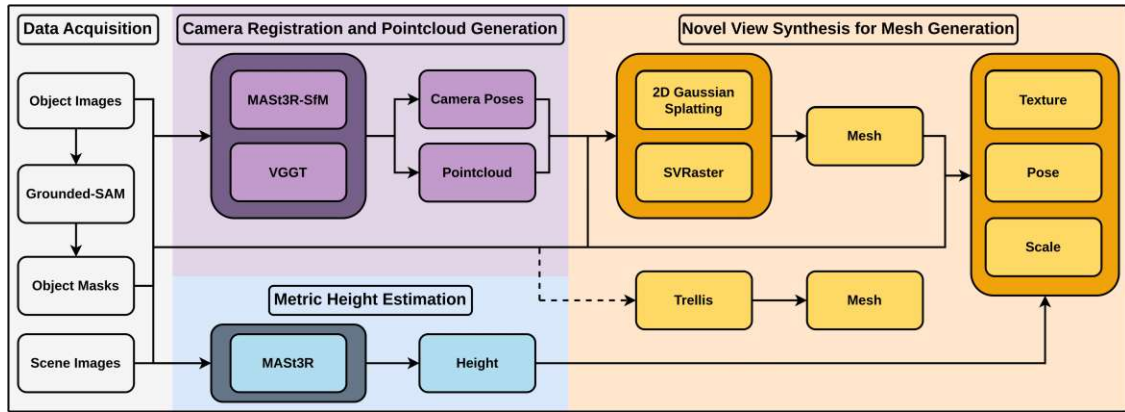


Figure 3.18: Detailed overview of the mesh generation pipeline, comprising four stages: Data Acquisition, Camera Registration and Pointcloud Generation, Metric Height Estimation, and Novel View Synthesis for Mesh Generation. Data Acquisition yields an object-specific dataset captured by a camera and Grounded-SAM. The data consists of object/scene images and object masks, which are subsequently used for mesh generation. Camera Registration and Pointcloud Generation utilizes MAST3R-SfM or VGGT to generate camera poses and a pointcloud from the object images. Independent of this, Metric Height Estimation reconstructs a metric scene pointcloud from object and scene images using MAST3R, refining it by projecting onto the first registered camera plane and utilizing an object mask to infer the height of the object. Finally, Novel View Synthesis for Mesh Generation creates and post-processes the mesh with texture and pose, optionally scaling it to correct dimensions. 2DGS and SVRaster are the primary NVS methods analyzed in this thesis, while Trellis provides a baseline without camera pose information.

object are removed by projecting the pointcloud onto the first registered camera plane and comparing it to the object mask. This leaves only those points that define the dimensions of the object, as described in Section 3.6, from which the height is estimated.

In the last step, Novel View Synthesis for Mesh Generation, a mesh is generated and post-processed with correct texture, provided with a pose, and scaled to the correct dimensions. This thesis focuses on the two NVS methods, 2DGS and SVRaster, for a comprehensive analysis. Trellis is also included in Figure 3.18 to provide a comparison to SOTA generative 3D reconstruction methods. It generates a mesh directly from unregistered object images and their corresponding masks; the absence of camera pose information rules out post-processing, and consequently, robotic manipulation. Compared to this, 2DGS and SVRaster densify the generated pointcloud using camera poses and their corresponding object images, while the masks allow for differentiation between 3D information belonging to the object and that to be removed. These training-based approaches iteratively generate an object-specific pointcloud or voxel representation and convert it into a mesh. This initial mesh is then textured by triangulating RGB data from the registered object images, provided with the correct object pose inferred from the camera poses, and finally scaled to the correct height.

The four-stage pipeline facilitates the automatic generation of suitable meshes for robotic manipulation with the additional benefit of being modular. With the option to choose between MAST3R-SfM or VGGT and 2DGS or SVRaster, depending on the available compute resources, desired accuracy, or processing speed requirements. MAST3R-SfM requires a GPU with 5GB VRAM and completes processing in minutes, whereas VGGT with full precision requires a GPU with at least 80GB VRAM but infers all data in seconds. 2DGS generates meshes with surfaces that more closely represent the GT, with a processing time of several minutes; SVRaster is less accurate, but offers processing in seconds. Which method performs best in which case is evaluated in the next Chapter. The Code is published at <https://github.com/St333fan/meshgen-zeroshop>.

4 Experiments and Results

To assess the quality and robustness of the proposed mesh generation pipeline for zero-shot object pose estimation, four sub-goals were examined. Beginning with camera registration and pointcloud generation via VGGT and MAST3R-SfM, 2DGS and SVRaster were evaluated for their reliability and mesh quality to determine which configuration yields the best 3D object. Furthermore, instant image-to-mesh generation with Trellis was incorporated into the comparison, which leverages SLAT representations. Secondly, because the generated meshes possess an inherently ambiguous scale, the cost-effective scaling method described in Section 3.6 was applied, and the estimated scale was compared with the actual scale of the reference objects. These two sub-goals represent the key aspects for assessing how the created objects will perform in object detection, segmentation, and pose estimation. To validate this, the standardized BOP benchmark was used to compare the generated objects on 2D segmentation and 6D pose estimation tasks. Finally, robotic manipulation with the HSR was examined in real-world settings using ROS implementations of CNOS, SAM-6D and FoundationPose.

4.1 Mesh Generation Reliability and Quality

To determine the reliability and quality of the automated mesh generation pipeline, the optimal combination of MAST3R-SfM, VGGT, 2DGS, and SVRaster was identified based on reconstruction success. Additionally, for comparison, Trellis objects are discussed. The best combinations were then evaluated on a subset of real YCB-V and MANiBOT objects to quantify reconstruction differences using real-world data. Importantly, VGGT was predominantly executed with only one quarter (1024) of the full set of tracking features (4096) enabled, to ensure that it could be run on local hardware.

4.1.1 Virtual YCB-V Objects

To establish a baseline for mesh reconstruction using NVS methods, initiated by VGGT and MAST3R-SfM, all 21 virtual YCB-V GT objects were rendered in BlenderProc to generate scene images consisting of 10 images per viewing angle. To ensure comprehensive capture of both geometric and visual object features, camera translation and rotation were evenly spaced around each object, centered on its bounding box. Figure 4.1 shows the three test environments, using camera angles of 0° , 45° , and -45° relative to the horizontal plane. To investigate the influence of the supporting surface, scenes were differentiated by high- and low-feature surfaces. In addition, segmented object masks were also included to assess the need for surface information for camera registration and NVS. Consequently, the first two cases comprised 20 images each, while the segmented case, unconstrained by surface limitations, yielded a denser representation of 30 images, including views from below.



Figure 4.1: YCB-V object rendering with high/low feature surface and extracted segmented RGB masks; rendering angles of 0° , 45° , and -45° to the horizontal plane.

Following rendering, VGGT and MAST3R-SfM were applied to each image set to generate camera poses and pointclouds. Subsequently, 2DGS and SVRaster were utilized to generate meshes, which are then categorized according to quality: "none" meshes are considered failed or unusable, "sparse" meshes are inconsistent but usable, and "good" meshes exhibit smooth and feature-rich geometry. The mesh generation with Trellis diverged from the standard pipeline; the ML model was inferred with two RGB masks per object, captured from opposing viewpoints at an 45° angle, to represent the whole object. Given the generative nature of Trellis, the mesh geometry and texture output vary considerably between inferences. Consequently, the optimal mesh of a set of six generations was manually selected for performance evaluation in BOP. Representative examples of mesh and texture quality are illustrated with four YCB-V objects in Figure 4.2, demonstrating smooth mesh surfaces while simultaneously revealing inaccuracies in texture generation.

The reconstruction success achieved with 2DGS is detailed in Table 4.1, categorized by VGGT (quarter precision) and MAST3R-SfM. The numbering for each object can be found in Figure 3.3. The three test cases per method are separated into the low-feature surface su_l , the high-feature surface su_h , and the segmented object without surface se . Figure 4.3 illustrates the baseline for the quality assessment of mesh reconstruction. The first object serves as an example of a \circ = "sparse" reconstruction, the second as an example of a \bullet = "good" reconstruction, and a not visible \times = "none" indicates a non-existent reconstruction, the third is the textured post-processed object, and the fourth is the GT object.

When comparing the different approaches, both registration methods exhibit distinct strengths. In particular, MAST3R-SfM with su_h is the only combination that achieves successful reconstructions for all objects. Although VGGT surpassed MAST3R-SfM in the se task, it was unable to reconstruct the featureless "bowl" object. Moreover, an increase



Figure 4.2: The first two objects represent qualitative examples of the surface of Trellis meshes, whereas the colored objects serve as representative instances of the inaccurate texture.



Figure 4.3: The first mesh shows a sparse SVRaster reconstruction, followed by the 2DGS mesh and its textured post-processed counterpart, and finally the YCB-V GT mesh.

in surface features is positively associated with reconstruction success for MAST3R-SfM, whereas VGGT benefits primarily from a larger number of inferred images.

The SVRaster reconstruction is detailed in Table 4.2 without MAST3R-SfM *su_h* and *se*, because it exhibited the same performance as 2DGS in Table 4.1. The remaining metrics show a lower reconstruction success relative to 2DGS. Additionally, the lower portion demonstrates 2DGS/VGGT performance with full feature tracking (4096), resulting in 77 GB of VRAM consumption on an H100 cluster without substantial reconstruction gains. The differences in mesh quality compared to 2DGS are visualized in Figure 4.4, with three geometrically distinct object reconstructions initiated by MAST3R-SfM *su_h*. In summary, SVRaster generates the same object dimensions as 2DGS, but the reconstructed mesh surface is not as smooth. The denser *se* object scenes produced meshes of comparable quality, including additional geometry on the underside. However, because the scene reconstruction is not reliable for all objects, they are not discussed further.

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21
VGGT <i>su_l</i>	○	●	●	●	○	●	○	●	●	○	●	●	×	●	●	●	○	●	●	●	●
VGGT <i>su_h</i>	●	●	●	●	●	●	○	●	×	●	●	×	●	●	●	×	●	●	○	●	●
VGGT <i>se</i>	●	●	●	●	●	●	●	●	●	●	●	●	×	●	●	●	●	●	●	●	●
MASt3R <i>su_l</i>	○	●	●	●	○	●	●	●	●	●	×	●	×	●	●	●	●	●	●	●	●
MASt3R <i>su_h</i>	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●
MASt3R <i>se</i>	○	●	●	●	○	●	●	●	○	×	●	×	×	×	○	×	○	×	●	●	●

Table 4.1: 2DGS YCB-V mesh reconstruction success initiated by VGGT (quarter precision) and MAST3R-SfM: × = none, ○ = sparse, ● = good

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21
MASt3R <i>su_l</i>	○	●	●	×	○	●	●	●	○	●	○	●	×	●	●	●	●	●	●	●	●
VGGT <i>su_l</i>	○	●	●	●	○	●	○	●	●	○	●	●	○	●	●	●	×	●	●	○	●
VGGT <i>su_h</i>	○	●	●	●	●	●	●	●	●	×	●	●	×	●	●	●	×	○	●	×	●
VGGT <i>se</i>	○	○	○	○	●	○	×	●	○	●	×	○	×	●	○	●	×	○	○	○	○
VGGT <i>su_l</i>	●	●	●	●	●	○	●	●	×	●	●	×	●	●	●	×	×	●	●	●	●
VGGT <i>su_h</i>	●	●	●	●	●	●	●	●	×	●	●	×	●	●	●	×	●	●	×	●	●
VGGT <i>se</i>	●	●	●	●	●	●	●	●	●	●	●	●	×	●	●	●	●	●	●	●	●

Table 4.2: SVRaster YCB-V mesh reconstruction success initiated by MAST3R-SfM and VGGT (quarter precision): × = none, ○ = sparse, ● = good. The lower part depicts 2DGS and VGGT (full precision).



Figure 4.4: Overview of the reconstruction quality of three geometrically distinct YCB-V objects. Of each pair, the left object is from 2DGS and the right from SVRaster.

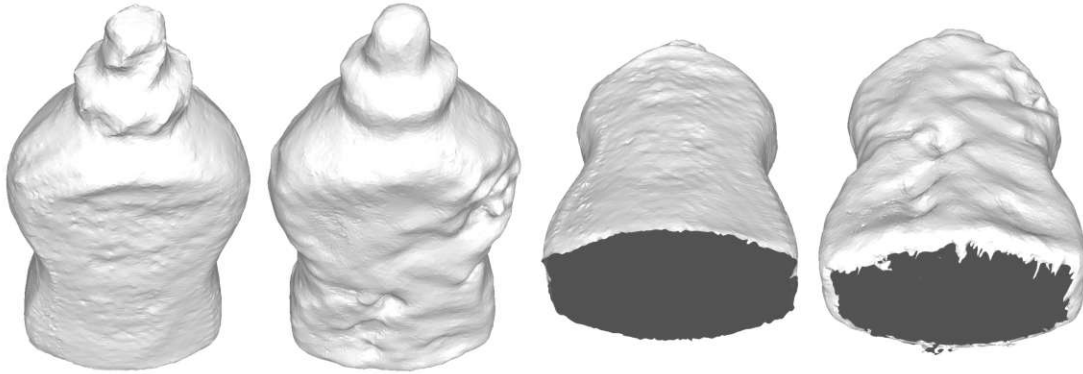


Figure 4.5: Mesh quality comparison between 2DGS and SVRaster on real-world data, showing top and bottom views, with 2DGS yielding more accurate surfaces.

4.1.2 Real YCB-V and MANiBOT Objects

Given that only the registration approach of MAST3R-SfM successfully reconstructed all virtual YCB-V object scenes, it was selected to evaluate a subset of real objects from the YCB-V and MANiBOT datasets. As detailed in Section 3.3.1, each object was placed on a flat surface, and a video was recorded while the camera moved around it in a circular path. The camera started parallel to and centered on the front side of the object, initializing the object pose as described in Section 3.1.3. For each video, 20 equally spaced frames were extracted and processed by Grounded-SAM as detailed in Section 3.3.2, to obtain object masks. The images were then registered and converted to pointclouds using MAST3R-SfM, which served as input for both 2DGS and SVRaster. All objects were successfully digitized into meshes and are shown in Figure 4.6 and Figure 4.7, where the left/upper meshes correspond to the 2DGS reconstruction and the right/lower meshes to SVRaster. A more detailed comparison between 2DGS and SVRaster of the YCB-V mustard object is shown in Figure 4.5, which depicts the same rough surface behavior as described in Section 4.1.1.



Figure 4.6: Reconstructed YCB-V objects, with the left/upper object illustrating reconstruction using 2DGS, followed by SVRaster.



Figure 4.7: Reconstructed MANiBOT objects, with the left/upper object illustrating reconstruction using 2DGS, followed by SVRaster. Upon application of the texture, no apparent differences become visible.

4.2 Object Scaling

Following the height estimation procedure from Section 3.6, the estimated height of each real object from the YCB-V and MANiBOT subsets is presented in Table 4.3 and Table 4.4, and compared to the measured height. Analysis of the difference column reveals an error limit of approximately $\pm 10\%$ for these objects, with the notable exception of the toothbrush, which exhibits a larger error. Additionally, Figure 4.8 shows an inverse relationship between error and object height.

	height [m]	mast3r-height [m]	difference [mm]	difference [%]
mustard_bottle	0.192	0.2031	11.12	5.79
potted_meat_can	0.084	0.092	8.04	9.57
bowl	0.055	0.0601	5.1	9.28
cracker_box	0.22	0.2244	4.42	2.01
master_chef_can	0.14	0.1413	1.34	0.96
gelatin_box	0.075	0.0819	6.89	9.19
large_marker	0.122	0.1259	3.85	3.16
extra_large_clamp	0.036	0.0334	-2.6	-7.22

Table 4.3: Comparison between the actual heights of YCB-V objects and those estimated using the MAST3R registration method.

	height [m]	mast3r-height [m]	difference [mm]	difference [%]
soap	0.158	0.148	-10.0	-6.33
ahorn_sirup	0.163	0.15	-13.0	-7.98
tomato_paste	0.195	0.208	13.0	6.67
kokos_can	0.113	0.1215	8.5	7.52
hand_cream	0.135	0.1468	11.8	8.74
wet_wipes	0.108	0.1058	-2.2	-2.04
razors	0.195	0.1756	-19.4	-9.95
balsamic	0.196	0.202	6.0	3.06
toothbrush	0.027	0.0312	4.2	15.56

Table 4.4: Comparison between the actual heights of MANiBOT objects and those estimated using the MAST3R registration method.

4.3 BOP with NVS Meshes

Although the reconstructed meshes of the virtual YCB-V objects generated by 2DGS and SVRaster appear visually plausible, they lack the underside geometry, which is expected to affect the detection, segmentation, and pose estimation performance to an unknown extent. In contrast, meshes generated by Trellis exhibit inconsistent geometric dimensions and inaccurate texture representations. To understand this influence on the performance of CNOS, SAM-6D, and FoundationPose, a benchmark validation was required. Therefore,

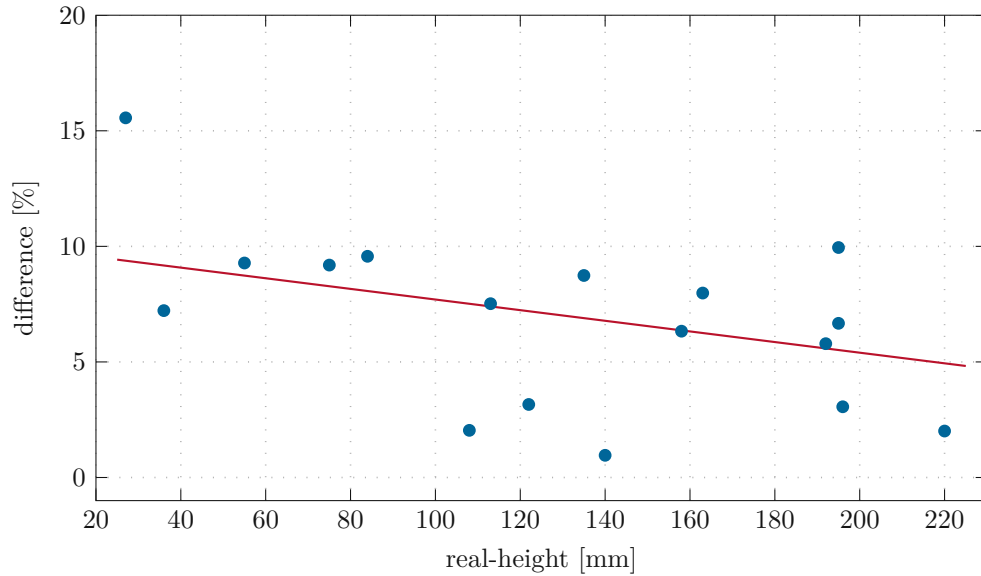


Figure 4.8: The relationship between the real object height and the absolute measurement error in percent for objects from the YCB-V and MANiBOT datasets. Individual measurements are represented as blue points, with a red line indicating the trend of decreasing error as object height increases.

BOP was utilized because the official performance scores for the YCB-V objects are available from previous work [56]. As a first step, the AP/AR scores reported in the original papers using virtual GT objects from the YCB-V dataset were reproduced. Subsequently, the reconstructed objects from Section 4.1.1 were used to benchmark each ML model to determine if they maintain accuracy.

Because these reconstructed objects were generated with BlenderProc scenes, they possess an arbitrary scale and were therefore aligned to their corresponding GT counterparts using the ICP algorithm to ensure the correct height. Importantly, the CNOS mask filtering of false segmentation hypotheses used in the original FoundationPose implementation was not publicly available and therefore had to be implemented as part of this thesis. Consequently, the masks used for the CNOS/FoundationPose evaluation were different. Therefore, GT masks, for which an implementation was available, were additionally employed to evaluate FoundationPose. To analyze how the generated meshes affect the AP score for segmentation and the AR score for pose estimation in relation to the official BOP meshes, four distinct scenarios were chosen:

- CNOS on GT/NVS meshes, with FastSAM
- ISM (SAM-6D) on GT/NVS meshes, with SAM
- FoundationPose with GT/CNOS masks on GT/NVS meshes
- PEM (SAM-6D) with ISM masks on GT/NVS meshes

AP/AR scores are presented in Table 4.5, divided into segmentation and pose estimation tasks, with the submission row reporting the official scores of the BOP benchmark. Focusing on the segmentation task, the AP scores of both methods, CNOS and SAM-6D ISM, were successfully reproduced. When employing NVS-generated meshes, all methods exhibit a comparable decrease in performance, with only a marginal deviation relative to the scores obtained using SVRaster meshes. When using FoundationPose on GT masks, the GT BOP meshes achieve performance that exceeds the official AR score. Furthermore, the 2DGS and SVRaster meshes exhibit only a minor decrease in performance, whereas the Trellis meshes show substantial performance degradation. When employing CNOS-based segmentation, the accuracy decreased across all evaluated domains. In addition, the official reported scores could not be fully reproduced, and the performance gap between the GT BOP meshes and the 2DGS/SVRaster meshes increased further. In comparison, the official SAM-6D PEM scores were almost reproducible and exhibited a smaller performance gap compared to 2DGS and SVRaster than FoundationPose/CNOS. However, the difference between 2DGS and SVRaster is slightly greater than that observed in the FoundationPose evaluation. Trellis again demonstrated comparatively poor performance.

	Segmentation		Pose Estimation		
	CNOS	ISM (SAM-6D)	FoundationPose	PEM (SAM-6D)	
Submission	0.599	0.605	0.882	0.845	
			GT	CNOS	
BOP	0.6	0.603	0.915	0.731	0.832
2DGS	0.56	0.561	0.889	0.543	0.751
SVRaster	0.541	0.567	0.877	0.539	0.71
Trellis	0.567	0.561	0.306	0.267	0.26

Table 4.5: Official BOP AP (segmentation) and AR (pose estimation) scores compared to reproduced scores across different mesh datasets from the BOP objects. FoundationPose was additionally benchmarked with GT masks because the official filtering of the CNOS masks was not accessible.

4.4 Robotic Object Manipulation

When using NVS-generated meshes, the BOP scores in Section 4.3 indicate a minor reduction in accuracy in the segmentation and pose estimation tasks. However, the applicability of this finding to scenarios that employ real YCB-V and MANiBOT objects, combined with the scaling method from Section 3.6, remained uncertain. Consequently, this section focuses on evaluating the suitability of the real reconstructed meshes for their intended purpose, accurate robotic manipulation. Providing insight into whether the combination of data aggregation, object reconstruction, and scaling is effective within a robotic grasp software pipeline.

To limit the number of experimental configurations, preliminary evaluations were conducted for the segmentation and pose estimation tasks. Specifically, combinations of CNOS, SAM-6D ISM, FastSAM, and SAM were assessed in scenes with clear object boundaries to identify outlier methods. Subsequently, FoundationPose and SAM-6D were subjected to the same evaluation procedure. Finally, the best-performing segmentation and pose estimation combination was evaluated within the HSR grasping pipeline.

4.4.1 Segmentation

To assess the segmentation performance of the established methods using NVS-generated meshes, several test scenes were analyzed. Representative results are shown in Figure 4.9, which illustrates the typical behavior observed, with CNOS shown in the second column and SAM-6D ISM in the third. In scenes without occlusions (first row), both CNOS/FastSAM and SAM-6D/FastSAM correctly segment all objects. However, when objects are flipped (second row), methods relying on FastSAM begin to exhibit failures for objects with sparse geometric reconstruction on the underside, such as the "bowl". By contrast, SAM provides more stable segmentations (third row), but only when combined with CNOS; in this configuration, segmentation performance improves, whereas SAM-6D does not yield further benefits and additionally exhibits difficulties with occluded objects.

This preliminary evaluation indicated that the CNOS/SAM configuration yields the most favorable performance. An additional illustrative example for CNOS/SAM is presented in Figure 4.10, which depicts MANiBOT objects. The remaining combinations of methods exhibited behavior comparable to that observed with the YCB-V objects. Importantly, no significant differences in segmentation performance were observed between the 2DGS and SVRaster meshes.

4.4.2 Pose Estimation

Considering the set of remaining available methods, an additional set of preliminary tests was conducted to determine which combination of methods is meaningful for evaluating grasp performance, given the large number of mesh variants and approaches. Based on the insights from the previous section, where SAM demonstrated superior performance compared to FastSAM, all combinations involving FastSAM were discarded for further consideration. Moreover, due to indications that CNOS exhibits greater robustness to NVS-generated meshes, FoundationPose was no longer evaluated using SAM-6D ISM segmentations. This constraint ultimately resulted in the selection of only the FoundationPose/CNOS configuration and the complete SAM-6D pipeline for evaluation. These configurations were examined to identify potential outlier behavior, specifically by assessing their relative robustness on NVS-generated meshes in comparison to GT meshes.

The preliminary evaluation was conducted on two YCB-V object scenes, because GT meshes are available: one with objects in upright orientation and one with objects inverted, as illustrated in Figure 4.11. Each pose estimation method was assessed on GT, 2DGS, and SVRaster, with three inference runs per mesh type. The quality of the estimated poses was subjectively evaluated by inspecting whether the predicted rotational and translational components of the pose were approximately aligned with the corresponding GT pose.



Figure 4.9: Representative test scenes, with the second column illustrating CNOS segmentations and the third column presenting SAM-6D ISM segmentations. The segmentations in the first and second rows are generated using FastSAM, whereas those in the final row are produced using SAM.



Figure 4.10: Representative segmentation outcome obtained using the most robust CNOS/SAM configuration on MANiBOT objects.

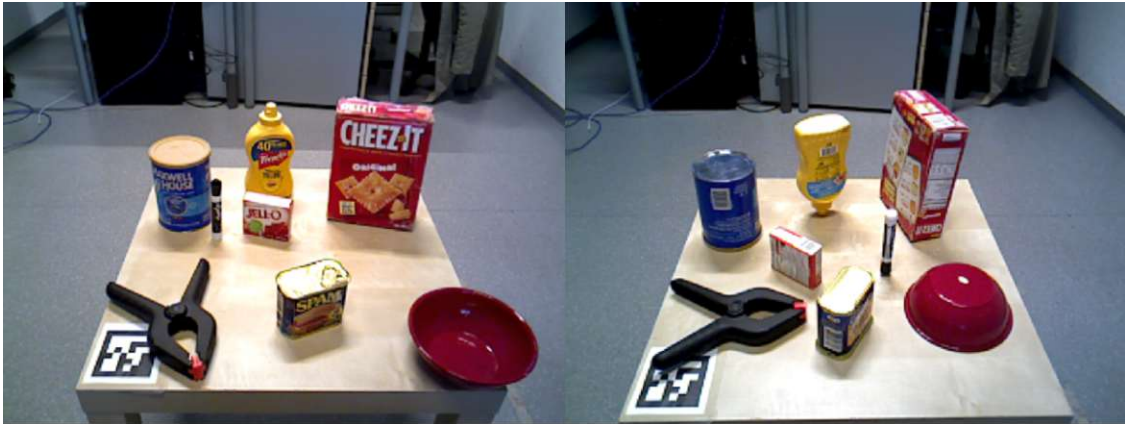


Figure 4.11: Preliminary pose evaluation test scenes to quantify relative differences among GT, 2DGS, and SVRaster meshes, with FoundationPose/CNOS and SAM-6D.

Analysis of the first scene indicated that FoundationPose with GT meshes achieves a consistent pose estimation. In comparison, when employing 2DGS and SVRaster meshes, FoundationPose continued to produce stable pose estimates; however, a clear limitation emerges in accurately estimating rotations around the z-axis for the objects "bowl" and "clamp". SAM-6D, when using GT meshes, exhibited slightly lower performance relative to FoundationPose, while manifesting the same systematic inaccuracies in rotation around the z-axis. A further degradation in performance was observed for SAM-6D when using 2DGS and SVRaster meshes, again consistently reproducing the aforementioned z-axis rotation error.

In the second scene, FoundationPose yielded stable and reliable pose estimations across all mesh types. In contrast, SAM-6D with GT meshes incorrectly estimated approximately half of the object poses; notably, several poses were incorrectly predicted as upright rather than flipped. When deployed with 2DGS and SVRaster meshes, the pose estimation of SAM-6D deteriorated, rendering the method unusable. In summary, compared to SAM-6D, FoundationPose demonstrates substantially higher robustness to the missing undersides of the NVS meshes generated in this thesis. Consequently, SAM-6D was excluded from the final robotic grasping experiments.

4.4.3 Grasp

The conclusions of the preliminary results of Sections 4.4.1 and 4.4.2 indicate that FoundationPose with CNOS/SAM is a robust combination for estimating 6-DoF poses of real objects with NVS-generated meshes. However, to prove this hypothesis, a robotic grasp success test was conducted. To keep grasp testing expenditures sufficient, a new subset consisting of distinct graspable supermarket objects from the YCB-V and MANiBOT datasets was selected. In addition to requiring a graspable surface, objects exhibiting varying heights and a high scaling error were selected; the naming scheme is adopted from Tables 4.3 and 4.4. The selected objects and their characteristic properties are

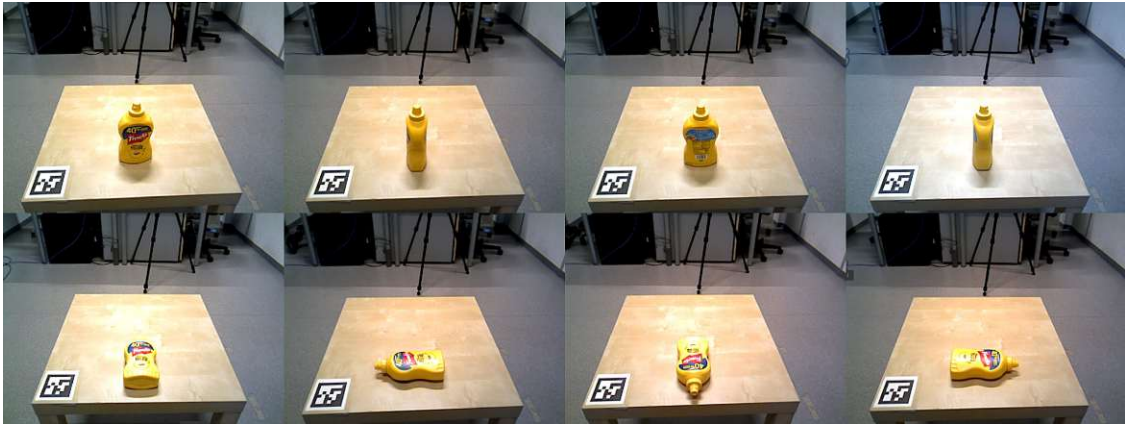


Figure 4.12: Overview of the eight object poses used for grasping experiments. If an object cannot be stably inverted, it is positioned on its side, as illustrated with the "mustard_bottle" object.

as follows: "mustard_bottle", featuring rounded edges; "potted_meat_can", a metallic object; "gelatin_box", a small cuboid container; "soap", composed of semi-transparent plastic; "ahorn_sirup", a semi-transparent glass bottle; "hand_cream", a tube-shaped object; "razors", a deformable object; and "toothbrush", a small object with transparent plastic components, which also exhibited the highest observed error. To enable object manipulation, each object was manually annotated with several grasp poses.

Given the large number of potential object placement variations, a rigid testing protocol was adopted to ensure a repeatable experimental environment. The test scenario was defined as follows: each object is positioned at the same location in front of the HSR, within its grasping range, and then rotated four times by 90° around the z-axis. Subsequently, the object is flipped onto its top and rotated four times again, resulting in a total of eight distinct poses per object. If an object could not be flipped, it was instead placed on its side, as illustrated in Figure 4.12. The objects to which this applied were "mustard_bottle", "soap", "ahorn_sirup", "hand_cream", and "razors".

For each scene, the target object was detected and segmented using CNOS, its pose was estimated with FoundationPose, and subsequently grasped by the HSR. An attempt to grasp was classified as successful if the robot established a stable grasp and lifted the object, as shown in Figure 4.13. In instances where the pipeline failed at any stage, it was re-executed; this repeated execution is defined as a re-grasp. The results of the grasp evaluation for each mesh type and object are summarized in Table 4.6. For each object, the first row reports the segmentation success, while the second row reports the grasp success. Results are encoded as: \times for failed segmentation or grasp, \circ for re-segmentation or re-grasp, and \bullet for successful first-attempt segmentation or grasp. Considering only a single grasp attempt, the pipeline using 2DGS meshes achieved a grasp success rate of 85.9%, while the pipeline using SVRaster achieved 89.1%. When re-grasps were included, the success rate for 2DGS increased to 93.8%, while that for SVRaster increased to 90.6%. The two primary outlier cases were the inverted "toothbrush", which was not detectable using CNOS, and the upright "soap", for which FoundationPose consistently estimated

the pose as lying horizontally. In addition, the following observations were made: CNOS exhibited some difficulty in segmenting the "gelatin_box"; the HSR lost its otherwise stable grasp on the "mustard_bottle" twice during lifting; and the "razors," which were meshed in a compressed configuration with an underestimated object height, appear to be affected by the cumulative error in one specific pose.

	2DGS								SVRaster							
	1	2	3	4	5	6	7	8	1	2	3	4	5	6	7	8
potted_meat_can	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•
gelatin_box	•	•	•	•	•	•	○	•	•	○	•	•	•	○	•	•
mustard_bottle	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•
razors	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•
hand_cream	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•
soap	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•
ahorn_sirup	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•
toothbrush	•	•	•	•	•	○	×	○	•	•	•	•	×	×	×	×
	○	•	•	•	•	○	×	○	○	•	•	•	×	×	×	×

Table 4.6: Robotic manipulation performance comparison using 2DGS and SVRaster meshes. For each object, the first row reports segmentation success, and the second row the grasping success for each evaluated pose with: × = failure, ○ = multi-shot, • = single-shot.

In multi-object scenes with clear object boundaries, where the objective is to grasp all items resting on a surface, a comparable behavior was observed for the "toothbrush". As illustrated in Figure 4.14, two types of scenes are presented: one containing upright objects and the other with objects lying on the surface. For both configurations, the segmentations and the estimated object poses projected onto the image plane are shown. The HSR successfully grasped all objects in both scenes, utilizing 2DGS and SVRaster meshes, except for the flipped "toothbrush". This object could not be segmented by CNOS, which instead produced an incorrect prediction corresponding to the "mustard_bottle".

In a final exploratory analysis involving six multi-object scenes with overlapping object boundaries, the objective was to assess the robustness of CNOS and to determine the proportion of cases in which it successfully produced a correct estimate of object poses. An example of a scene is shown in Figure 4.15. Importantly, because the grasping pipeline in such cases is prone to errors, particularly due to its failure to reliably detect boundaries and collisions with surrounding surfaces or objects, no actual object manipulation was



Figure 4.13: Example of a successfully executed robotic grasp on the smallest test object, the "toothbrush".

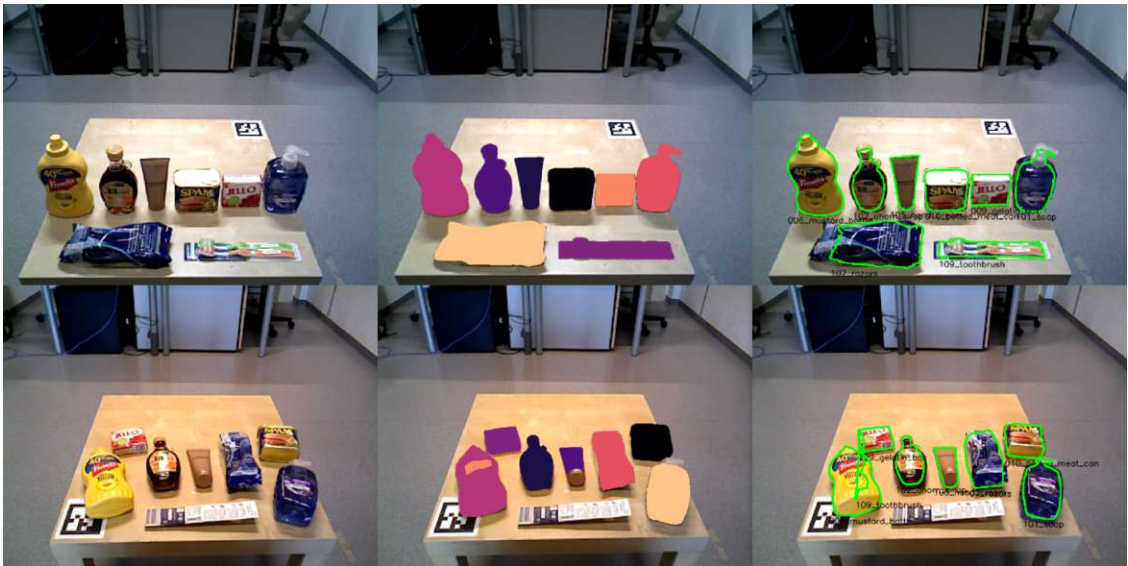


Figure 4.14: Overview of the multi-object grasp evaluation scenes, including object segmentations and corresponding 6-DoF poses projected onto the image plane. Two recurring issues are illustrated. In the first row, the deformable object was generated within a compressed configuration; consequently, FoundationPose estimates its pose with a systematic offset. Additionally, the "toothbrush" is incorrectly segmented, resulting in the estimated pose being wrongly associated with an incorrect object instance.



Figure 4.15: Illustrative example of one of the six exploratory multi-object scenes with overlapping object boundaries, including object segmentations and their corresponding 6-DoF poses projected onto the image plane. The visualization highlights recurrent failure modes: mutual misclassification between "soap" and "razors", duplicate segmentation of a single object with two distinct class labels, as exemplified by "ahorn_sirup", and missing segmentations for objects with more strongly overlapping boundaries.

possible. Instead, only a subjective assessment of the resulting grasp poses was conducted. Of 48 objects, 2DGS/CNOS correctly segmented 63%, for which 100% of the corresponding grasp poses were correct, while SVRaster/CNOS correctly segmented 65% of the objects, for which 94% of the resulting grasp poses were correct. Typical CNOS failure modes are depicted in Figure 4.15: mutual misclassifications between the classes "soap" and "razors"; duplicate segmentation of a single object with inconsistent class labels, for example, "ahorn_sirup"; and missed segmentations for objects that exhibit substantial boundary overlap with neighboring items.

5 Conclusion

Taking into account the research questions introduced in Section 1.2, the results presented in Chapter 4 demonstrate a clear differentiation among the methods evaluated within the automatic mesh generation pipeline, an acceptable mesh-scaling error associated with a cost-effective depth estimation approach, and consequently accurate robotic manipulation in real-world scenarios. Nevertheless, the findings warrant careful discussion; several methods exhibit distinct advantages and limitations, and certain results are only valid under specific experimental conditions or operational scenarios.

In terms of camera registration, MAST3R-SfM outperformed VGGT across all surface-based scene categories, with the exception of segmented object scenes, for which the VGGT registration strategy yielded substantially more successful object reconstructions. This suggests that MAST3R-SfM relies on the presence of extended surfaces to achieve robust camera registration, a dependence that likely arises from its stereo-matching approach. In contrast, VGGT, which processes all images during one inference, is better able to capture the global geometric structure of a segmented object. The "bowl" object constitutes a notable exception. VGGT exhibited difficulties in this case due to the largely featureless appearance and symmetric geometry of the object, which induces an inherently ambiguous registration problem. The only combination that produced a high-quality mesh of the "bowl" was based on camera poses estimated by MAST3R-SfM, which registers the feature-rich surface and thus reconstructs the "bowl" geometry essentially as a byproduct. Finally, varying the number of tracking features in VGGT for the BlenderProc scenes reveals a saturation effect; reducing the feature count from full precision (4096 features) to quarter precision (1024 features) has only a marginal impact on the reconstruction quality, indicating that the object and surface features of the scene are already adequately captured with a lower count of features.

When interpreting the reconstruction performance of the NVS-generated meshes, it became evident that the quality of the mesh is dependent on accurate camera registration and the resulting pointcloud. In the BlenderProc scenes, all 2DGS meshes that were successfully reconstructed were preceded by correct camera registrations, while failed mesh reconstructions were consistently associated with inaccurate registrations. In contrast, SVRaster produced unsuccessful meshes in cases where the camera registration itself had been successful. For example, SVRaster systematically underperformed in all scene types using VGGT compared to 2DGS. In particular, the segmented VGGT scenes suggest that SVRaster is sensitive to how VGGT generates the pointcloud, because this effect is not observed for pointclouds produced by MAST3R-SfM, where both 2DGS and SVRaster exhibit comparable behavior.

A clear distinction emerged when comparing the rendering time and mesh quality between SVRaster and 2DGS. In scenarios where both methods were able to produce meshes,

SVRaster is an order of magnitude faster than 2DGS, although at the cost of inaccurate mesh surfaces. However, despite the visibly inferior geometric quality of the SVRaster meshes, this degradation is not reflected in the BOP scores, where both methods achieve comparable performance. This observation does not hold for the Trellis meshes. Although their approximate geometric dimensions and moderately accurate texture may appear promising at first glance, they induced a substantial loss in accuracy within the BOP pose estimation evaluation. However, segmentation methods based on DINOv2 features performed on a par with using Trellis meshes compared to using 2DGS and SVRaster meshes. This indicates that the latent space learned by Trellis, which is constructed using DINOv2 features, is robust for downstream methods that also rely on DINOv2 feature representations.

For cost-effective depth estimation, the MAST3R registration method was used to anchor the scene to a metric scale using far-view images. This strategy enabled robust and consistent object height estimation within an acceptable error margin, as empirically demonstrated in the grasping experiments. However, this behavior was only validated in a feature-rich indoor environment. It may not generalize to featureless scanning scenarios, where the lack of distinctive visual features could significantly degrade the accuracy of height estimation. Furthermore, when digitizing deformable objects, insufficient height accuracy is especially problematic because object deformations entail changes in shape and size, which propagate errors to downstream tasks. With respect to the observed error behavior, the magnitude of the height estimation error was found to increase for smaller objects. Currently, it is unclear whether this effect is primarily attributable to the limitations of MAST3R or to the specific procedure used to project the pointcloud onto the image plane. In particular, outlier 3D points may exert a disproportionately large influence on the final height estimate, thereby exacerbating the error for small objects.

Analysis of BOP scores reveals distinct performance characteristics for the different meshes. For the aforementioned Trellis objects, their strong performance with DINOv2-based segmentation methods, contrasted with the degraded performance of the pose estimation models, suggests that Trellis meshes may require pose estimation approaches explicitly leveraging DINOv2 features to achieve reasonable accuracy. The remaining segmentation results indicate that it is largely irrelevant whether 2DGS or SVRaster meshes are employed, although SVRaster produces meshes that are visually less accurate. In the context of pose estimation, the comparison of GT and CNOS masks with FoundationPose showed that this method is highly dependent on the selection of the best object mask, while no systematic difference between 2DGS and SVRaster meshes was observed. Additionally, the results of FoundationPose on CNOS masks are not representative, as the mask instance selection threshold could not be tuned appropriately. Finally, the SAM-6D BOP scores do not reflect the performance observed in the grasping experiments, indicating that the geometric matching procedure employed is not suitable for objects with inaccurate scale.

When comparing SAM and FastSAM as a backbone for generating segmentation hypotheses outside of the YCB-V BOP scenes, it is notable that FastSAM produced masks with lower boundary accuracy and precision. This is likely a primary factor in its inferior performance relative to SAM. Given that the edges of NVS-generated meshes exhibit

fine geometric detail, they provide valuable cues for classification, and any degradation in edge quality can negatively impact downstream tasks. Furthermore, the difference in performance observed for CNOS, which can be regarded as a simplified SAM-6D ISM, suggests that CNOS is more robust to inaccuracies in mesh geometry and object dimensions. A plausible explanation is that SAM-6D ISM incorporates a geometric matching score, which impedes segmentation when the NVS mesh is geometrically inaccurate, thus lowering the overall score.

The same behavior observed for SAM-6D ISM is also apparent in the complete SAM-6D pipeline. When GT meshes from the BOP benchmark are employed for pose estimation, the resulting performance is consistent with the reported BOP scores. However, when inaccurate meshes are used, particularly those with wrong scaling or absent underside geometry, the pose estimation quality degrades to the point of becoming practically unusable. This effect is especially pronounced for inverted object configurations, where the missing underside likely disrupts the point correspondence matching of SAM-6D PEM. In contrast, FoundationPose exhibited substantially greater robustness to mesh imperfections such as holes or scale inaccuracies. This robustness can be attributed to its methodological similarity to CNOS, building on a render-and-compare strategy guided by ML, rather than being heavily engineered around pointcloud information as in the case of SAM-6D.

The single-object grasping experiment conducted with CNOS/FoundationPose was successful, demonstrating that the NVS-generated meshes and their associated scaling are adequate for object manipulation. However, this was verified only in close-range scenarios. In addition, grasp performance for deformable, semitransparent, and small objects is close to the failure margin. Deformable and semitransparent objects, which are often poorly represented or absent in depth maps, are generally recognized as challenging for both pose estimation and robotic manipulation. The small object "toothbrush" is particularly affected by inaccurate scaling and the substantial absence of geometry on its underside, which is not captured in the reconstruction. This limitation can make the utilization of a static object pose in the input video a problematic strategy for NVS mesh generation.

Finally, multi-object detection in scenes with substantial overlap often results in frequent misclassifications and missing segmentations, primarily because it is difficult to determine the appropriate confidence thresholds required to filter out incorrect mask hypotheses. Consequently, CNOS exhibited reduced robustness in such scenarios. In contrast, the iterative strategy employed by FoundationPose is well-suited to overlapping scenes and still achieves reliable pose estimation when the mask is correctly classified, as it continuously refines the pose by rendering object hypotheses and matching them to the observed visual segmentation. This behavior indicates that the segmentation stage constitutes the main bottleneck in the overall pipeline.

In summary, the proposed automated mesh generation pipeline, which integrates Grounded-SAM, MAST3R-SfM, and 2DGS/SVRaster, consistently produces geometrically accurate meshes that allow accurate zero-shot object pose estimation with CNOS and FoundationPose. Furthermore, the proposed scaling method provides sufficient accuracy for robotic manipulation in real-world scenarios. However, outside of this setup, specifically grounding near-view object-centric images with far-view images, no robust ML depth

estimation method has been identified that can reliably replace hardware depth sensors. Finally, grasping experiments conducted with the HSR and the best-performing pose estimation pipeline CNOS/FoundationPose demonstrated that the NVS-generated meshes and their associated scaling are adequate for real-world object manipulation. However, this approach has limitations when applied to deformable, low-profile, or semi-transparent objects. These shortcomings are primarily attributable to scaling inaccuracies, incomplete mesh reconstruction, or unreliable HSR sensory data.

6 Outlook

The rapid advancement and black-box nature of ML-based camera registration, NVS, instance segmentation, and pose estimation methods demand ongoing empirical evaluation. The methods used in this thesis share this requirement and can be further tested or improved. Beginning with the registration and pointcloud step, the test image count was static; therefore, it would be interesting to evaluate how MAST3R-SfM and VGGT handle fewer images. Additionally, the objects that were tested were not fully transparent. Future research could therefore include objects composed of optically clear materials, such as glass or plastic.

The strong performance of VGGT on segmented images demonstrates its ability to register object masks captured from multiple viewpoints. This suggests the need for additional experiments involving dynamic data acquisition of real-world objects to evaluate the robustness of VGGT on realistic data characterized by a wider range of rotations, rather than on constrained image sets that are limited in pose diversity. If this does not work, another possible option would be to adapt Trellis to infer only the missing underside of an object and fuse it with the NVS-generated mesh. Furthermore, more recent methods that build on a similar strategy, such as VGGT, may outperform MAST3R-SfM, warranting a new evaluation. In parallel, approaches that directly output gaussian splats or SVRaster voxels are emerging as a promising alternative. A version of VGGT trained on metric data is likely to be transformative, as it would enable the joint estimation of depth and camera poses within a single forward pass of the ML model to guide NVS surface reconstruction with frame-consistent depth maps.

Given that the most effective methods of instance segmentation and pose estimation for NVS-generated meshes rely on a render-and-compare strategy, it may become feasible to eliminate the explicit mesh representation and instead render the objects directly using 2DGS or SVRaster. Alternatively, the SLAT representation of Trellis introduced a novel object representation paradigm, enabling a new domain for object generation. Within this domain, a newly developed pose estimation model based on SLAT or DINOv2 features could fundamentally transform the entire automatic mesh generation pipeline. However, the BOP evaluation and HSR grasping experiments in this thesis already show that NVS-generated meshes perform robustly, suggesting that the system is close to production readiness and warrants validation in an actual supermarket setting.

Bibliography

- [1] L. Custodio and R. Machado, „Flexible automated warehouse: A literature review and an innovative framework,“ *The International Journal of Advanced Manufacturing Technology*, vol. 106, no. 1, pp. 533–558, 2020.
- [2] P. Sun, H. Kretzschmar, X. Dotiwalla, *et al.*, „Scalability in perception for autonomous driving: Waymo open dataset,“ in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2020, pp. 2446–2454.
- [3] R. Sapkota, Y. Cao, K. I. Roumeliotis, and M. Karkee, „Vision-language-action models: Concepts, progress, applications and challenges,“ *arXiv preprint arXiv:2505.04769*, 2025.
- [4] A. A. Malik, T. Masood, and A. Brem, „Intelligent humanoid robots in manufacturing,“ in *Companion of the 2024 ACM/IEEE International Conference on Human-Robot Interaction*, 2024, pp. 20–27.
- [5] P. Ausserlechner, D. Habegger, S. Thalhammer, J.-B. Weibel, and M. Vincze, „Zs6d: Zero-shot 6d object pose estimation using vision transformers,“ in *2024 IEEE International Conference on Robotics and Automation (ICRA)*, IEEE, 2024, pp. 463–469.
- [6] S. Wang, V. Leroy, Y. Cabon, B. Chidlovskii, and J. Revaud, „Dust3r: Geometric 3d vision made easy,“ in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2024, pp. 20 697–20 709.
- [7] O. Ronneberger, P. Fischer, and T. Brox, „U-net: Convolutional networks for biomedical image segmentation,“ in *International Conference on Medical image computing and computer-assisted intervention*, Springer, 2015, pp. 234–241.
- [8] J. Lin, L. Liu, D. Lu, and K. Jia, „Sam-6d: Segment anything model meets zero-shot 6d object pose estimation,“ in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2024, pp. 27 906–27 916.
- [9] B. Wen, W. Yang, J. Kautz, and S. Birchfield, „Foundationpose: Unified 6d pose estimation and tracking of novel objects,“ in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2024, pp. 17 868–17 879.
- [10] B. Huang, Z. Yu, A. Chen, A. Geiger, and S. Gao, „2d gaussian splatting for geometrically accurate radiance fields,“ in *ACM SIGGRAPH 2024 conference papers*, 2024, pp. 1–11.
- [11] J. Xiang, Z. Lv, S. Xu, *et al.*, „Structured 3d latents for scalable and versatile 3d generation,“ in *Proceedings of the Computer Vision and Pattern Recognition Conference*, 2025, pp. 21 469–21 480.
- [12] C. Sun, J. Choe, C. Loop, W.-C. Ma, and Y.-C. F. Wang, „Sparse voxels rasterization: Real-time high-fidelity radiance field rendering,“ in *Proceedings of the Computer Vision and Pattern Recognition Conference*, 2025, pp. 16 187–16 196.

- [13] Y. Zuo, K. Kayan, M. Wang, K. Jeon, J. Deng, and T. L. Griffiths, „Towards foundation models for 3d vision: How close are we?“ *arXiv preprint arXiv:2410.10799*, 2024.
- [14] F. Gorschlüter, P. Rojtberg, and T. Pöllabauer, „A survey of 6d object detection based on 3d models for industrial applications,“ *Journal of imaging*, vol. 8, no. 3, p. 53, 2022.
- [15] S. Xu, S. Chen, R. Xu, C. Wang, P. Lu, and L. Guo, „Local feature matching using deep learning: A survey,“ *Information Fusion*, vol. 107, p. 102344, 2024.
- [16] V. Leroy, Y. Cabon, and J. Revaud, „Grounding image matching in 3d with mast3r,“ in *European Conference on Computer Vision*, Springer, 2024, pp. 71–91.
- [17] J. Wang, M. Chen, N. Karaev, A. Vedaldi, C. Rupprecht, and D. Novotny, „Vggt: Visual geometry grounded transformer,“ in *Proceedings of the Computer Vision and Pattern Recognition Conference*, 2025, pp. 5294–5306.
- [18] A. Bochkovskii, A. Delaunoy, H. Germain, *et al.*, „Depth pro: Sharp monocular metric depth in less than a second,“ *arXiv preprint arXiv:2410.02073*, 2024.
- [19] T. Yamamoto, K. Terada, A. Ochiai, F. Saito, Y. Asahara, and K. Murase, „Development of human support robot as the research platform of a domestic mobile manipulator,“ *ROBOMECH journal*, vol. 6, no. 1, pp. 1–15, 2019.
- [20] W. E. Lorensen and H. E. Cline, „Marching cubes: A high resolution 3d surface construction algorithm,“ in *Seminal graphics: pioneering efforts that shaped the field*, ACM, 1998, pp. 347–353.
- [21] M. Kazhdan, M. Bolitho, and H. Hoppe, „Poisson surface reconstruction,“ in *Proceedings of the fourth Eurographics symposium on Geometry processing*, vol. 7, 2006.
- [22] J. L. Schonberger and J.-M. Frahm, „Structure-from-motion revisited,“ in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 4104–4113.
- [23] M. Kazhdan, M. Chuang, S. Rusinkiewicz, and H. Hoppe, „Poisson surface reconstruction with envelope constraints,“ in *Computer graphics forum*, Wiley Online Library, vol. 39, 2020, pp. 173–182.
- [24] T. Hodaň, M. Sundermeyer, B. Drost, *et al.*, „Bop challenge 2020 on 6d object localization,“ in *Computer Vision–ECCV 2020 Workshops: Glasgow, UK, August 23–28, 2020, Proceedings, Part II 16*, Springer, 2020, pp. 577–594.
- [25] MANIBOT Project Consortium, *Manibot project*, Accessed: 2025-06-04, 2025. [Online]. Available: <https://manibot-project.eu/>.
- [26] V. N. Nguyen, T. Groueix, G. Ponimatkin, V. Lepetit, and T. Hodan, „Cnos: A strong baseline for cad-based novel object segmentation,“ in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2023, pp. 2134–2140.
- [27] M. Quigley, K. Conley, B. Gerkey, *et al.*, „Ros: An open-source robot operating system,“ in *ICRA workshop on open source software*, Kobe, vol. 3, 2009, p. 5.
- [28] M. Denninger, M. Sundermeyer, D. Winkelbauer, *et al.*, „Blenderproc,“ *arXiv preprint arXiv:1911.01911*, 2019.

- [29] T. Ren, S. Liu, A. Zeng, *et al.*, „Grounded sam: Assembling open-world models for diverse visual tasks,“ *arXiv preprint arXiv:2401.14159*, 2024.
- [30] B. Duisterhof, L. Zust, P. Weinzaepfel, V. Leroy, Y. Cabon, and J. Revaud, „Mast3r-sfm: A fully-integrated solution for unconstrained structure-from-motion,“ *arXiv preprint arXiv:2409.19152*, 2024.
- [31] T. Hodan, F. Michel, E. Brachmann, *et al.*, „Bop: Benchmark for 6d object pose estimation,“ in *Proceedings of the European conference on computer vision (ECCV)*, 2018, pp. 19–34. [Online]. Available: <https://bop.felk.cvut.cz>.
- [32] T. Yamamoto, T. Nishino, H. Kajima, M. Ohta, and K. Ikeda, „Human support robot (hsr),“ in *ACM SIGGRAPH 2018 emerging technologies*, ACM, 2018, pp. 1–2.
- [33] B. Kuipers, E. A. Feigenbaum, P. E. Hart, and N. J. Nilsson, „Shakey: From conception to history,“ *Ai Magazine*, vol. 38, no. 1, pp. 88–103, 2017.
- [34] H. P. Moravec, „The stanford cart and the cmu rover,“ *Proceedings of the IEEE*, vol. 71, no. 7, pp. 872–884, 1983.
- [35] O. Khatib, K. Yokoi, O. Brock, K Chang, and A. Casal, „Robots in human environments: Basic autonomous capabilities,“ *The International Journal of Robotics Research*, vol. 18, no. 7, pp. 684–696, 1999.
- [36] M. Hutter, C. Gehring, A. Lauber, *et al.*, „Anymal-toward legged robots for harsh environments,“ *Advanced Robotics*, vol. 31, no. 17, pp. 918–931, 2017.
- [37] G. Wiedebach, S. Bertrand, T. Wu, *et al.*, „Walking on partial footholds including line contacts with the humanoid robot atlas,“ in *2016 IEEE-RAS 16th International Conference on Humanoid Robots (Humanoids)*, IEEE, 2016, pp. 1312–1319.
- [38] T. M. Corporation, *Toyota shifts home helper robot into high gear with new developer community and upgraded prototype*, Accessed: 2025-10-25, 2015. [Online]. Available: <https://global.toyota/en/detail/8709541>.
- [39] ANYbotics, *Anybotics: Autonomous robotic inspection solutions*, Accessed: 2025-10-25, 2025. [Online]. Available: <https://www.anybotics.com/>.
- [40] B. Dynamics, *Boston dynamics – official website*, Accessed: 2025-10-25, 2025. [Online]. Available: <https://www.bostondynamics.com/>.
- [41] D. G. Lowe, „Object recognition from local scale-invariant features,“ in *Proceedings of the seventh IEEE international conference on computer vision*, Ieee, vol. 2, 1999, pp. 1150–1157.
- [42] N. Dalal and B. Triggs, „Histograms of oriented gradients for human detection,“ in *2005 IEEE computer society conference on computer vision and pattern recognition (CVPR’05)*, Ieee, vol. 1, 2005, pp. 886–893.
- [43] P. J. Besl and N. D. McKay, „Method for registration of 3-d shapes,“ in *Sensor fusion IV: control paradigms and data structures*, Spie, vol. 1611, 1992, pp. 586–606.
- [44] V. Lepetit, F. Moreno-Noguer, and P. Fua, „Ep n p: An accurate o (n) solution to the p n p problem,“ *International journal of computer vision*, vol. 81, pp. 155–166, 2009.
- [45] Z. Wang, „3d representation methods: A survey,“ *arXiv preprint arXiv:2410.06475*, 2024.

- [46] A. S. Gezawa, Y. Zhang, Q. Wang, and L. Yunqi, „A review on deep learning approaches for 3d data representations in retrieval and classifications,“ *IEEE access*, vol. 8, pp. 57 566–57 593, 2020.
- [47] J. Guan, Y. Hao, Q. Wu, S. Li, and Y. Fang, „A survey of 6dof object pose estimation methods for different application scenarios,“ *Sensors*, vol. 24, no. 4, p. 1076, 2024.
- [48] A. Kendall, M. Grimes, and R. Cipolla, „Posenet: A convolutional network for real-time 6-dof camera relocalization,“ in *Proceedings of the IEEE international conference on computer vision*, 2015, pp. 2938–2946.
- [49] Y. LeCun, Y. Bengio, and G. Hinton, „Deep learning,“ *nature*, vol. 521, no. 7553, pp. 436–444, 2015.
- [50] Y. Xiang, T. Schmidt, V. Narayanan, and D. Fox, „Posecnn: A convolutional neural network for 6d object pose estimation in cluttered scenes,“ *arXiv preprint arXiv:1711.00199*, 2017.
- [51] C. Wang, D. Xu, Y. Zhu, *et al.*, „Densefusion: 6d object pose estimation by iterative dense fusion,“ in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2019, pp. 3343–3352.
- [52] S. Peng, Y. Liu, Q. Huang, X. Zhou, and H. Bao, „Pvnet: Pixel-wise voting network for 6dof pose estimation,“ in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2019, pp. 4561–4570.
- [53] A. Bansal, K. Sikka, G. Sharma, R. Chellappa, and A. Divakaran, „Zero-shot object detection,“ in *Proceedings of the European conference on computer vision (ECCV)*, 2018, pp. 384–400.
- [54] J. Liu, W. Sun, H. Yang, *et al.*, „Deep learning-based object pose estimation: A comprehensive survey,“ *arXiv preprint arXiv:2405.07801*, 2024.
- [55] H. Wang, S. Sridhar, J. Huang, J. Valentin, S. Song, and L. J. Guibas, „Normalized object coordinate space for category-level 6d object pose and size estimation,“ in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2019, pp. 2642–2651.
- [56] V. N. Nguyen, S. Tyree, A. Guo, *et al.*, „Bop challenge 2024 on model-based and model-free 6d object pose estimation,“ *arXiv preprint arXiv:2504.02812*, 2025.
- [57] Y. He, Y. Wang, H. Fan, J. Sun, and Q. Chen, „Fs6d: Few-shot 6d pose estimation of novel objects,“ in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2022, pp. 6814–6824.
- [58] Y. Labbé, L. Manuelli, A. Mousavian, *et al.*, „Megapose: 6d pose estimation of novel objects via render & compare,“ *arXiv preprint arXiv:2212.06870*, 2022.
- [59] M. Oquab, T. Darcet, T. Moutakanni, *et al.*, „Dinov2: Learning robust visual features without supervision,“ *arXiv preprint arXiv:2304.07193*, 2023.
- [60] A. Caraffa, D. Boscaini, A. Hamza, and F. Poiesi, „Freeze: Training-free zero-shot 6d pose estimation with geometric and vision foundation models,“ in *European Conference on Computer Vision*, Springer, 2024, pp. 414–431.
- [61] N. Ravi, V. Gabeur, Y.-T. Hu, *et al.*, „Sam 2: Segment anything in images and videos,“ *arXiv preprint arXiv:2408.00714*, 2024.

- [62] Y. Liu, Z. Jiang, B. Xu, *et al.*, „Hippo: Harnessing image-to-3d priors for model-free zero-shot 6d pose estimation,“ *arXiv preprint arXiv:2502.10606*, 2025.
- [63] T. Lee, B. Wen, M. Kang, G. Kang, I. S. Kweon, and K.-J. Yoon, „Any6d: Model-free 6d pose estimation of novel objects,“ in *Proceedings of the Computer Vision and Pattern Recognition Conference*, 2025, pp. 11 633–11 643.
- [64] Z. Zhao, Z. Lai, Q. Lin, *et al.*, „Hunyuan3d 2.0: Scaling diffusion models for high resolution textured 3d assets generation,“ *arXiv preprint arXiv:2501.12202*, 2025.
- [65] L. Pan, D. Baráth, M. Pollefeys, and J. L. Schönberger, „Global structure-from-motion revisited,“ in *European Conference on Computer Vision*, Springer, 2024, pp. 58–77.
- [66] P. Weinzaepfel, T. Lucas, V. Leroy, *et al.*, „Croco v2: Improved cross-view completion pre-training for stereo matching and optical flow,“ in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2023, pp. 17 969–17 980.
- [67] R. Murai, E. Dexheimer, and A. J. Davison, „Mast3r-slam: Real-time dense slam with 3d reconstruction priors,“ in *Proceedings of the Computer Vision and Pattern Recognition Conference*, 2025, pp. 16 695–16 705.
- [68] J. Wang, N. Karaev, C. Rupprecht, and D. Novotny, „Vggsfm: Visual geometry grounded deep structure from motion,“ in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2024, pp. 21 686–21 697.
- [69] T. Zhou, R. Tucker, J. Flynn, G. Fyffe, and N. Snavely, „Stereo magnification: Learning view synthesis using multiplane images,“ *arXiv preprint arXiv:1805.09817*, 2018.
- [70] B. Mildenhall, P. P. Srinivasan, M. Tancik, J. T. Barron, R. Ramamoorthi, and R. Ng, „Nerf: Representing scenes as neural radiance fields for view synthesis,“ *Communications of the ACM*, vol. 65, no. 1, pp. 99–106, 2021.
- [71] Q. Xu, Z. Xu, J. Philip, *et al.*, „Point-nerf: Point-based neural radiance fields,“ in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2022, pp. 5438–5448.
- [72] B. Kerbl, G. Kopanas, T. Leimkühler, and G. Drettakis, „3d gaussian splatting for real-time radiance field rendering,“ *ACM Trans. Graph.*, vol. 42, no. 4, pp. 139–1, 2023.
- [73] S. Liu, Z. Zeng, T. Ren, *et al.*, „Grounding dino: Marrying dino with grounded pre-training for open-set object detection,“ in *European conference on computer vision*, Springer, 2024, pp. 38–55.
- [74] B. Curless and M. Levoy, „A volumetric method for building complex models from range images,“ in *Proceedings of the 23rd annual conference on Computer graphics and interactive techniques*, 1996, pp. 303–312.
- [75] J. T. Barron, B. Mildenhall, D. Verbin, P. P. Srinivasan, and P. Hedman, „Mip-nerf 360: Unbounded anti-aliased neural radiance fields,“ *CVPR*, 2022.
- [76] H. Aanæs, R. R. Jensen, G. Vogiatzis, E. Tola, and A. B. Dahl, „Large-scale data for multiple-view stereopsis,“ *International Journal of Computer Vision*, vol. 120, pp. 153–168, 2016.

- [77] M. M. Ibrahim, Q. Liu, R. Khan, J. Yang, E. Adeli, and Y. Yang, „Depth map artefacts reduction: A review,“ *IET Image Processing*, vol. 14, no. 12, pp. 2630–2644, 2020.
- [78] L. Yang, B. Kang, Z. Huang, X. Xu, J. Feng, and H. Zhao, „Depth anything: Unleashing the power of large-scale unlabeled data,“ in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2024, pp. 10 371–10 381.
- [79] Z. Wang, S. Chen, L. Yang, *et al.*, „Depth anything with any prior,“ *arXiv preprint arXiv:2505.10565*, 2025.
- [80] B. Wen, M. Trepte, J. Aribido, J. Kautz, O. Gallo, and S. Birchfield, „Foundation-stereo: Zero-shot stereo matching,“ *arXiv preprint arXiv:2501.09898*, 2025.
- [81] S. Chen, H. Guo, S. Zhu, *et al.*, „Video depth anything: Consistent depth estimation for super-long videos,“ *arXiv preprint arXiv:2501.12375*, 2025.
- [82] C. Yeshwanth, Y.-C. Liu, M. Nießner, and A. Dai, „Scannet++: A high-fidelity dataset of 3d indoor scenes,“ in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2023, pp. 12–22.
- [83] B. Mildenhall, P. P. Srinivasan, R. Ortiz-Cayon, *et al.*, „Local light field fusion: Practical view synthesis with prescriptive sampling guidelines,“ *ACM Transactions on Graphics (TOG)*, 2019.
- [84] A. Knapitsch, J. Park, Q.-Y. Zhou, and V. Koltun, „Tanks and temples: Benchmarking large-scale scene reconstruction,“ *ACM Transactions on Graphics (ToG)*, vol. 36, no. 4, pp. 1–13, 2017.
- [85] J. Reizenstein, R. Shapovalov, P. Henzler, L. Sbordone, P. Labatut, and D. Novotny, „Common objects in 3d: Large-scale learning and evaluation of real-life 3d category reconstruction,“ in *Proceedings of the IEEE/CVF international conference on computer vision*, 2021, pp. 10 901–10 911.
- [86] T. Wu, J. Zhang, X. Fu, *et al.*, „Omniobject3d: Large-vocabulary 3d object dataset for realistic perception, reconstruction and generation,“ in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2023, pp. 803–814.
- [87] M. A. Fischler and R. C. Bolles, „Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography,“ *Communications of the ACM*, vol. 24, no. 6, pp. 381–395, 1981.
- [88] G. Tolias, Y. Avrithis, and H. Jégou, „To aggregate or not to aggregate: Selective match kernels for image search,“ in *Proceedings of the IEEE international conference on computer vision*, 2013, pp. 1401–1408.
- [89] A. Kirillov, E. Mintun, N. Ravi, *et al.*, „Segment anything,“ in *Proceedings of the IEEE/CVF international conference on computer vision*, 2023, pp. 4015–4026.
- [90] X. Zhao, W. Ding, Y. An, *et al.*, „Fast segment anything,“ *arXiv preprint arXiv:2306.12156*, 2023.
- [91] Z. Qin, H. Yu, C. Wang, Y. Guo, Y. Peng, and K. Xu, „Geometric transformer for fast and robust point cloud registration,“ in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2022, pp. 11 143–11 152.

- [92] C. R. Qi, L. Yi, H. Su, and L. J. Guibas, „Pointnet++: Deep hierarchical feature learning on point sets in a metric space,“ *Advances in neural information processing systems*, vol. 30, 2017.
- [93] M. Deitke, D. Schwenk, J. Salvador, *et al.*, „Objaverse: A universe of annotated 3d objects,“ in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2023, pp. 13 142–13 153.
- [94] L. Downs, A. Francis, N. Koenig, *et al.*, „Google scanned objects: A high-quality dataset of 3d scanned household items,“ in *2022 International Conference on Robotics and Automation (ICRA)*, IEEE, 2022, pp. 2553–2560.

Erklärung

Hiermit erkläre ich, dass die vorliegende Arbeit ohne unzulässige Hilfe Dritter und ohne Benutzung anderer als der angegebenen Hilfsmittel angefertigt wurde. Die aus anderen Quellen oder indirekt übernommenen Daten und Konzepte sind unter Angabe der Quelle gekennzeichnet.

Die Arbeit wurde bisher weder im In- noch im Ausland in gleicher oder in ähnlicher Form in anderen Prüfungsverfahren vorgelegt.

Vienna, January 2026



Stefan Lechner